의료기기 GMP 적용 가이드라인
MDQMT-AG-2007-03

의료기기 소프트웨어 밸리데이션 가이드라인

(Guidelines for Software Validation of Medical devices)

의료기기본부
의료기기품질팀

Ⅰ. 소프트웨어 밸리데이션 접근 방법

1. 개요

“검증(Verification)”, “밸리데이션(Validation)”이라는 용어는 다양한 산업과 다양한 표준에서 다양한 방식으로 정의하고 사용하기 때문에 혼동이 발생할 수 있다. 어떤 면에서 “의료기기 소프트웨어 개발”은 "의료기기 소프트웨어를 위한 소프트웨어 밸리데이션"이라고 부를 수도 있다.

우리의 품질관리기준(식약청 고시 제2005-69호)은 국제기준 ISO 9000:2000과 조화를 이루어 검증과 밸리데이션을 분리된 별개의 용어로 사용한다. 그러나 다수의 소프트웨어 공학 관련 문헌에서는 검증과 밸리데이션을 번갈아 사용하기도 하고, “소프트웨어 검증, 밸리데이션 및 시험(VV&T)”을 단일 개념으로 사용하기도 한다. 이는 소프트웨어 산업에서 소프트웨어 VV&T의 목적은 소프트웨어가 의도한 환경에서 사용자가 사용하기에 적합한지 확인하기 위한 것으로 “의도한 플랫폼에서 소프트웨어를 최종적으로 시험하는 것”으로 정의하는 경우도 많기 때문이다.

이러한 정의 또는 다른 정의가 올바르다거나 잘못되니 것으로 간주하면 안 된다. 중요한 것은 안전하고 효과적이며 신뢰성 있는 소프트웨어를 개발하고 유지하기에 필요한 모든 활동과 프로세스 관리를 이해하는 것이다.

소프트웨어 밸리데이션은 의료기기의 유용성과 신뢰도를 증가시킴으로 결함 발생률, 리콜 및 시정조치 감소, 환자 및 의료기기취급자에 대한 위험 감소, 의료기기 제조업체에 대한 부담 경감을 위한 것으로, 소프트웨어 밸리데이션은 의료기기에 소프트웨어를 결합시킨 후 실제 또는 모의 사용환경에서 소프트웨어의 적절한 작동에 대한 점검을 포함하는 것이다. 용어 “소프트웨어 밸리데이션”을 사용하는 것은 안전하고 효과적인 소프트웨어를 보장하는 소프트웨어 개발의 모든 공식적인 측면을 정의하는 것일 수도 있다.

검증이란 “특정 요구사항이 충족되었음을 객관적인 증거의 제공을 통하여 확인하는 것”을 의미하며, 설계 밸리데이션은 “의료기기가 사용자의 요구 및 사용 목적에 일치함을 객관적 증거로 입증하는 것”을 의미한다. 소프트웨어의 밸리데이션은, 소프트웨어가 사용된 기능에 대한 사용자 요구와 사용목적이 소프트웨어와 일치함을 객관적 증거로 입증하는 것이다.

넓은 범위에서 소프트웨어의 밸리데이션은 소프트웨어에 의해 자동화된 의료기기의 성능과 특성에 대하여 모든 요구사항 및 사용자의 기대사항이 만족되었는지에 대한 "신뢰의 수준" 문제이다.

소프트웨어에 대한 검증과 밸리데이션은 제조업체에서 계속 시험할 수도 없고 어느 정도의 정보로 충분한지 파악하기가 곤란하기 때문에 어렵다. 일반적으로 모든 가능한 입력에 대해 소프트웨어를 시험하는 것은 불가능하며 소프트웨어 실행 시 발생할 수 있는 모든 가능한 데이터 처리 경로를 시험하는 것도 불가능하다. 이는 가장 간단한 소프트웨어를 제외하고 소프트웨어는 철저하게 시험할 수 없음을 의미한다. 소프트웨어 제품을 완전하게 시험하였다고 확신할 수 있는 하나의 시험 형식이나 시험방법이 있는 것도 아니다. 소프트웨어의 모든 코드를 시험하는 것으로 그 소프트웨어에 모든 필요한 기능이 구축되어 있다는 것과 소프트웨어의 모든 기능 및 코드에 대한 시험으로 소프트웨어가 완전하게 정확함을 의미하는 것은 아니다. 소프트웨어 시험결과 오류가 발견되지 않은 것이 소프트웨어에 오류가 없는 것으로 판명된 것은 아니다. 결과적으로 소프트웨어에 대한 시험은 피상적이라고 볼 수도 있다.

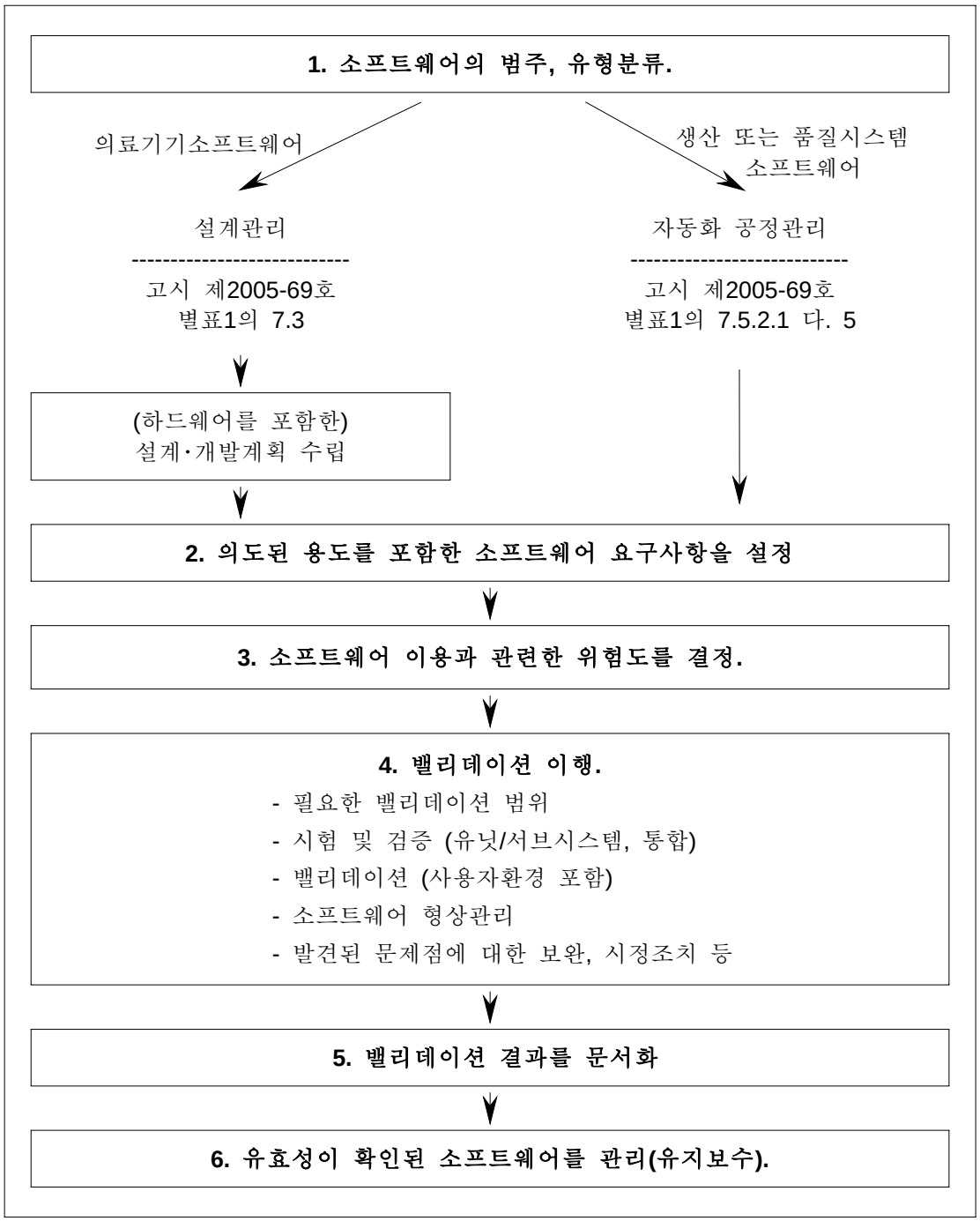
의료기기에 포함되는 소프트웨어 및 자동화공정 소프트웨어의 밸리데이션은 조직의 책임과 무관하게 어떤 기본 원칙들이 소프트웨어 밸리데이션 활동에 적용된다고 할 수 있다. 일반적으로 소프트웨어 밸리데이션 접근법은 그림 1에서 규정한 일련의 순서에 따른다.

기본적으로 의료기기에 포함되거나, 제조업체에서의 생산 및 품질시스템 운영에 사용되는 소프트웨어는 품질경영시스템(의료기기 제조・품질관리기준)과 위험관리시스템(ISO 14971)과 조화를 이루는 범위에서 개발 • 유지되고 사용되어야 한다.

설계에서의 소프트웨어 밸리데이션 활동은 소프트웨어에 대한 포괄적인 시험과 소프트웨어 개발 수명주기(Life-Cycle)[1]의 각 단계에서 이미 완료한 다른 검증 작업에 상당히 의존한다. 기획, 검증, 추적성, 형상관리 및 소프트웨어 엔지니어링의 다양한 요소들은 소프트웨어가 유효하다는 결론 도출에 도움을 주는 중요한 활동이다.

---

1) 요구사항의 정의에서부터 제조를 위한 릴리즈에 이르기까지 소프트웨어의 수명 전체를 포괄하는 개념적 구조로서,
- 소프트웨어 제품 개발에 관련된 프로세스, 활동 및 업무를 확인(identify).
- 활동과 업무 사이의 순서와 의존성을 설명.
- 특정 산출물의 완벽성을 검증할 수 있는 일정을 확인(identify).



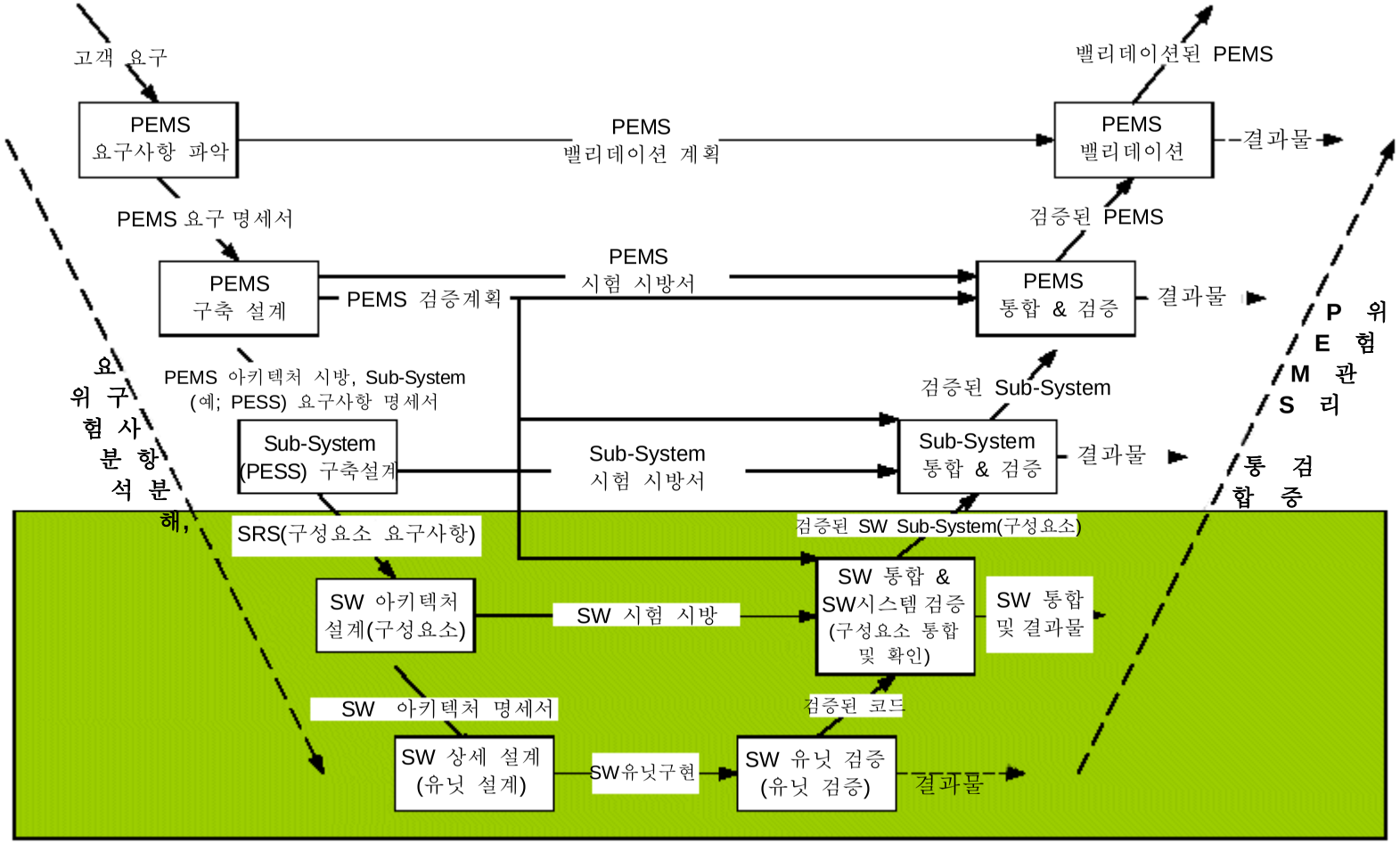
[그림 1] 소프트웨어 밸리데이션 접근법

철저하고 엄격한 소프트웨어의 검증과 밸리데이션은 단계적으로 구분하여 실시하는 것이 통례이다. 그림 2는 PEMS(프로그래밍이 가능한 전기 의료기기, Programmable Electrical Medical System)의 개발 과정과 현상을 설명하기 위한 V 모델로서, 소프트웨어 요구사항의 설정에서부터 소프트웨어 유닛이 소프트웨어 시스템에 통합할 때까지 적용되는 흐름을 도해한 것이다. 이 흐름은 위험관리 활동(프로세스)이 수반되어야 함을

의미한다. 소프트웨어의 위해요인(Hazard) 관리는 전체적인 의료기기 위험관리의 일부, 특히 안전에 대한 위험으로, 따로 취급할 수는 없다. 위해요인 관리의 목적은 식별한 기능성에 있어 소프트웨어의 결함, 소프트웨어의 고장이나 예상하지 못한 결과 등을 사전에 방지하기 위함이다. 따라서 소프트웨어의 위험관리와 관련하여 그 위험도의 등급에 따라 밸리데이션의 범위, 시험방법 및 그 수준(정도)는 달라질 것이다.

컴퓨터 및 자동화 설비 등 생산이나 품질시스템 운영에서 규정된 요구사항을 충족하기 위해 제품성능에 영향을 미치는 컴퓨터 소프트웨어를 사용하는 경우에는, 설계, 제조, 추적관리 등 품질시스템의 모든 측면에서 사용하는 소프트웨어의 설치가 정확하며 그 의도한 용도·목적에 따라 유효한 결과를 만들어낸다는 것을 입증하기 위하여 소프트웨어의 밸리데이션에 대한 문서화된 절차를 수립하고 최초 사용 전에 밸리데이션을 실시하여 소프트웨어가 의도한 바와 같이 일관되게 기능을 발휘함을 밸리데이션을 통하여 보증할 필요가 있다.

자동화 설비나 운용되는 소프트웨어의 밸리데이션에 필요한 정도는 소프트웨어 이용과 관련한 위험과 비례한다. 즉, 자동화 운영에 따라 노출되는 위험에 적절하며, 해당 소프트웨어 고유의 기능성뿐만 아니라 제조업체에서의 의도된 용도도 고려하여 밸리데이션에 필요한 시험 방법과 범위가 설정될 것이다.



Key:
상자 안의 내용은 대표적인 개발 수명주기 활동을 나타낸다.
실선 화살표는 활동의 입출력으로 전달되는 대표적인 산출물을 나타낸다.
점선 화살표는 위험관리 파일에 대한 산출물을 나타낸다.

* PEMS: 프로그램이 가능한 전기 의료기기
(Programmable Electrical Medical System)

[그림 2] 소프트웨어 개발 수명주기(V 모델)

2. 소프트웨어 분류 및 적용기준

본 가이드라인의 목적상, 밸리데이션이 필요한 소프트웨어는 의료기기의 사용목적에 따라 다음과 같이 분류한다.

a) 소프트웨어 자체가 의료기기이거나 또는 의료기기의 구성품, 또는 부속품으로 사용되는 소프트웨어
   - 의료기기 목적으로 독립적(stand-alone) 사용되는 소프트웨어
   - 의료기기의 기능에 영향을 주거나 조절하는 소프트웨어
   - 분석/모니터링이 목적인 의료기기에 의해 나온 환자의 데이터를 분석하는 소프트웨어

b) 자동화 설비(Automated Process Equipment) 및 품질시스템 운영에 사용되는 소프트웨어
   - FA 제어용 소프트웨어
   - PLC(Programmable Logic Controller)
   - 생산관리, 환경프로세스 관리, 포장, 라벨링, 추적성(traceability), 재고관리, SPC, 문서관리, 고객불만관리 등 품질시스템의 모든 측면에서 광범위하게 사용되는 소프트웨어

의료기기 사용목적에 따른 소프트웨어의 분류별 밸리데이션은 다음과 같이 적용한다.

・상기 2의 a)에 해당하는 소프트웨어는 “설계에서의 밸리데이션”을 적용

・상기 2의 b)에 해당하는 소프트웨어는 “공정에서의 밸리데이션”을 적용

3. 품질시스템과의 조화

의료기기 제조업체는 제조・수입및품질관리기준(식약청 고시)에 규정된 바와 같이, 수립된 품질시스템에서의 관련 절차에 따라 소프트웨어의 개발 및 유지를 위한 활동을 수행하여야 한다. 본 가이드라인에서는 소프트웨어의 밸리데이션에 대한 세부적 활동을, 이미 품질시스템이 확립되어 있거나 수립 중인 제조업체에서 그들의 절차에 통합시켜 채택하거나 확장하여 실행될 수 있도록 제시할 뿐이다. 이는 소프트웨어의 밸리데이션이 필요한 정도(범위)에 따라 제조업체에서 적절한 방식으로 실행할 수 있음을 의미하는 것이다. 소프트웨어를 전문 외주업체에 주문하여 개발할 경우, 소프트웨어 개발・유지 활동에서 설명한 절차/활동이 외주업체에서 적절하게 적용되는지에 대한 보증의 책임은 제조업체에 있다.

표 1은 제조 및 품질관리기준 별표 1의 요구사항과 본 가이드라인에서 언급된 소프트웨어 개발·유지활동과의 비교표이다.

제조 및 품질관리기준 [별표 1] 관련 조항		소프트웨어 개발·유지활동
7.3.1	설계 및 개발 기획	소프트웨어 개발 기획
7.3.2	설계 및 개발 입력	소프트웨어 요구사항 분석
7.3.3	설계및 개발 출력	소프트웨어 아키텍처 설계
		소프트웨어 상세 설계/코딩
7.3.4 7.3.5 7.3.6	설계 및 개발 검토 설계 및 개발 검증 설계 및 개발 밸리데이션	소프트웨어 유닛 이행 및 검증
		소프트웨어 통합 및 통합시험
		소프트웨어 시스템 시험(현장시험)
		소프트웨어 릴리즈
7.3.7 (7.3.5) (7.3.6)	설계 및 개발 변경 관리 (설계 및 개발 검증) (설계 및 개발 밸리데이션)	형상관리
		문제 발견 및 수정 분석
		수정 이행
		위험한 상황에 대한 소프트웨어 영향 분석
		위험관리 방법
		위험관리 방법의 검증
		소프트웨어 변경의 위험관리
7.5.3	식별 및 추적성	형상관리(형상식별, 형상상태)
		변경관리

[포 1] 품질시스템과 소프트웨어 개발·유지활동과의 관계

4. 위험관리(Risk Management)와의 조화

이미 언급한 바와 같이 의료기기 소프트웨어는 기본적으로 품질시스템과 위험관리 범위 안에서 개발·유지되어야 한다.
소프트웨어에 대한 위해요인(Hazard)의 관리는 의료기기의 전체적인 위험관리의 일부로서,

이를 따로 취급할 수는 없다. 소프트웨어가 포함된 의료기기의 안전성과 효과성의 확립에는 소프트웨어 사용 의도 및 소프트웨어 사용에 따른 허용할 수 없는 위험이 발생하지 않고 그 의도가 충족된다는 사실의 증명이 필요하다. 소프트웨어로 인하여 위해요인이 유발되거나 위해요인의 감소 여부를 판단하기 위하여 제조업체는 의료기기 위험관리를 이행할 수 있는 최소한의 절차, 활동 및 업무를 수행하여야 한다.

위험관리의 본질은 의료기기에서 발생할 수 있는 위해요인을 식별하고, 그 위해요인으로 인한 위험스러운 상황과 그에 대응하는 위험관리방법을 설정하는 것이다. 이렇게 설정된 위험관리 방법을 실행함으로 위험스러운 상황의 발생가능성과 심각성을 허용수준 이하로 감소시킬 수 있을 것이다.

이러한 위험관리활동은 이미 국제규격 ISO 14971에 규정되어 있다. ISO 14971에 정의된 위험관리는 특히 안전에 대한 위험을 취급한다. FTA(Fault Tree Analysis, 결함수분석), FMEA(Failure Mode and Effect Analysis, 고장형태 영향 분석) 등 몇몇 기법들은 위험관리의 체계적인 운영을 돕는데 사용된다. 소프트웨어의 밸리데이션 활동을 수행하는 자는 담당 분야인 소프트웨어의 위험관리를 위한 최소한의 요구사항을 이해하여야 할 것이다. 표 2에 소프트웨어 개발·유지활동에 있어 ISO 14971과의 관계를 표시한다.

의료기기 소프트웨어의 고장/결함은 성격상 체계적인 것이기 때문에, 이 고장 정도를 정밀하게 추정하기는 어렵다. 체계적인 고장 추정에 적합한 수준을 확실하게 설정하기가 곤란한 경우, 소프트웨어의 위해요인 관리의 엄격성, 관련 문서의 상세한 수준 및 경감의 적절성은 소프트웨어에 관련된 위해요인으로 발생할 수 있는 위해(Harm)의 심각성에 따라 결정하여야 한다. 소프트웨어의 부분은 필수적인 안전 기능을 직접 실현하지 않으며, 경감시킨 경우에도 안전에 관련된 소프트웨어에 부작용을 미칠 수 있으므로 소프트웨어에 의하여 발생할 수 있는 가장 심각한 위해요인을 개발 프로세스 및 관련 위해요인 경감의 정의에 있어 결정적인 요인으로 사용하여야 한다. 다음 3항에서 서술한 “소프트웨어의 안전성 분류”는 이러한 목적을 위하여 설정된 것이다.

중요한 것은 위해요인을 분석할 때, 하드웨어와 소프트웨어 위해요인을 포함하여 의료기기 사용목적과 관련된 모든 의료기기 위해요인을 고려하여야 한다. 식별된 각 위해요인에 대하여 위해요인의 심각성, 위해요인의 원인, 관리방법(예: 경고(alarm), 하드웨어 설계 등), 위험스러운 상황을 제거하거나 감소하기 위하여 취해지는 조치, 관리방법의 적절한 이행여부를 판단하기 위한 검증 활동 등은 문서화되어야 할 것이다.

ISO 14971:2000 조항	소프트웨어 개발・유지활동
4.1 위험 평가 절차	
4.2 의료기기 안전 관련 의도한 용도/목적 및 특성의 식별	
4.3 식별 또는 예측 가능한 위해요인의 식별	- 위험스러운 상황에 대한 소프트웨어의 영향 분석
4.4 각 위해요인에 대한 위험 추정	- 소프트웨어 안전성 분류
5 위험 평가	
6.1 위험 축소	
6.2 옵션 분석	- 위험관리 방법 정의
6.3 위험관리 방법의 구현	- 소프트웨어에 구현된 위험관리 방법 - 위험관리 방법 검증
6.4 잔류 위험 평가	
6.5 위험/장점 분석	
6.6 기타 발생 위해요인	- 상황(Event)의 새로운 발생순서 문서화
6.7 위험 평가 완료	
7 전체 잔류 위험 평가	
8 위험관리 보고서	- 추적성 문서화
9 생산 후 정보	- 소프트웨어 변경에 대한 위험관리

[표 2] 위험관리활동(ISO 14971)과의 관계

소프트웨어의 유지・관리과정에서 문제점 해결이나 업그레이드 등을 위하여 소프트웨어, 하드웨어, 기성품(OTS) 소프트웨어 또는 사용목적에 대한 변경(change)이 이루어지는 경우, 이러한 변경이 기존 시스템에 방해되는지 여부 및 다음의 판단을 위하여 분석하여야 한다.

・위해요인에 대한 새로운 경감이 필요한지 여부
・변경으로 새로운 위해요인이 추가 발생하는지 여부
・기존 위해요인에 대한 새로운 소프트웨어 원인이 발생하는지 여부

5. 소프트웨어 안전성 분류

소프트웨어 밸리데이션 활동에 앞서 제조업체는 의료기기 소프트웨어의 고장이나 잠재적인 결함으로 환자, 작업자 또는 기타 사람에게 미치는 위해(Harm)의 영향에 따른 소프트웨어의 안전성을 분류하고, 해당 소프트웨어의 가장 높은 안전성 등급을 결정하여야 한다.

표 3의 안전성 분류는 1절 4항에서 언급한 위험관리 산출물이기도 하다. 다만, 이러한 안전성 분류는 설계・개발과정의 초기 단계에서 결정되어야 할 것이다.

등 급	설 명
하(Minor)	고장, 잠재적 설계 결함으로 환자나 조작자에게 어떤 부상이나 신체적 피해가 발생할 가능성이 없는 경우
중(Moderate)	고장이나 잠재적 설계 결함이 직접적 원인이 되어 환자나 조작자에게 경상을 입힌 경우, 또는 고장 또는 잠재적 결함이 간접적인 원인으로 작용하여 부정확하거나 지연된 정보로 환자나 조작자에게 경상을 입힌 경우
상(Major)	고장이나 잠재적인 결함이 직접적인 원인으로 환자나 조작자에게 사망 또는 중상을 입힌 경우, 또는 고장이나 잠재적 결함이 간접적 원인으로 작용하여 부정확하거나 지연된 정보로 인하여 환자나 조작자에게 사망이나 중상을 입힌 경우

[표 3] 안전성 분류

용어 “안전성”은 “소프트웨어의 고장이나 잠재적인 결함으로 초래될 수 있는 위해의 심각성”으로 이해하여야 한다. 위험관리시스템(ISO 14971)에서 위험 평가는 식별된 각 위해요인(hazard)으로 인한 위해의 심각성과 발생확률(probability of occurrence)을 정성적 또는 정량적으로 산출하고 이 결과에 따라 위험관리를 수행토록 규정되어 있으나, 소프트웨어의 경우 발생확률/고장률은 가용 정보나 데이터를 활용하여 쉽게 예측할 수 없다는 특성을 갖고 있기 때문에 심각성에 근거한 소프트웨어 위험을 관리함이 보다 타당할 것이다.

본 가이드라인에서 제시하는 “안전성 분류”는 제조업체에서 수행하는 소프트웨어의 개발・유지활동 및 활동의 산출물(예: 소프트웨어 요구사항 명세서(SRS), 소프트웨어

설계 규격(SDS) 등)에 어떤 제약을 두려 함이 아니다. 다만, 제조업체는 요구되는 소프트웨어 안전성 분류, 소프트웨어의 규모 및 사용목적에 근거하여 그들이 수행하여야 할 개발・유지활동의 범위와 단계(세부 활동)를 설정하고, 이미 수립된 개발・유지 절차에 이 범위와 단계(세부 활동)를 통합 또는 확장하여 실행하도록 제안하는 것이다. 예를 들어 소프트웨어의 안전성 등급이 낮으면서 작은 규모의 소프트웨어인 경우는 아주 적은 밸리데이션 활동(시험・검증)으로도 충분히 입증될 수 있을 것이다.

Ⅱ. 설계에서의 소프트웨어 밸리데이션

본 가이드라인에서는 소프트웨어 개발에 대한 품질보증활동을 “소프트웨어의 수명주기 프로세스”와 조화될 수 있도록 “① 소프트웨어 개발 활동”과 “② 소프트웨어 유지보수(변경관리) 활동”으로 구분하고, 각 활동별로 이행할 필요성이 있는 구체적인 세부 활동을 표 4와 같이 설정한다. 이 활동 분류의 목적은 제조업체에게 항상 이러한 세부 활동들을 요구하려 함이 아니다. 앞서 몇 차례 전술한 바와 같이 제조업체가 스스로 설정한 소프트웨어 안전성 등급 및 개발·유지하려는 소프트웨어의 규모, 복잡성에 따라 세부 활동의 통합이나 단축도 가능하다.

구 분	세 부 활 동
소프트웨어 개발 활동	1. 소프트웨어 기획(Planning)
	2. 소프트웨어 요구사항 수립 및 평가
	3. 소프트웨어 아키텍처(architecture) 설계 및 검증
	4. 소프트웨어 상세 설계 및 유닛구현
	5. 소프트웨어 검증 및 밸리데이션 (V&V, Verification & Validation) - 유닛 시험(Unit Test) - 통합(Integration) 시험 - 시스템 시험 - 사용자 현장 시험(User Site Testing) - 밸리데이션 결과 보고서
	6. 소프트웨어 릴리즈(Release)
소프트웨어 유지보수 (변경관리) 활동	1. 변경 및 문제해결
	2. 문서화
	3. 형상관리

[표 4] 설계에서의 소프트웨어 밸리데이션 활동 개요

1. 소프트웨어 개발 활동

1.1 소프트웨어 기획(Planning) 활동

제조업체는 개발하려는 소프트웨어 시스템[2]의 범위, 규모 및 소프트웨어 안전에 적합한 개발 활동 진행을 위하여 소프트웨어 개발 계획을 수립하여야 한다. 이 개발 계획의 목적은 소프트웨어에서 발생하는 위험을 감소시키고 절차와 목적을 개발팀 구성원에게 알리며 의료기기 소프트웨어에 대한 품질 요구사항의 충족을 보장하는 개발 업무를 계획하는 것이다.

소프트웨어 개발 계획은 하드웨어를 포함한 의료기기 전체 개발 계획에 통합할 수도 있고, 독자적으로 "○○소프트웨어 개발 계획서"와 같이 고유의 명칭을 부여하여 수립하는 것도 무방하다. 수립된 개발 계획서는 자격이 있는 자가 검토 및 승인을 하도록 한다.

의료기기 소프트웨어 개발에 적용되는 방침과 절차를 이미 설정한 제조업체도 있을 수 있다. 이 경우 계획은 기존 방침과 절차를 참조할 뿐이다. 이와 달리 개발하려는 소프트웨어 고유 계획을 각각의 특정 활동으로 자세히 규정한 개발 계획서를 작성하는 제조업체도 있을 수 있다. 또한 의료기기 소프트웨어의 개발을 적절하게 조정한 계획도 있을 수 있다.

개발계획은 개발 수행에 필요한 상세 수준으로 지정하여야 하며, 위험에 비례하여 수립하는 것이 중요하다. 예를 들어 위험 수준이 높은 소프트웨어(시스템이나 항목)는 보다 엄격한 개발과정이 필요하며, 개발 계획서는 매우 상세하게 작성되어야 할 것이다.

일반적으로 소프트웨어 개발 계획서에는 소프트웨어 요구사항 분석, 아키텍처, 코딩, 통합, 시험, 설치 및 지원(훈련, 유지보수) 등 세부 활동을 구분하여 서술하며, 해당 내용을 포함한 독립적인 문서이거나, 다른 문서 일부 또는 몇 개의 문서와 연결시킨 형태이어도 무방하다.

개발 계획서에 언급될 내용은 다음과 같다.

a) 소프트웨어 시스템 개발에 대한 세부 활동
- 개발 Life Cycle에서 수행되어야 할 구체적 작업
- 수행되어야 할 작업의 주요 일정 계획
- 각 작업 방법, 절차 및 지원 활동

2) Software System, 특정 기능이나 일련의 기능을 수행하도록 구성된 소프트웨어 항목(item)을 통합한 집합체

· 작업 수용 기준
· 관련 자원 및 책임
- 조직 내의 팀(예: Sub-Project 부서, 엔지니어링 부서, A/S 등), 서로 다른 개인이나 그룹(예: 외주 개발업체, 의료기기 취급자 등)
- 개발을 위한 도구 및 기법, 그러한 도구 및 기법의 자격 부여 및 형상관리를 포함

b) 활동 및 (문서화를 포함한) 업무의 산출물
· 프로젝트 전체에 대한 입력 및 출력의 정의
- 입력 요구사항 확인을 평가할 수 있는 출력 정의 및 기준
· 각 작업을 위한 입력
· 각 작업에서 요구되는 출력
· 각 단계에서 소프트웨어 산출물

c) 소프트웨어 시스템 요구사항, 시험(V&V)
· 주요 품질요소(예: 신뢰성, 유지성(maintainability) 및 유용성(usability) 등)의 문서화
· 보안 및 사용자 요구사항의 문서화
· 소프트웨어 검증 및 밸리데이션 계획
- 검증 및 밸리데이션 작업 및 수용 기준
- 검증 및 밸리데이션 활동을 위한 일정 및 자원
- 정규 설계 검토 요구사항
· 기타 기술적 요구사항

d) 소프트웨어 관련 위험관리
· 소프트웨어 위험관리 프로세스의 활동과 업무 수행을 위한 계획
- 개발과 관련하여 발생 가능한 위험, 가정(assumptions), 의존도 및 문제점의 분석

e) 소프트웨어 형상 및 변경관리
· 형상관리 계획(Configuration Management Plan)

f) 개발 활동에서 확인(detected)되는 문제 취급을 위한 해결
· 문제점 해결 절차
· 그 밖의 지원 활동
- 우발적인 사건, 바이러스 방지를 포함한 보관, 백업 및 보존을 위한 절차

범용 컴퓨터의 사용이 보다 활성화됨에 따라 기성품(OTS: Off-the-shelf) 소프트웨어가 의료기기와 혼합되는 경향도 있다. 의료기기에 OTS 소프트웨어를 적용시키는 경우, 의료기기 사용 시에 지속되어야 하는 안전성과 유효성 문제점을 검토할 수 있도록 OTS 소프트웨어 관리에 필요한 활동과 업무를 포함하여 개발계획을 수립하여야 한다.

OTS 소프트웨어를 사용할 때의 문제점은 적용되는 의료기기마다 다르고, 해당 OTS 소프트웨어의 고장 발생시 환자, 조작자나 제3자에게 얼마나 영향을 미치는가에 따라서 다르다. 그러므로 OTS 소프트웨어는 의료기기 설계 과정의 일부가 되는 위험분석(risk analysis)을 고려하는 것이 특히 중요하다.

수립된 계발 계획은 개발의 진행에 따라 갱신하여야 한다. 개발계획이란 개발이 진행되는 동안에 계속 관찰하고 갱신하는 회귀적인 활동이다. 따라서 파악된 소프트웨어 시스템이나 위험관리 활동 결과를 반영할 수 있도록 계획을 갱신하여 개발과정을 적절히 관리하는 것이 중요하다.

1.2 소프트웨어 요구사항 수립 및 평가

소프트웨어 요구사항의 수립

소프트웨어가 지나치게 복잡하거나 사용자의 직관적인 예상과 반대되는 설계로 인한 동작/사용상의 오류는 감독기관/규제기관에 보고되는 가장 지속적이고 중대한 문제점 중의 하나이다.

소프트웨어를 제조업체에서 자체적 개발하거나 외주 개발업체(계약업체)에서 개발하거나 또는 기성품(OTS)로 구매하건 간에, 밸리데이션을 위하여서는 소프트웨어 요구사항을 분명히 문서화하는 것이 무엇보다 중요하다. 사용자 요구와 사용의도가 명확하지 않을 경우 소프트웨어 시스템이 그 요구와 목적에 일관되게 충족하는지의 확인은 사실상 불가능하다는 것이 일반적인 통념이다.

소프트웨어 요구사항이란 소프트웨어가 어떤(what) 기능을, 어떻게(how) 수행할 것인지에 대하여 설명한 것으로, 소프트웨어가 수행할 수 있는 논리적이고 물리적인 표현으로 변환된 것이다.

프로젝트의 복잡성 또는 다양한 기술적 책임의 수준을 지닌 사람들이 설계 정보를 명확하게 이해하도록 하기 위하여 소프트웨어 요구사항은 설계 및 세부 설계 정보에 대하여 높은 수준의 요약을 포함할 수 있다. 완성된 소프트웨어 요구사항은 개발 작업자가 임기응변식으로 설계 결정을 하는 필요성을 경감시키고 설계 목적 내에서 작업하도록 한다.

제조업체는 의료기기에 사용되는 시스템 요구사항에서 소프트웨어 요구사항을 결정하고 문서화하여야 한다. 소프트웨어 요구사항은 다음과 같은 사항들을 포함한다.

a) 물리적 특성, 기능 및 성능 요구사항
- 소프트웨어가 수행할 모든 기능, 신뢰성(reliability) 및 시간성(timing)
- 요구되는 응답시간(response time)
- 물리적 특성(예: 코드 언어, 기반(platform), 운영체제)
- 소프트웨어의 사용 환경(예: 사용되는 하드웨어, 메모리 크기, 처리장치, 네트워크)
- 제어논리(control logic)를 포함한 논리적 구조와 논리적 처리단계(예: 알고리즘)
- 하드웨어, 소프트웨어 및 사용 환경과의 관계를 포함하여 소프트웨어 시스템의 동작 의도를 설명한 소프트웨어 시스템의 개관(context)
- 복수 SOUP[3] 또는 기타 장비(예: 운영 드라이버, 다른 응용 소프트웨어)와의 호환성

b) 소프트웨어 시스템 입력과 출력
- 측정 또는 기록되는 변수(Data) 특성(예: 숫자, 영숫자, 형식)
- 소프트웨어가 수용할 수 있는 모든 범위, 한계 및 기본 값(value)
- 데이터베이스 요구사항 및 데이터 흐름도

c) 소프트웨어 시스템 내·외부의 연계성
- 모든 내부 소프트웨어와 시스템간의 인터페이스 및 모든 외부와 사용자 인터페이스의 정의
- 링크(communication links)(예: 소프트웨어 내부 모듈간 링크, 보조 소프트웨어와 링크, 하드웨어와 링크 및 사용자와의 링크)

d) 소프트웨어 구동(driven) 경보, 경고 및 운영자 메시지
- 오류 생성 요인과 처리방법
- 오류, 경보(alarm) 및 경고 메시지

e) 물리적 및 논리적인 보안 요구사항
- 중요 정보의 노출 위험에 관련되는 요구사항
- 보안 수단
  - 접근 권한자(authentication)
  - 인가
  - 통신 무결성

f) 사람으로 인하여 발생하는 오류와 훈련에 민감한 인체공학적 요구사항
- 수동 운영 지원

3) Software Of Unknown Provenance(기원이 확인되지 않은 소프트웨어)의 약어, 이미 개발되어 일반적으로 사용할 수 있거나, 의료기기에 채택할 목적으로 개발되지 않은 기성품(Off-the-shelf, OTS) 소프트웨어 또는 이전에 개발되어 개발 프로세스에 대한 적절한 기록이 없는 소프트웨어

- 인간-장치 상호작용(사용자가 시스템에 상호작용 할 수 있는 방법)
- 담당자에 대한 제약사항
- 사람의 집중적인 주의가 요구되는 영역

h) 운영 및 유지보수 현장에 인도되는 의료기기 소프트웨어의 설치 및 인수 요구사항

i) 운영 및 유지보수 방법과 관련된 요구사항
- 사용자 운영 및 실행과 관련하여 개발할 사용자 설명서
- 사용자 유지보수 요구사항

j) 관련 법규, 규제 요구사항

소프트웨어 요구사항에는 소프트웨어에서 수행되는 모든 안전 요구사항뿐만 아니라 소프트웨어 결함으로 발생할 수 있는 잠재적인 위해요인(hazard)도 명확히 규정하여야 한다. 또한 위험관리 활동에서 식별된 위험관리 요구사항이 소프트웨어 요구사항과 관련되는 경우 이들 요구사항은 위험관리방법이 소프트웨어 요구사항까지 추적이 가능하도록 소프트웨어 요구사항에서 구분할 수 있어야 한다.

소프트웨어 요구사항은 개발활동과 밀접하게 연결된 기술적인 위험관리활동에서 기인한다. 그러므로 제조업체는 하드웨어의 고장 및 잠재적인 소프트웨어 결함에 대해 소프트웨어에 이행된 위험관리방법, 특히 위험분석(risk analysis)을 의료기기 소프트웨어 요구사항에 포함시켜야 한다. 이 사항은 소프트웨어 개발을 시작할 때는 적용할 수 없을 수도 있으며, 소프트웨어가 설계되고 위험관리 방법이 계속 정의되는 동안 변할 수도 있다.

소프트웨어 요구사항에서의 “기능 및 성능 요구사항”과 관련하여 제조업체에서 OTS 소프트웨어를 사용하는 경우, 추가적으로 다음과 같은 소프트웨어 요구사항을 포함시켜야 한다.

a) 사용될 OTS 소프트웨어 제목과 제조업체

b) OTS 소프트웨어의 적절한 운영 지원에 필요한 시스템 하드웨어와 소프트웨어 (예; 프로세서 유형과 속도, 메모리 유형과 크기, 소프트웨어 시스템 유형, 디스플레이 요구사항)

c) 버전 수준, 릴리즈 날짜, 패치(patch) 번호 및 의료기기에 설치 시험할 업그레이드 명칭

d) OTS 소프트웨어 구성요소에 따른, 안전과 관련된 중요한 위험관리 방법

수립된 요구사항 평가

소프트웨어 요구사항이 충분하게 명시되어 있고 적절하다는 것을 확인하기 위하여, 소프트웨어 설계 작업을 시작하기 전에 소프트웨어 요구사항을 평가하여야 한다. 이는 소프트웨어 요구사항에 의하여 소프트웨어에 이행되는 내용이 결정되기 때문에 "요구사항 평가"는 매우 중요하다.

이런 평가를 수행하려면 무엇보다도 의료기기 소프트웨어가 허용되는 작동상태를 나타내는지, 완성된 의료기기 소프트웨어를 즉시 사용할 수 있는지에 대한 요구사항 확립이 필수적이다. 요구사항대로 이행될 수 있는지에 대하여 입증하려면 요구사항이 객관적으로 판단할 수 있는 방식으로 표현하여야 한다.

이와 관련하여 우리 품질관리기준(식약청 고시 제2005-69호, 7.3.2의 나)에서는 "요구사항은 완전하고, 불명확하거나 다른 요구사항과 상충되지 않을 것"과 "적정성을 검토, 승인"하도록 규정하고 있다. 제조업체에서 소프트웨어 요구사항을 결정한 이후 그 정확성, 시험가능성(testability) 및 명확성(clarity) 등에 대하여 다음과 같은 사항을 평가하고 문서화한다.

a) 소프트웨어 요구사항은 위험관리에 관련된 위해요인(hazard)에 대하여 적절하며, 결함의 허용오차, 안전성 및 보안 요구사항은 완벽하고 정확하다;

b) 요구사항 내에서 내부적으로 서로 모순되거나, 상충하지 않는다.

c) 모든 수행 요구사항은 모호한 표현이 아닌 명확한 용어로 서술되어야 한다.

d) 기능 및 성능 기준에 충족여부를 판단하기 위한 시험 사항은 명확하며 적절하다.

- 소프트웨어 설계 가능성 및 기능의 할당
- 모든 요구사항은 측정 가능하거나 객관적으로 검증할 수 있는 용어로 설명.
- 검증 및 밸리데이션의 가능성(testability), 시기 및 허용 기준:
  - 유닛[4] 시험, 통합시험, 시스템시험

e) 시스템 요구사항이나 검증 및 밸리데이션을 위한 시험, 이행된 위험관리방법까지 추적할 수 있어야 한다.

- 기기 요구조건에 대한 추적성
- 위해요인 경감 요구사항의 적합성
- 위해요인 분석과 검증/밸리데이션 시험간의 추적성

4) Software Unit, 다른 항목(item)으로 다시 분류할 수 없는 소프트웨어 항목, 소프트웨어 유닛은 소프트웨어 형상관리 또는 시험 목적에 사용할 수 있음.

자주 혼동이 발생하는 것을 방지하려면, 고객 요구(Need), 설계 입력, 소프트웨어 요구사항(Requirement), 소프트웨어 기능 명세서(Functional Specification) 및 소프트웨어 설계 명세서(Design Specification)를 명확하게 구별(이해)하는 것이다.

- 설계 입력이란 고객 요구를 의료기기 요구사항으로 문서화한 것이며,
- 소프트웨어 요구사항이란 소프트웨어가 고객 요구와 설계 입력을 충족하도록 공식적으로 문서화된 것이다.
- 소프트웨어 기능 명세서는 소프트웨어가 요구사항을 충족하기 위하여 수행해야 할 작업을 상세하게 정의한 것으로, 소프트웨어 요구사항에 포함시키는 경우가 많으며 다른 대안도 가능하다.
- 소프트웨어 설계 명세서는 요구사항 및 기능 명세서의 구현을 위하여 소프트웨어를 설계하고 분해하는 방법을 정의한 것이다.

일반적으로 소프트웨어 요구사항, 기능 명세서 및 설계 명세서는 하나 이상의 문서로 작성한다.

1.3 소프트웨어 아키텍처(architecture) 설계 및 검증

소프트웨어 아키텍처 설계

제조업체는 의료기기 소프트웨어 요구사항을 소프트웨어 구조(structure)로 설명하며 소프트웨어 항목(item)[5]을 식별하는 아키텍처로 변환하여야 한다. 소프트웨어 항목과 소프트웨어 항목 외부의 구성요소(소프트웨어와 하드웨어)간의 연계성과 소프트웨어 항목간 연계성을 위한 아키텍처를 개발하고 문서화하여야 한다.

이 활동을 수행하려면 제조업체가 소프트웨어의 주요 구조적 구성요소, 외부에서 확인할 수 있는 특성 및 특성간의 관계를 정의하여야 한다. 어느 한 구성요소의 작동상태가 다른 구성요소에 영향을 줄 경우 그 작동상태를 소프트웨어 아키텍처에 서술하여야 한다. 이 서술은 소프트웨어 이외의 의료기기 구성요인에 영향을 미칠 수 있는 작동상태에 특히 중요하다. 다른 구성요인에 영향을 줄 수 있는 구성요인의 작동상태를 이해(파악)하지 못하면 시스템이 안전하다고 언급하는 것은 거의 불가능에 가깝다. 소프트웨어 아키텍처는 소프트웨어 요구사항의 정확한 이행에 필수적이다. 소프트웨어 아키텍처는 모든 소프트웨어 요구사항이 식별된 소프트웨어 항목으로 이행할 수 없는 한 완성되지 않는다. 소프트웨어 설계와 이행은 아키텍처에 따라 결정되기 때문에 아키텍처를 확인하여야 하며, 일반적으로 아키텍처의 확인은 기술 평가에 의해서 이루어진다.

소프트웨어 항목이 SOUP로 식별된 경우 제조업체는 의도한 사용에 필요한 SOUP 항목에 대한 기능, 성능 요구사항 및 적절한 운영 지원에 필요한 시스템 하드웨어와 소프트웨어(프로세서 유형과 속도, 메모리 유형과 크기, 시스템 소프트웨어 유형, 통신 및 디스플레이 소프트웨어 요구사항이 포함하여)를 명시하여야 한다.

소프트웨어 아키텍처 검증

제조업체는 다음을 검증하고 문서화하여야 한다.

a) 소프트웨어 아키텍처가 위험관리에 관련된 요구사항을 포함해 시스템 및 소프트웨어 요구사항을 이행한다.

b) 소프트웨어 아키텍처는 소프트웨어 항목 간의 연계성과 소프트웨어 항목 및 하드웨어 간의 연계성을 지원할 수 있다.

c) 의료기기 아키텍처는 모든 SOUP 항목의 적절한 운영을 지원한다.

1.4 소프트웨어 상세 설계 및 유닛구현

소프트웨어 상세 설계

제조업체는 소프트웨어 아키텍처(architecture)를 소프트웨어 유닛(Unit)으로 표현될 때까지 개량하고, 각 소프트웨어 유닛에 대한 상세 설계를 수행하고 검증하여야 한다. 또한 제조업체는 소프트웨어 유닛 및 외부 구성요인(하드웨어나 소프트웨어) 사이의 연계성 및 소프트웨어 유닛 사이의 연계성을 위한 상세 설계를 수행하고 문서화한다.

소프트웨어 유닛을 단일 기능, 프로그램이나 모듈로 간주하는 경우가 많지만 이러한 견해가 항상 정확한 것은 아니다. 본 가이드라인에서는 소프트웨어 유닛을 더 작은 항목으로 분해할 수 없는 소프트웨어 항목으로 정의한다. 소프트웨어 항목은 새로운 소프트웨어 항목 중 극히 일부만 원래 소프트웨어 항목의 안전성 관련 요구사항을 이행하도록 분해할 수 있다.

이 활동을 수행하려면 아키텍처에서 정의한 소프트웨어 항목과 연계성을 다시 정의하여 소프트웨어 유닛과 연계성을 작성하여야 한다. 제조업체는 소프트웨어 유닛의 상세 수준을 정의하고 알로기즘, 데이터 표현, 다양한 소프트웨어 유닛 사이의 연계성 및 소프트웨어와 데이터 구조사이의 연계성을 명시한다. 또한 상세 설계는 소프트웨어 제품의 패키지구성에도 관련한다. 소프트웨어 유닛을 정확히 실현할 수 있도록 각 소프트웨어 유닛의 연계성을 문서화하여야 한다.

구현은 상세 설계에 따라 결정되므로 활동을 완료하기 전에 상세 설계를 검증하여야 한다. 상세 설계의 검증은 일반적으로 기술 검토에 의해 이루어진다. 제조업체는 소프트웨어 상세 설계가 다음과 같은지 검증하고 문서화하여야 한다.

a) 소프트웨어 아키텍처를 이행한다.

b) 소프트웨어 아키텍처와 상충하지 않는다.

안전성에 중요하다고 판단될 수 있는 설계 특성 역시 검증하여야 한다. 그러한 특성들의 몇 가지 예는 다음과 같다.

- 의도된 사상(Event), 입출력, 논리적 흐름, CPU 할당, 메모리 자원의 할당 및 오류와 예외 정의, 오류와 예외 확인(isolation)과 오류 복구의 이행
- 위험한 상황을 유발할 수 있는 모든 결함이 사상과 전환으로 취급되는 기본 상태의 정의변수, 메모리 관리의 초기화
- 위험관리 방법에 영향을 줄 수 있는 콜드 리셋과 웜 리셋, 대기 및 기타 상태 변경

소프트웨어 유닛구현

이 활동을 수행하려면 제조업체는 소프트웨어 유닛에 대한 코드를 작성하고 검증해야 한다.

각 유닛에 대한 코드가 정확히 구현되지 못할 경우 소프트웨어는 의도된 대로 기능을 수행하지 않기 때문에 제조업체는 코드를 검증하여야 한다. 소스 코드 평가는 코드 실행 전에 오류를 파악할 수 있는 매우 효과적인 방법이다. 이는 격리된 상황에서 각 오류 시험을 가능하게 하고 추후 소프트웨어의 동적(dynamic) 시험에 집중할 수 있도록 해준다.

코딩은 세부적인 설계사양을 소스 코드(source code)로서 수행되는 소프트웨어 활동으로, 코딩은 설계사양에 대한 분해(decomposition)가 종료되고 실행 가능한 소프트웨어의 구성(composition)이 시작되는 지점을 의미하며 소프트웨어 개발 프로세스 중 가장 낮은 수준의 개념(abstraction)이다.

소프트웨어 코딩의 경우 다음과 같은 요구사항을 취급하여야 한다.

- 코드가 정확하며 요구사항에 부합.
- 코드가 안전 요구사항을 정확히 실현.
- 코드가 소프트웨어 유닛의 상세 설계에 문사화한 연계성과 일관성 유지.
- 사용 코딩 방법과 표준이 적절.
- 확립한 프로그래밍 절차나 코딩 표준과의 일치

· 사용 검증 전략의 적합성
· 시험 절차의 정확성
· 소프트웨어 통합 및 시험 가능성
· 유지보수 가능성

1.5 소프트웨어 검증 및 밸리데이션(V&V, Verification & Validation)

개요

전제 작업(예: 코드 검사)이 성공적으로 완료되면 소프트웨어에 대한 시험은 시작된다. 예를 들어 소프트웨어 제품의 시험은 표 5에서와 같이 유닛, 통합 및 시스템 시험의 단계로 체계화될 수 있다. 이는 유닛시험에서 시작하고 소프트웨어 시스템시험에서 종결됨을 의미한다. 통합시험과 시스템 시험을 단일 활동으로 결합할 수도 있다.

	유닛 시험	통합시험	시스템 시험	현장시험(필요시)
시험 대상	Black Box, White Box 경계조건	통합 유닛 인터페이스 수행기능	시스템 요구사항 및 기능, 성능	주요 기능, 문서 및 절차
완료 시점	완료된 유닛에 대해 오류가 없다고 판단된 경우	통합 관련 유닛에서 오류가 없다고 판단된 경우	모든 시스템 요구기능이 만족된 경우	사용자가 만족
시험 도구	시험사례, 커버리지 도구, 정적 분석 등	빌드 검증 시험사례	시험사례, Data생성기/비교기, 성능모의	인수시험, 시험사례, 시험 비교기
오류 보고	유닛시험 또는 개발 담당자가 처리	발견된 오류/버그는 추적가능토록 기록	발견된 오류/버그는 추적가능토록 기록	발견된 오류/버그는 추적가능토록 기록

[표 5] 소프트웨어 시험 단계

· 유닛 시험은 보조 프로그램(sub-program) 기능에 대한 초기 시험과 시스템에서 확인하기 곤란한 기능 검사가 확실히 이루어짐에 초점을 두고 있다.
· 통합시험은 유닛 내·외부 인터페이스에서의 데이터 및 관리의 이전에 초점을 두고 있다.
· 시스템 시험은 소프트웨어 요구사항을 고려하여 완성된 기능 및 성능을 검증하는 소프트웨어의 신뢰성 보증에 초점을 두고 있다.

소프트웨어에 대한 검증과 밸리데이션에 사용되는 시험은 화이트박스(White Box) 시험[5], 블랙박스(Black Box) 시험[6], 점증적 시험(Incremental Test), 연쇄식 시험(Thread

Test)으로, 어떻게 시험할 것인지에 대한 관점으로 분류할 수 있별 장·단점 비교는 표 6와 같다.

	시험단계	시험방법	장점	단점
White Box 시험	유닛시험	구조시험, 루프시험	• 프로그램 내부 모두 시험 가능 • 기능시험에서 확인불가한 부분도 시험가능	• 단순 소스코드 검사로 실제 사용자요구 수준을 만족하는지 알 수 없음 • 규모가 큰 경우 수행곤란
Black Box 시험	유닛 시험을 포함한 큰 규모의 시험	동등분할7), 경계값 분석8), 원인-결과도 오류 예측9)	• 실제 실행환경에서 시험 • 내부구조를 몰라도 시험 가능	• 잘못된 로직으로 발생가능성 있는 오류검출이 어려움 • 서로 다른 곳에 사용하고 있는 동일한 모듈에 대해 중복 시험할 수 있음
점증적 시험	모듈과 컴포넌트에 대한 유닛 시험	상향식 하향식	• 오류의 조기발견이 용이 • 문제점발견을 위하여 원시코드를 많이 볼 필요 없음 • 새로 통합하려는 모듈에 관심을 집중할 수 있음 • 철저한 시험 수행 가능	• 드라이버, 스텁 필요 • 수행시간이 많이 소모
연쇄식 시험	통합시험 초기에 중요 기능 시험		• 최선의 통합방법 • 시스템의 중요 기능을 담당하는 모듈부터 통합시험 • 초기에 시스템 골격을 보여주고 사용자의 의견을 받아 수정가능	

[표 6] 시험별 장단점 비교표

그 외 시험 기법에는 회귀(Regression) 시험, 비버깅(Bebugging) 및 뮤테이션(Mutation) 시험 등이 있다. 회귀시험은 '시스템 구성부품에 대한 변경이 기능성, 신뢰성

5) 제품의 수행기능을 알고 있을 때 프로그램의 내부구조 및 특성을 고려하지 않고, 각 기능의 완전한 동작을 증명해 보이는 것으로 소프트웨어 인터페이스에서 실시되는 시험을 의미한다. 즉, 소프트웨어 기능의 작동과 입력 적합성, 출력 정확성, 자료파일과 같은 외부 정보의 무결성을 입증하는 것이다. 이는 소프트웨어 내부 논리적 구조는 고려하지 않고 기본적인 시스템 모델의 기능을 시험하는 것이다.

6) 제품의 내부수행 동작을 알고 있을 경우 명세화에 따른 내부수행 동작을 시험하는 것으로 순차적이고 세부적이며 소프트웨어의 논리적 경로, 조건 반복과 같은 부분들을 시험하는 것이다. 특히 발생가능한 모든 경우 및 경로의 시험을 소모적 시험(Exhaustive test)이라 하는데 이는 현실적으로 어려워 중요부분을 대상으로 White-box test가 이루어진다.

7) Equivalence Partitioning, 프로그램 관련 명세서로부터 시험 요소를 정하고, 식별된 각 요소에 따른 적합한 동치집합과 부적합한 동치집합을 식별하여 시험케이스를 설계.

8) Boundary-Value Analysis, 입출력의 경계치 조건을 중적으로 분석하여 시험 케이스를 설계.

9) Error Guessing, 특정 시험 기법과는 관계없이 직관과 경험에 의해 발생 확률이 높은 오류를 추측하고, 이러한 오류를 검출하는 시험 케이스를 설계 오류 발생 확률이 높은 상황을 주시

또는 성능에 부정적 영향을 미치지 않으며 추가 결함이 발생하지 않음을 판단하기에 필요한 시험'으로 오류 발견시 문제점 수정을 확인하는 것으로 수정한 결과로 그 기대치를 만족하는지 또는 수정한 부분으로 다른 부분에 영향을 미치어 또 다른 오류를 발생시키지 않는지 확인하는 것이고, 비버깅 시험이란 의도적인 오류를 포함하여 시험이 얼마나 효과적으로 오류를 감지하는지 확인하는 것이며, 뮤테이션 시험은 의도적으로 프로그램에 약간의 수정을 가해 시험이 얼마나 효과적으로 수정한 것을 오류로 판단하는지 확인한다.

실제 작업환경에서는 이러한 모든 소프트웨어 시험 기법을 사용하는 것은 필요하지 않다. 보다 중요한 것은 어떤 항목은 가중치를 높이고 어떤 것은 제외할 것인지를 정하는 것 즉, 시험 관리, 시험 계획, 시험 결과 분석 등의 시험 절차를 수립하는 것이다.

유닛 시험(Unit Test)

각 유닛에 대한 코드가 정확하지 못할 경우 의료기기 소프트웨어는 의도된 대로 기능을 수행하지 않기 때문에 제조업체는 코드를 검증하여야 한다. 소스코드에 기초한 이러한 평가는 코드 실행 전에 오류를 파악할 수 있는 매우 효과적인 방법으로 white-box" 시험으로도 알려져 있다. 이는 소스코드, 구체적인 설계 규격과 다른 개발 문서로부터 획득된 지식에 기초하는 시험사례(Case)를 명확히 한다. 구조적인 시험은 프로그램이 실행될 경우 한 번도 수행되지 않는 dead 코드를 명확히 할 수 있다. 구조적인 시험은 유닛 시험으로 우선적 수행되어지지만 소프트웨어 시험의 다른 수준까지 확대될 수는 없다.

이러한 코드평가는 격리된 상황에서 각 오류 시험을 가능하게 하고 추후 소프트웨어의 동적 분석(dynamic Analysis)에 집중할 수 있도록 해준다. 동적 분석도 밸리데이션의 중요한 일부이지만 이것만으로 시스템 성능을 완벽하고 정확하게 입증하기는 사실상 불가능할 것이다. 어떤 시스템의 밸리데이션 결과는 시스템 개발과정의 전체에 걸쳐 수행되는 다수의 검증 단계로도 입증된다. 이들 단계로는 문서 및 코드 검사, 웍스루 및 기술 검토와 같은 정적(static) 분석이 있다. 이 활동과 결과에 관한 정보를 이용할 수 있을 경우 중점적으로 다뤄야 할 시험들을 용이하게 구분할 수 있으며, 소프트웨어가 사용자의 요구와 사용목적을 충족하는지 검증하기 위하여 사용자의 현장에서 수행될 시스템 차원에서의 기능시험의 양을 줄일 수도 있다.

평가를 위한 원시코드를 입수하지 못할 경우는 다음의 항목들을 이행함으로써 소프트웨어 구조의 무결성에 대한 유효성을 추론하여야 한다.

- 유닛 사용 이력을 조사

1) 이미 알려진 프로그램 한계의 파악
2) 다른 사용자 경험들을 검토 평가
3) 알려진 소프트웨어 문제점과 해결방법의 파악

- 공급자의 소프트웨어 개발 활동을 평가하여 현행 표준에 대한 적합성을 판단; 이 평가는 최종사용자 조직 또는 신뢰할 수 있으며 자격을 가진 제3자가 수행하는 평가 결과에 따르는 것이 바람직하다.

해당되는 경우, 제조업체는 규모가 큰 소프트웨어 항목으로 통합하기 전 소프트웨어 유닛에 대한 허용기준을 확립하고, 소프트웨어 유닛이 허용기준을 충족하는지 검증하여야 한다. 허용 기준의 예는 아래와 같다.

- 소프트웨어 코드가 위험관리 방법을 포함하여 요구사항을 구현하는가?
- 소프트웨어 코드가 소프트웨어 유닛의 상세 설계에 문서화한 연계성과 상충하지 않는가?
- 소프트웨어 코드가 프로그래밍 절차나 코딩 표준과 일치하는가?

필요한 경우, 제조업체는 다음과 같은 추가 허용 기준을 포함시켜야 한다.

- 적절한 사상(Event)의 순서
- 데이터 및 관리 흐름의 정확성
- 계획한 자원 할당
- 결함 취급(오류 정의, 확인 및 복구)
- 변수의 초기화
- 자체 진단
- 메모리 관리 및 메모리 용량초과
- 경계(boundary) 조건

제조업체는 소프트웨어 유닛 검증을 수행하고 그 결과를 문서화하여야 한다.

통합(Integration) 시험

이 활동을 수행하려면 소프트웨어 유닛을 집합 소프트웨어 항목으로 통합하기 위한 계획을 설정하여야 한다. 이 통합계획에는 OTS 소프트웨어 구성요소를 모두 포함하여 접근방식, 책임 및 순서가 포함되어야 하며, 전체적으로 통합된 유닛을 통합 계획에 따라 시험하여, 소프트웨어가 의도한 바와 같이 작동하는지 확인하여야 한다.

제조업체는 통합계획에 따라 소프트웨어 통합에 관한 다음 사항을 검증하고 기록한다. 이 검증은 항목(item)이 의도한 바와 같이 기능을 수행하는지에 대한 검증이 아닌, 계획에 따라 항목(item)이 통합되었는지 검증하는 것이다.

- 소프트웨어 유닛이 소프트웨어 항목과 소프트웨어 시스템으로 통합여부
- 하드웨어 항목, 소프트웨어 항목 및 시스템의 수동 운영(예; 사람 연계성, On-Line 도움말 메뉴, 음성 식별, 음성관리)을 위한 지원이 시스템에 통합여부

통합에 대한 접근방식은 비점증적(non-incremental) 통합에서부터 점증적 통합에 이르기까지 다양할 수 있다. 비점증적 통합 방법을 사용하여 프로그램을 구축한 경우 제조업체는 모든 유닛을 각각 시험을 실시하고 유닛시험 종료 후 모든 유닛들을 동시에 통합하여 시험하는 빅뱅(Big-bangt) 시험을 수행하며, 점증적(incremental) 통합 방법을 사용한 경우 제조업체는 이전에 통합한 소프트웨어의 다른 곳에서 문제점이 발생하지 않음을 확실하게 입증하기에 적합한 충분한 회귀분석[10] 및 회귀시험을 수행하여야 한다.

소프트웨어 통합시험은 소프트웨어 항목의 내·외부 연계성에서 데이터 전달과 관리에 집중한다. 외부 연계성은 운영체제 소프트웨어가 포함된 다른 소프트웨어와 의료기기 하드웨어와의 연계성을 의미한다.

소프트웨어 통합시험의 경우 제조업체는 통합된 소프트웨어 항목이 의도한대로 기능을 발휘하는지 확인한다. 통합시험 및 소프트웨어 시스템 시험은 단일 계획/활동으로 결합할 수도 있으며, 시험에 고려하여야 할 예는 다음과 같다.

- 소프트웨어에 요구되는 기능성
- 위험관리 방법의 이행
- 지정한 시점 및 기타 작동 상태
- 내부 및 외부 연계성의 지정 기능
- 예측할 수 있는 오용을 포함해 비정상 조건에서 시험

소프트웨어 통합시험은 시뮬레이션 환경, 실제 대상 하드웨어나 전체 의료기기에서 수행할 수 있다. 통합시험 조건에는 정상적인 응답뿐 아니라 예측되는 스트레스가 높은 최악 조건(예: 많은 수의 사용자가 동시에 네트워크에 접속 등)도 포함되어야 하다. 시험조건의 범위에는 경계값, 우발적 데이터 입력, 오류 조건, 분기(branches), 데이터 흐름, 입력 조합 등을 포함한다.

---

10) 회귀분석(Regression analysis)은 변수들 사이의 관계를 조사하여 모형화 시키는 통계적 기법으로서 경제, 경영, 교육, 정치 등의 사회과학 그리고 물리, 화학, 생물, 공학, 농학, 의학 등 자연과학의 모든 분야에서 널리 응용되고 있다. 즉, 변수들 간의 함수적인 관련성을 규명하기 위하여 수학적 모형(통계모형)을 가정하고, 관측된 자료로부터 이 모형을 추정하는 통계분석방법으로 주로 예측에 사용된다.

통합시험은 시뮬레이터를 사용해서 수행하기도 하며, 이 경우 대개 실제의 사용자 전산환경을 벗어난 오프라인으로 실행된다. 또한 실제 운영조건으로 최종사용자의 전산환경에서 수행[11]되는 시험은 시스템이 광범위한 조건과 사상을 겪기에 충분한 시간동안의 연속 조작들을 포함시켜 통상적인 활동 중에는 명확하게 드러나지 않는 잠재적 결함을 검출할 수 있어야 한다.

통합시험의 엄격성과 통합시험과 연관된 문서화의 상세 수준은 기기와 연관된 위험, 잠재적으로 위험한 기능에 대한 기기의 의존성 및 위험도가 높은 기기 기능에 있어 특정 소프트웨어의 역할에 적합하여야 한다. 예를 들어 모든 소프트웨어는 시험하여야 하지만 안전성에 영향을 미치는 항목은 보다 직접적이고 철저하며 상세한 시험의 대상이 된다.

통합시험 계획에는 통합시험 일환으로 수행되는 화이트박스 시험 유형을 포함하도록 한다. 시스템이나 구성요소의 내부 메커니즘(구조)을 평가하는 화이트박스 시험은 glass box, 구조적(structural) clear box 또는 open box 시험이라고도 한다. 이 시험은 소프트웨어 항목의 내부 작용에 대한 명백한 지식을 시험데이터 선택에 사용하는 경우의 시험으로 소스코드, 구체적인 설계 규격 및 다른 개발 문서로부터 획득된 지식을 기초로 하는 시험사례들을 명확하게 한다. 화이트박스 시험은 출력의 관찰을 위하여 소프트웨어의 특정 지식을 사용하며, 소프트웨어 항목(item)이 어떤 기능을 발휘하는지 시험자가 아는 경우에만 정밀하게 시험을 수행할 수 있다. 이 경우 시험자는 소프트웨어 항목이 의도된 목표에서 벗어나는지 여부를 확인할 수 있다. 화이트박스 시험은 소프트웨어 항목 구현의 시험에 집중하기 때문에 전체 시방서의 구현을 보장하지 못한다.

이와 달리 블랙박스 시험은 작동상태(behavioural), 기능, 불투명 박스(opaque-box) 및 closed- box 시험이라고 하며, 정의에 근거(definition-based)하거나 규격에 근거(specific-based)하는 시험이다. 이 시험은 각 유닛에서부터 시스템 시험까지 소프트웨어 시험의 전 수준에 적용될 수 있다. 블랙박스 시험은 기능적 시방서에 집중하므로, 이행의 전 부분이 시험되었다고 보증하지 못한다. 이처럼 화이트박스 시험은 이행과 비교한 시험을 수행하여 임무의 결함을 확인함으로써 이행의 그 부분이 결함임을 나타내고, 블랙박스 시험은 이미 알려진 조건에서 규정 입력으로 프로그램을 실행하고 결과를 문서화하여 설정된 기대값과 비교하는 과정을 수행하여 생략의 결함을 확인함으로써 시방서의 그 부분이 충족되지 않았음을 나타낸다.

11) Real time field test

소프트웨어에 대한 완전한 시험을 위하여 Black box와 White box 시험이 모두 필요할 수 있다.

제조업체는 통합 및 시험결과를 문서화하여야 한다. 문서화된 통합시험 결과에는 시험결과, 발견된 변종(anomalies), 시험한 소프트웨어의 버전[12], 관련 하드웨어와 소프트웨어 시험 형상, 관련 시험 도구 및 시험자 신원을 포함하여야 한다.

시험결과 기록에는 단순한 정성적 표현(예: 합격/불합격)보다는 정량적 표현을 사용하도록 한다. 이러한 정량적 표현은 시험결과의 후속 검토, 독립적 평가 및 시험을 반복할 수 있도록 해준다. 시험을 반복할 수 있는 충분한 기록은 예를 들어 다음을 유지함으로써 가능할 수 있다.

- 요구되는 조치와 예상되는 결과를 나타내는 시험 사례 시방서
- 장치 기록
- 시험에 사용하는 시험 환경(소프트웨어 도구 포함)의 기록

시스템 시험

제조업체는 모든 소프트웨어 시스템이 요구사항을 충족하는지 확인하기 위한 소프트웨어 시스템 시험 수행을 위한 계획(접근방식, 책임 및 순서 등) 및 일련의 시험(입력, 예상 결과, 시험 기준)절차를 수립하고 수행하여야 한다. 계획된 시스템 시험이 제조업체가 직접적으로 관리하지 않는 현장(site)에서 수행될 때 시험계획은 사용범위가 설정되고 예상된 시험결과의 정의 및 모든 시험 출력의 기록을 확실히 하기 위하여 충분한 관리를 필요로 한다. 또한 이전의 통합시험 및 소프트웨어 시스템 시험은 단일 계획 및 활동으로 결합할 수도 있다.

소프트웨어 시스템 시험은 다양한 지점에서 발생하고 다양한 조직에서 수행하기 때문에 시험에 대한 책임이 분산될 수 있다. 그러나 업무 분산, 계약에 따른 관계, 구성요인 발생원 또는 개발 환경에도 불구하고 제조업체는 소프트웨어가 의도한 사용에 적합한 기능을 발휘하는지 확인하는 궁극적인 책임을 져야 한다.

시스템 시험은 규격화된 모든 기능과 소프트웨어가 신뢰할 수 있음을 입증하는 것이다. 이 활동을 수행하려면 제조업체는 소프트웨어에 대한 기능, 성능 및 사용목적 등과

12) 형상 항목(item)의 식별된 인스턴스(instance)로 라벨은 디렉토리 구조에서 산출물 경로 및 특정시점에서 어떤 의미를 갖는 버전임을 파악하는데 유용한 형상식별 수단이다. 각종 소스의 버전을 관리할 수 있도록 도와주는 도구인 CVS(공동 버전 시스템, Concurrent Version System)이나 Visual Source Safe와 같은 관리 도구의 이용을 권장한다.

관련된 의료기기 소프트웨어의 다음과 같은 요소들이 성공적으로 이행되었는지 검증한다.

- 성능; 시스템 실행시간, 응답시간, 처리능력 등
- 스트레스 조건에서의 반응; 최대 부하(load), 지속적인 사용에서의 반응(behavior)
- 내·외부 안전성(Security); 시스템 내·외부로부터의 침입에 대한 보안
- 고장 복구 대책(disaster recovery)을 포함한 회복(Recovering)의 유효성
- 유용성(usability)
- 다른 소프트웨어와의 호환성
- 정의된 각 하드웨어 형상(configuration)에서의 반응
- 문서/기록들의 정확성

소프트웨어 시스템시험은 통합 소프트웨어를 시험하며 의도된 운영 환경에서 모의실험(simulation), 실제 대상 하드웨어 또는 전체 의료기기에서 수행하여 시스템이 광범위한 조건과 사상을 겪기에 충분한 시간동안 연속 조작들을 포함시켜 통상적인 활동 중에는 명확하게 드러나지 않는 잠재적 결함을 검출할 수 있어야 한다.

시스템 시험조건은 보다 효율적으로 특정 시험들을 수행하고 스트레스 조건이나 결함을 유발하며 적격성(qualification) 시험의 코드 적용범위 확대를 위하여, White box 시험이 바람직하더라도 Black box 시험에 집중하는 것을 권한다. 유형 및 단계별 시험 조직에 융통성을 부여할 수 있지만 요구사항, 위해요인 경감, 활용성 및 결함, 설치, 스트레스와 같은 시험유형에 대한 적용범위를 입증하고 문서화하여야 한다.

소프트웨어 시스템 시험 중 사소한 변경이라도, 변경이 발생할 경우

- 문제해결에 있어서 변경의 유효성을 확인하기 위하여 시험을 반복하거나, 수정한 시험을 수행하는 등의 추가 시험을 수행한다.
- 의도치 않은 부작용이 발생하지 않음을 입증하기에 충분하도록 소프트웨어 시스템 시험을 완전히 반복하지 않는 이론적 근거를 포함하여 회귀 시험을 수행한다.
- 관련 위험관리 활동을 수행하여야 한다.

소프트웨어 시스템 시험과정에서 발견된 오류는 소프트웨어의 릴리즈(release) 전에 등재(log), 분류(classify), 검토 및 해결하여야 한다. 시험 중 확인된 변종(anomalies)을 해결하기 위한 조치를 취하지 않을 경우, 위해요인 분석과 관련하여 그러한 변종들을 검토하고, 의료기기의 안전과 효율성에 영향을 미치지 않음을 입증하여야 한다. 또한 변종의 근본 원인, 증상 및 해결/조치를 하지 않은 이론적 근거를 문서화하여야 한다.

시험절차, 시험자료 및 시험결과는 객관적인 검토 및 판정을 하기에 적절하여야 하고 추후 발생할 수 있는 모든 회귀시험에서 사용하기에 적절하여야 한다. 제조업체는 시험결과의 후속 검토, 독립적 평가 및 시험을 반복할 수 있도록 시험 결과를 문서화하여야 한다. 문서화된 시험 결과에는 모든 시스템 구성요소가 시험과정에서 실행되었음과 시험한 소프트웨어 버전, 하드웨어, 시험 도구/장치, 시험에 사용되는 환경, 요구되는 조치와 예상 결과를 나타내는 시험 사례, 비정상 상태/오류 목록, 개정 수준이나 개정일, 시험일 및 시험자 신원을 포함하여야 한다.

소프트웨어 시스템 시험의 결과를 평가해 예상한 결과가 도출되는지 확인(ensure)하여야 한다.

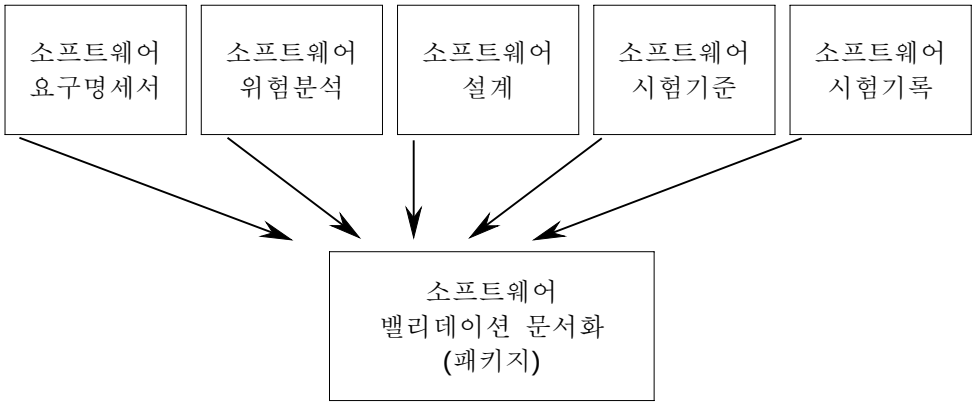
사용자 현장 시험(User Site Testing)

이에 관한 전문용어는 다소 혼란스럽다. 베타(beta) 테스트, 현장(site) 밸리데이션, 사용자 수용(user acceptance) 시험, 설치 검증 또는 설치 시험과 같은 용어가 사용되어 왔다.

상황에 따라 사용자의 현장에서 해당 소프트웨어의 시험이 이루어질 수도 있는데, 이 경우 시험은 설치되는 시스템 형상(configuration)의 일부가 될 수 있는 실제 하드웨어 및 소프트웨어를 갖춘 환경에서 수행되어야 한다. 시험계획은 안전을 확보하기 위한 관리기능을 명시하고, 의도한 적용범위와 충족 여부를 확인하여야 한다. 소프트웨어 제조업체가 직접적으로 관리하지 않는 현장에서 시험이 진행될 경우의 시험계획은 사용범위가 설정되고 예상 시험결과의 정의 및 모든 시험출력의 기록을 확실히 하기 위하여 충분한 관리를 필요로 한다.

밸리데이션 결과의 문서화

소프트웨어의 모든 요구사항이 충족되었음을 보장하는 밸리데이션 활동은 소프트웨어 개발 마지막 단계뿐만 아니라 그 과정에서 실시될 수 있다. 일반적으로 소프트웨어는 하드웨어 시스템의 일부분이기 때문에 개발 활동의 각 단계에서 실시되는 시험, 검사, 분석 및 다른 검증 활동에 크게 영향을 받는다. 소프트웨어의 모든 요구사항이 적절하고 완벽하게 되었음을 입증하는 밸리데이션 활동이 완료되면 그 결과를 문서화하여야 한다.



[그림 3] 소프트웨어 밸리데이션 문서화

문서화된 보고서는 시험결과 등, 밸리데이션 결과를 자세히 기술하거나 요약하여 기술하고 문서화된 다른 상세한 증빙자료와 추적성이 가능하도록 연결될 수 있도록 한다. 보고서에는 단순하게 '합격/불합격'으로 결과를 표시하기보다 가능한 정량적으로 표현하고, 밸리데이션 과정에서 발견된 소프트웨어 결함 및 이를 해결하기 위하여 취해진 조치에 대해서도 언급되어야 한다.

문서화된 보고서에 포함되는 내용은 다음과 같다.

- ・밸리데이션 일자 및 활동 수행자
- ・대상 소프트웨어의 버전
- ・밸리데이션에 사용된 시험 방법 및 유형
- ・시험에 사용된 도구, 관련 하드웨어
- ・밸리데이션/시험 결과
  - 결함(Fault), 경고(Alarm), 확인된 비정상 상태(변종, 버그 등)
  - 취해진 조치
- ・위험관리
  - 소프트웨어 안전성 등급, 위해요인 분석
- ・해당되는 경우 OTS 소프트웨어의 식별

문서화된 보고서는 지정된 자가 검토하고 승인한 후 유지・관리되어야 한다.

1.6 소프트웨어 릴리즈(Release)

소프트웨어를 릴리즈하기 전에 모든 밸리데이션 활동을 만족스럽게 완료하였으며, 지정된 권한자의 승인여부를 확인하여야 한다. 제조업체는 개발한 소프트웨어가 현재 릴리즈되는 소프트웨어인지 나타낼 수 있도록 릴리즈되는 소프트웨어의 버전과 설명서의

버전을 문서화하도록 한다. 또한 제조업체는 릴리즈한 소프트웨어 제품이 개악이나 무단 변경 없이 사용지점까지 인도될 수 있는 절차를 확립하여야 한다.

소프트웨어, 소스 코드 및 관련 문서의 정본(master copies)을 저장하고 유지하여야 한다.

릴리즈한 소프트웨어의 유지보수는 의료기기 소프트웨어의 생산 후 경험에 적용된다. 제조업체는 품질시스템에서 수립한 절차에 따라 소프트웨어에 수록된 매체의 손상과 오용이 없도록 생산과 취급(복제, 매체 라벨링, 포장, 보호, 보관, 인도)을 처리하여야 한다.

2. 소프트웨어 유지보수(변경관리) 활동

2.1 변경 및 문제해결

소프트웨어의 유지보수는 하드웨어에서와 같은 의미는 아니다. 하드웨어 및 소프트웨어의 많은 차이점 때문에 유지보수 활동방식도 다르다. 소프트웨어 유지보수 활동은 문제 보고서에 관한 상위 차원의 결정(문제의 존재 여부, 문제가 의료기기 안전성에 영향을 미치는지 여부, 필요한 변경[13] 및 변경의 이행 시점)을 취급하며, 소프트웨어 문제해결을 통하여 모든 암시를 파악하고, 변경 필요가 있는 모든 형상 항목(item)과 필요한 모든 밸리데이션 활동을 식별하는 것이다.

소프트웨어의 변경은 다음과 같은 이유에서 발생할 수 있다.

- 오류 발견 및 결점을 수정하기 위한 디버깅(debugging)
- 새로운 요구사항 발생 또는 요구사항의 변경
- 성능 또는 다른 소프트웨어 시스템 속성(attribute)을 향상시키기 위한 개선

소프트웨어 유지보수 활동의 초점은 소프트웨어의 릴리즈 후에 발생하는 피드백에 적절히 대응하는 것이다. 이를 위하여 다음과 같은 사항을 종합적으로 관리 운영하여야 한다.

- 발견된 문제점을 분석하고 문제의 모든 측면을 식별
- 안전과 관련된 문제점들을 해당 감독기관(예: 식품의약품안전청 등) 및 해당 사용자에게 보고
- 위험관리를 포함하여 소프트웨어 항목의 일관성을 유지하면서 변경을 이행
- 문제 해결 및 후속 문제 방지를 보장할 수 있도록 소프트웨어 변경 후 유효성을

13) 소프트웨어의 수정(modification), 향상(enhancement) 또는 추가(addition)를 통칭한다.

재확인.

- 영향을 받을 수 있는 다른 소프트웨어 제품에 대한 적절한 조치

소프트웨어 문제해결[14]은 소프트웨어 유지활동의 중요한 부분이다. 릴리즈된 소프트웨어에 문제가 발생하거나 개선의 필요성 때문에 소프트웨어의 코드 및 관련 문서가 수정되는 경우 유지보수 활동이 시작된다. 목적은 릴리즈된 소프트웨어 완전성(integrity)을 보존하면서 소프트웨어를 수정하는 것이다.

제조업체는 보고받은 문제점과 그 영향 등을 분석하여, 변경 사항을 이행하여야 한다. 제조업체는 릴리즈된 소프트웨어 제품의 문제 대응에 있어 문제 해결은 물론 관련 법규를 준수하여야 한다. 경우에 따라 릴리즈된 소프트웨어에서 발견된 문제점 및 기타 변경 필요사항은 의료기기의 안전성이나 허가받은 내용에 영향을 미칠 수 있다. 해당되는 경우 관련 감독기관 등으로의 통보도 고려하여야 한다.

제조업체는 소프트웨어 개발 활동보다 적은 규모의 자원으로 긴급한 문제를 신속하게 대응할 수 있다. 제조업체에서의 문제해결을 위한 세부적 절차는 다음과 같이 진행될 것이다.

- 소프트웨어와 관련된 문제점 수신 및 문서화
- 제조업체 내부 및 사용자에게 릴리즈된 소프트웨어에 대한 피드백(추적성)을 모니터링
  - 피드백이 문제로 간주되는지 여부를 결정
  - 문제점 발생시 모두 기록하고 그 영향을 평가
- 소프트웨어에 대한 위험관리를 실시
- 소프트웨어 릴리즈 후 발생하는 문제 분석과 해결을 위한 변경 활동 이행
  - 갱신, 버그 해결, 패치, 폐기 등
- 기존 소프트웨어의 변경 관리를 위하여 형상관리활동을 이행
- 변경된 소프트웨어에 대한 밸리데이션을 이행하고 승인
- 변경된 소프트웨어를 다시 릴리즈.

소프트웨어 유지보수과정에서 발견된 모든 문제는 반드시 문서화되어야 한다. 각 문제는 그 문제가 확실하게 시정되었음을 입증하기 위하여 기록하여야 하며, 소프트웨어 변경에 따라 개발활동에서 생성된 문서들의 변경여부를 주의 깊게 고려하여야 한다.

14) 소프트웨어 엔지니어링 문헌에서는 이를 “결함 추적(defect tracking)”이라 하는 경우가 많고, 국제규격 ISO/IEC 12207에서는 "문제해결(problem resolution)"으로, IEC 60601-1-4 규격에서는 "문제 보고(problem reporting)"라는 용어를 사용한다.

승인된 기존 문서(예: SRS, 사용자 설명서 등)의 변경이 필요하다고 판단되는 경우, 품질시스템에서의 문서관리 및 형상관리(Configuration Management) 절차에 따라 변경 승인되어야 할 것이다.

소프트웨어의 다양한 특징 중 하나는 변화 속도 및 용이성이다. 이 때문에 많은 사람들은 소프트웨어 문제는 쉽게 수정될 수 있다고 믿으며, 하드웨어에서와 같은 엄격한 변경관리는 필요 없다고 생각하여 소프트웨어의 변경관리는 철저하게 이루어지지 않는 경우가 많다.

일부 산업에서는 제품을 구성하는 수많은 부품(하드웨어)을 관리하기 위해 부품을 구분하고 각 부품간의 관계를 BOM(Bill of Material)에 정의하고, 완성된 제품의 설계 변경을 엄격하게 관리하기 위하여 ECC(Engineering Change Order)를 발행하고, 그 변경에 따른 영향 분석을 실시한다. 그러나 소프트웨어의 경우, 변경이 용이하기 때문에 빈번하게 변경이 발생되며 또한 이를 자연스럽게 생각한다. 중요한 것은 하나의 변경으로 인해 다른 부분의 안정성도 보장 받지 못할 수 있다. 따라서 변경으로 인한 영향은 파악해야 하고 변경을 받아들이기 위해 어느 정도의 노력이 필요한지 알아야 한다. 사소한 변경이든 중대한 변경이든 변경은 통제 대상에 놓여야 한다.

문제해결에서 매우 중요한 것은 소프트웨어가 변경됨에 따라 소프트웨어/하드웨어의 다른 부분/영역에 부작용이 발생하지 않음을 증명하는 것이다. 소프트웨어의 복잡성으로 인하여 소프트웨어에서 중요하지 않은 것처럼 보이는 자그마한 변경이 시스템에 광범위하게 영향을 미치거나, 다른 부분에 예상하지 못한 심각한 문제를 발생시킬 수 있다. 변경된 소프트웨어를 신규 개발한 소프트웨어로 취급하지 않는 한, 전체 의료기기에 대한 변경의 영향을 분석하여야 하는 것이다.

소프트웨어의 각 변경이 의료기기 전체 또는 소프트웨어 시스템에 미치는 변경의 정도와 영향을 평가하는데 필요한 검증 및/또는 밸리데이션 활동의 범위는 변경 형태, 영향을 받는 대상에 따라 결정될 것이다. 최초 개발활동에서 철저하고 완벽하게 이행하고 그 결과물을 문서화하였다면, 변경의 영향 평가에 필요한 밸리데이션은 작은 활동으로도 가능할 것이다.

2.2 문서화

소프트웨어의 개발 및 유지보수 과정에서 의료기기 소프트웨어가 어떻게 설계되었는지, 설계의도에 따라 어떻게 시험되었는지, 적절한 위해요인을 식별하고 위험관리가 효과적으로 이행되었는지 등을 입증하는 다양한 문서(전자매체 포함), 기록들이 산출될 것이다.

제조업체는 소프트웨어 개발 및 유지보수 활동에서 생성된 문서, 기록을 식별하여야 한다. 생성되는 문서, 기록의 범위 및 종류는 개발·유지 관리되는 소프트웨어 규모, 복잡성 및 소프트웨어 안전성 분류에 따라 다르게 된다. 소프트웨어 개발환경에서 활동 결과로 산출된 문서 및 기록들에 대한 체계적인 관리는 개발·유지보수 활동을 원활하게 해줄뿐만 아니라 품질관리에도 큰 영향을 미치게 된다. 또한 이 문서 및 기록의 일부는 의료기기 허가/승인 등을 위하여 관계기관에 제출되기도 한다.

다양한 문서, 기록들은 검색이 용이하고 효율적인 코드, 날짜 등을 부여하여 개정·갱신에 따른 혼란이 일어나지 않도록 품질시스템의 문서관리, 형상관리 및 기록관리 절차에 따라 관리하여야 한다.

제조업체에서 관리가 필요할 주요 문서, 기록들은 다음과 같다.

- · 의료기기/소프트웨어 개발계획서
- · 소프트웨어 요구사항 명세서(SRS, Software Requirements Specification)
- · 소프트웨어 아키텍처 설계도(Software Architecture Design Chart)
- · 소프트웨어 설계 기술서(SDD, Software Design Description)
- · 소프트웨어 설계 명세서(SDS, Software Design Specification)
- · 소프트웨어 검증 및 밸리데이션
  - 검증 및 밸리데이션 활동 기록, 활동 결과에 대한 조치, 미해결된 변종(버그 또는 결함) 등을 포함.
  - 소프트웨어 형상관리(SCM, Software Configuration Management)
- · 기타 문서화
  - 사용자를 위한 문서화: 소프트웨어 설명서, 매뉴얼, 지침서 등
  - 제조업체에서의 문서화: 소프트웨어 유지보수 매뉴얼 등

본 가이드라인에서 이러한 문서화를 언급하는 것은 이 범위를 한정하기 위함이 아니다. 제조업체는 그들이 개발·유지하려는 소프트웨어의 사용목적, 규모/복잡성 및 안전성 분류에 따라 산출되는 문서화의 범위가 다를 수 있음을 인지하여야 한다.

소프트웨어 요구사항 명세서(SRS)

SRS는 소프트웨어가 무엇을 하는지에 대하여 기술한 것이다. 전형적으로 SRS에는 소프트웨어의 기능, 수행능력, 설계 제약, 속성 및 인터페이스 등에 대하여 각각 명확히 기술하고, 이 요구사항들을 시험·검사, 분석 등으로 객관적인 검증 및 밸리데이션이 가능하도록 설정한다. 여기에는 다음과 같은 내용 등이 서술될 것이다.

· 하드웨어 요구사항
- 마이크로프로세서(microprocessors), 메모리, 센서 등
· 프로그램밍 언어
- 메모리 렉(memory leaks) 관리에 대한 정보와 제한 또는 프로그램 크기(size)등을 포함
· 인터페이스
- 시스템 구성요소들 간의 커뮤니케이션 및 프린터, 모니터, 키보드, 마우스와 같은 유저(user)와의 커뮤니케이션 모두를 포함
· 기능
- 소프트웨어로 인한 의료기기의 제한 조건(limitations), 오류와 인터럽트(interrupt) 처리, 소프트웨어 시험 및 확인, 결함 감지, 허용차(tolerance) 및 복구(recovery) 특성, 타이밍(timing), 안전 요구사항
- 치료, 진단, 감시, 경고, 분석을 위한 알고리즘이나 제어 특성 및 필요한 경우 전체 텍스트 참고문헌(references)이나 보조(supporting) 임상자료의 해석이 포함.
- 해당되는 경우 OTS 소프트웨어의 식별

소프트웨어 아키텍처 설계도(Software Architecture Design Chart)

네트워크 연결(networking)과 같이 하드웨어와 데이터 흐름과의 관계를 포함하여 소프트웨어에서의 주요 기능별 유닛간 관계에 대한 흐름도(flowchart) 또는 이와 유사한 설명을 기술한 것이다. 일반적으로 여기에 모든 기능 및 모듈을 포함시킬 필요는 없다. 다만, 소프트웨어의 기능 및 사용 목적에 관련된 소프트웨어 아키텍처를 검토하기에 충분한 정보가 포함되어야 한다. 소프트웨어에 대한 안전성 분류가 '중'이거나 '상'으로 평가된 경우는 도해(diagrams)와 같이 소프트웨어 기능별 유닛간의 관계를 명확히 묘사한 상세한 정보가 유용할 수 있다.

소프트웨어 설계 기술서(SDD, Software Design Description)

SDD는 SRS 요구사항을 만족시키기 위하여 어떻게 구성되어야 하는지 설명하는 문서이다. SDD는 데이터베이스와 내부 인터페이스를 포함하여 소프트웨어 설계의 구성요소와 하위 구성요소들에 대해서도 기술한다.

소프트웨어 설계 명세서(Software Design Specification)

SDS는 소프트웨어에 대한 요구사항 실현(implementation)을 설명한 문서이다. SRS와 SDS 관계에서, SRS는 '소프트웨어 기기가 무엇을 하는가?'를 설정한 것인 반면, SDS는 'SRS에 설정한 요구사항을 어떻게 실현하는가?'로 정의된다. SDS에 기술된 내용은

소프트웨어를 만드는 설계팀·엔지니어가 설계 수행과정에서 즉흥적이거나 임시적인 설계 결정이 없도록 분명하고 명확하여야 한다.

소프트웨어 형상관리(SCM, Software Configuration Management)

형상관리는 소프트웨어 개발·유지보수 활동에서 매우 중요한 요소이다. 형상관리는 2.3항에서 후술한다.

사용자를 위한 문서화

사용자 문서화는 소프트웨어 설명서, 매뉴얼, 지침서 등과 같이 소프트웨어의 성공적 실행을 위해 요구되는 자료 및 제어입력, 입력 순서, 선택사항, 프로그램 한계 및 다른 조치나 항목을 명시한 것들이다. 내장형(embedded) 소프트웨어이며 사용자와 직접적인 상호작용이 없는 경우는 사용자 문서화가 필요하지 않다.

2.3 형상관리

소프트웨어 형상관리(SCM, Software Configuration Management)는 최근 몇 년 사이에 소프트웨어의 품질보증활동에서 아주 중요하게 다뤄지고 있는 분야이다. 특히 소프트웨어 규모가 커짐에 따라, 혹은 소프트웨어에서 품질 요소가 중요한 위치를 차지함에 따라 소프트웨어 형상관리의 중요성이 더욱 부각되고 있다.

소프트웨어의 가장 큰 특징은 변경이고, 이 변경으로 인하여 개발과 유지보수 활동을 아주 어렵게 만들어 버린다. 변경이라는 소프트웨어의 큰 특징을 무시할 수 없기에 이를 효과적으로 해결할 수 있는 수많은 연구결과의 하나가 "소프트웨어 형상관리"이다.

소프트웨어에서 변경은 언제나 일어날 수 있기 때문에, 형상관리 활동은 ①변경을 알아내기 위하여, ②변경을 관리하기 위하여, ③변경이 적절히 수행되고 있음을 확인하기 위하여, ④변경과 관련된 사람들에게 이것을 통보하기 위한 것"이다.

소프트웨어 형상관리는 "형상 항목(Configuration Item)의 완벽성과 정확성 확보를 위하여 소프트웨어 수명주기 전반에 걸쳐 형상을 적용하는 프로세스[15]"로 정의하고 있다.

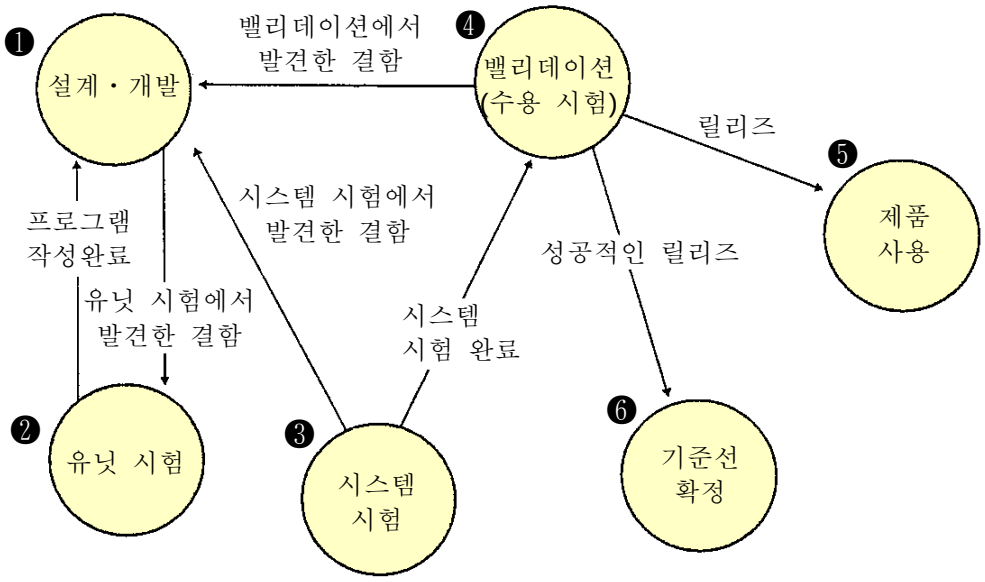
형상관리에서 "형상" 용어의 의미를 광의적으로 해석하면 "소프트웨어 그 자체"이다. 소프트웨어는 PC에서 실행되는 작은 규모도 있고 네트워크와 연결되어 복수의 서버에서

15) "형상항목을 식별하여 기능적 물리적 특성을 문서화하고, 그 특성에 대한 변경을 제어하고, 변경 처리 상태를 기록 및 보고하고, 명시된 요구사항에 부합하는지 확인하는 일련의 사항에 대하여 기술적 행정적인 지침과 사후 관리를 적용하는 원칙"으로 정의하기도 한다.

운영되는 크고 복잡한 규모도 있다. 따라서 엄밀하게 말하여 소프트웨어를 구성하는 모든 것으로서 소프트웨어를 개발하는 조직, 개발자, 소프트웨어가 포함되는 하드웨어, 소프트웨어 자체, 소프트웨어 개발활동 및 형상관리에 필요한 형상관리 도구와 개발 도구 모두가 해당된다고 할 수 있다.

형상관리 사용은 소프트웨어의 규모, 복잡성 및 위험에 상응하는 정도(안전성 등급)에 의한 것으로, 형상관리의 가장 큰 목적은 변경에 의해 점차적으로 변해가는 소프트웨어의 형상을 관리하는 것이다. 형상항목 변경은 특정 상태를 거쳐 이뤄지는데, 그 흐름은 그림 4와 같다.

형상항목의 수명주기는 크게 6단계로 나뉠 수 있는데 개발 초기에 설계・개발(①) 상태에 위치하게 된다. 개발이 진행됨에 따라 유닛시험(②, Unit test)을 하고 이를 통과하면 시스템 시험(③) 준비상태로 변하게 된다. 시스템 시험(③, System test)도 통과하면 밸리데이션(④)에서 소프트웨어의 수용(acceptance)여부를 결정하게 되고 이후 모든 결함, 문제점을 제거하면 기준선(⑥, baseline)을 확정하고 제품을 릴리즈(⑤)하게 된다. 이때 각각의 시험에서 문제점을 발견하면 그 전의 상태로 돌아가고 발견된 모든 문제점이 제거될 때까지 과정을 반복하게 된다. 그림 4에서와 같이 우선 식별된 형상 항목을 최초로 작성하고 각각의 시험을 거치면서 형상 항목은 변경된다. 유닛시험에서 발견된 문제점은 수정되어 버전 통제를 받게 될 것이고, 유닛시험을 통과한 유닛들은 의미 있는 집합(소프트웨어 항목)으로 묶여 시스템시험을 받게 되며 이때 결함이 발견되면 해당 유닛들과의 추적(trace)을 포함한 변경 절차가 진행된다. 이후 밸리데이션에서는 소프트웨어의 요구사항과 직접 관련되므로 변경 절차가 강화되고 요구사항과 변경과의 추적을 관리하여야 하며, 소프트웨어의 수용이 결정되면 기준선이 확정되고 이 기준선은 변경할 수 없는 매우 중요한 지표가 된다.



[그림 4] 형상항목의 변경 흐름

소프트웨어에서 식별과 추적성을 달성할 수 있는 수단이 형상관리이다. 형상관리의 목적중 하나는 현재 형상과 요구사항의 달성 상태에 대한 완전한 가시성을 문서화하고 제공하는 것이다 또 다른 목적은 소프트웨어의 모든 개발・유지보수 활동자들에게 언제라도 올바르고 정확한 정보를 제공하는 것이다. 이처럼 형상관리는 문서를 포함하여 시스템의 소프트웨어 항목을 식별하고 정의하며 기준을 설정하고, 변경을 관리하고, 소프트웨어 항목의 상태와 변경 요청을 기록 보고하기 위한, 개발・유지보수 활동 전체에서 행정적 및 기술적 절차를 적용시킬 수 있는 수단이다. 이 형상관리는 관련 문서와 하드웨어에 적용 가능하며, 소프트웨어 항목을 재창출하고, 항목에 이루어진 변경의 이력 제공에도 필요하다.

형상관리를 통하여 다음과 같은 활동이 가능할 것이다.

- 각 소프트웨어 항목의 고유한 버전을 식별
- 완성된 소프트웨어의 특정 버전과 함께 구성하는 각 소프트웨어 항목의 버전을 식별
- 개발 중이거나 인도 또는 설치된 소프트웨어 제품의 형상상태를 식별
- 독립적 또는 종속적으로 개발・유지보수 활동을 수행하는 2명 이상의 수행자에 의한 해당 소프트웨어 항목의 동시 변경관리
- 요구되는 경우, 한 곳 이상의 위치에서의 다수 소프트웨어의 변경을 위한 조정
- 개발착수에서 릴리즈에 이르기까지 변경의 필요성, 요구 또는 문제해결을 위한 모든 조치 및 변경의 식별

최소한으로, 제조업체는 SOUP와 같은 다른 소프트웨어 제품을 포함[16]하여 형상항목을

식별하기 위한 방법을 수립하고, 이에 따라 관리할 형상항목과 그 버전을 문서화하며, 형상항목에 대한 이력(형상 상태)을 검색할 수 있는 기록을 유지하여야 할 것이다.

16) 예를 들어 버전, 릴리즈 날짜, 패치 번호 또는 업그레이드 명칭일 수도 있다.

Ⅲ. 공정에서의 소프트웨어 밸리데이션

의료기기 산업계에서 컴퓨터 및 자동화 설비는 광범위하게 사용되고 있다. 소규모의 제조업체에서도 소프트웨어를 이용하여 제조공정을 자동화하고 아웃풋을 확대함으로서 생산성을 개선하며 인력 비용을 감소시키고 있다. 비록 이러한 대부분의 소프트웨어 시스템들은 외부 외주업체에 주문 개발되지만 궁극적으로 의료기기 제품의 안전성과 적합성을 보증할 일차적 책임은 의료기기 제조업체에게 있다.

이러한 공정 소프트웨어의 예는 다음과 같다.

a) 생산공정 또는 품질시스템 활동에 관한 정보를 작업자가 개입하지 않고 수집, 분석하여 활동을 관리하고 결과를 검증하는 경우 그리고 그런 결과에 대해서 조치를 취하는 programmable logic controller, Digital function controller 등과 같은 임베디드 시스템(embedded system) 설비
b) PCB 및 전자부품 같은 반제품(Sub-Assembly)의 시험과 완제품의 최종 합격판정 시험에 사용되는 시험용 소프트웨어
c) 시험 및 검사 장치에 의한 물질 또는 제품의 평가, 교정, DMR의 관리, 불만처리, 설계 및 경향분석 같은 품질시스템 절차를 실행하고 지원하는 데 사용되는 소프트웨어
d) 웨이브 솔더 머신, 로봇(Robotics), 컴퓨터수치제어(CNC) 기계, 환경관리 설비, 멸균장치, 연속조립공정 등의 각종 생산공정과 장치를 관리하는 데 사용되는 생산관리 소프트웨어

우리의 의료기기 제조 및 품질관리기준 별표 1의 7.5.2.1에서 “제조업자는 규정된 요구사항을 충족하기 위하여 제품 성능에 영향을 미치는 컴퓨터 소프트웨어 적용(소프트웨어 및/또는 적용의 변경을 포함)의 밸리데이션을 위한 문서화된 절차를 수립하고, 이러한 소프트웨어 적용에 있어 최초 사용 전에 유효성을 확인”하도록 요구하고 있다. 이 요구사항 목적은 설계, 제조, 유통, 추적관리 등 품질시스템의 모든 측면에서 사용되는 소프트웨어는 설치가 정확하며 의도한 용도·목적에 따라 유효한 결과가 산출됨을 입증하는 것이다. 따라서 제조업체에서는 설정된 밸리데이션 절차(Protocol)에 따라 소프트웨어의 유효성을 확인하여 의도한 바와 같이 일관되게 기능을 발휘함을 보증할 수 있어야 한다.

의료기기 제조업체에서 사용하는 자동화 설비 소프트웨어 및 제품 품질의 통계, 경향분석에 사용되는 스프레드시트, 그래픽처리, 의료기기 이력 또는 고객 불만관리에

사용되는 다수의 소프트웨어는 자체적으로 또는 계약에 따라 개발되거나 제3자로부터 구매한 기성품(OTS; off-the-shelf) 소프트웨어를 사용한다. 소프트웨어를 제조업체가 자체적으로, 또는 주문계약에 의해 개발하거나 또는 OTS 제품에 상관없이 제조업체는 생산 및 품질시스템에서 시용하는 소프트웨어에 대한 밸리데이션을 어느 정도까지, 어떻게 수행할 것인지를 결정하여야 한다.

제조업체에서 곤란스러워 하는 것은 소프트웨어 밸리데이션 수준(정도)의 설정일 것이다. 고품질 소프트웨어에 대한한 다양한 문헌과 국제표준의 유용성에도 불구하고, 많은 소규모 제조업체에서는 “소프트웨어 밸리데이션이 필요한 컴퓨터 자동화 설비가 대체 무엇인가?”와 “소프트웨어 밸리데이션에 적절한 기록은 무엇인가?”라는 질문에 초점을 둘 것이다.

의료기기의 안전, 성능 및 신뢰성과 관련된 자동화 설비 및 품질시스템에서의 소프트웨어 밸리데이션이 요구되는 수준은 소프트웨어 이용과 관련한 위험도와 일치한다고 할 수 있다. 부연하자면, 자동화 운영에서 제품 성능에 중요하거나 또는 안전성에 영향을 미치는 요인은 노출되는 위험에 적절하며, 안전하고 효과적인 의료기기 생산을 위한 밸리데이션의 일부분으로서 확인하여야 할 것이다. 또한 이러한 범위와 확인방법에는 해당 소프트웨어 고유의 완전한 기능성뿐만 아니라 위해요인(hazard) 분석, 제조업체에서 의도한 용도도 고려되어야 한다.

제조업체에서 수행하는 소프트웨어의 밸리데이션은 표 7과 같이 몇 가지로 분류하여 고려할 수 있다.

‘범주 I’에 해당하는 소프트웨어 종류는 제품 품질의 통계, 경향분석에 사용되는 스프레드시트, 그래픽처리, 의료기기 이력이나 고객 불만관리 등 품질관리를 위한 소프트웨어로서 대개 PC, PC네트워크, 또는 대형 컴퓨터에 상주한다. 또 컴퓨터 수치제어설비(CNC), 자동 선별기(시험・검사용 소프트웨어), PCB 시험도구 등과 같은 소프트웨어도 여기에 해당된다.

이 범주에 속하는 소프트웨어는 설비/장비에 내장되어 사용자가 메뉴에서 여러 가지 옵션을 선택하고 여러 파라미터를 입력하여 보고서를 인쇄하며 제한된 데이터를 이용하여 공정을 관리할 수 있게 하여준다. 이러한 소프트웨어는 의료기기 제조업체에서 아주 적은 노력과 시험으로도 가능하다. 경우에 따라 품질관리를 위한 기성품(OTS) 소프트웨어는 제조업체의 의도된 용도로 출력여부를 확인(기능상의 확인)하는 것으로 충분할 수도 있다.

유 형	설명	밸리데이션 정도
범주 I	- 공정 또는 의료기기가 제품의 안전, 일관성, 성능에 영향을 주지 않거나, 경미하다.	- 예방차원의 관리 - 간단한 시험・검사 또는 교정
범주 II	- 공정 또는 의료기기가 제품의 안전, 일관성, 성능에 미치는 영향을 무시할 수 없다. - 소프트웨어 자체에 대한 밸리데이션이 곤란하다.	- 시험・검사 - 전체적 또는 부분적인 공정 밸리데이션(설치, 운영, 성능 적격성평가)
범주 III	- 공정 또는 의료기기가 제품의 안전, 일관성, 성능에 주는 영향이 지대하거나, 영향을 파악할 수 없다.(소프트웨어가 주문설계되거나, 신뢰성에 대한 의심이 크다)	- 완전한 소프트웨어 밸리데이션

[표 7] 소프트웨어의 밸리데이션 유형

예 를 들어 CNC 프로그램은 통상 기계도면 형식으로 기술되며 출력물은 일반적으로 기구류(부품, 구조물)이다. CNC 프로그램은 다수의 부품들을 가공한 후 프로그램에 사용된 도면의 모든 치수가 규정된 치수 허용한계 이내인지를 측정함으로서 밸리데이션이 가능하다. CNC 프로그램을 네트워크에서 다운로드(download)를 받거나, 단독적인 전자매체(CD, Disk) 형태로 CNC에 설치하는 경우도 있다. 이런 프로그램은 최초 개발 시점 및 업그레이드의 변경 시점마다 밸리데이션이 필요하다. CNC 부품・공구는 마모되는 경향이 있으므로 정기적인 조정(Adjust)이 필요할 수 있다. 통상적으로 이런 조정은 밸리데이션 활동이 아닌 설비의 유지보수 활동의 일환으로 이행된다.

'범주 II'에 속하는 소프트웨어는 컴퓨터로 제어되는 공정이나 설비에 조작자(운영자)가 개입하지 않고 데이터를 수집, 분석하여 자동으로 조치를 수행하는 programmable logic controller, Digital function controller 소프트웨어 등이다. 예를 들어, 조작자가 미리 정해진 범위한계 내에서 시간, 온습도 등을 설정하고, 이를 컴퓨터로 조정・제어하는 환경관리용 소프트웨어는 제조업체에서 소프트웨어 자체만의 밸리데이션이 불가능하다. 이와 같이 하드웨어/설비와 같이 운용이 되는, "특별공정"으로 호칭되는 공정에서 사용되는 소프트웨어에 대해서는 소프트웨어 밸리데이션이 아닌 공정 밸리데이션(Process Validation)으로 평가될 필요가 있다. 공정 밸리데이션의 설치(Installation), 운용 적격성평가(Operating Qualification) 및 성능 적격성평가(Performance Qualification)를

통하여 밸리데이션이 보증되는 경우 내부 소프트웨어의 밸리데이션은 공정 밸리데이션 활동에 포함된다.

'범주 III'에 속하는 소프트웨어는 "2절 개발에서의 밸리데이션"에서 전술한 바와 같이 소프트웨어의 의도된 용도(사용목적), 의료기기 및 공정에 미치는 안전성 등을 평가하여야 한다. 제조업체에서 소프트웨어를 전문업체에 주문하여 개발하는 경우 제조업체는 제조품질관리기준 및 기타 요건에 적합한 소프트웨어를 요구하여야 한다. 이 경우 제조업체는 소프트웨어의 요건을 개발하고 검증과 합격판정 기준을 결정하는 것이 최상이다. 궁극적으로, 주문형 소프트웨어가 요구사항에 충족함을 보증하는 일은 제조업체의 책임이다.

제조업체는 기성품(OTS) 소프트웨어를 구매하여 수정·변형하여 사용할 수도 있다. 기성품 소프트웨어를 수정·변형하는 경우 요구사항은 특정 응용 프로그램과 관계된 소프트웨어의 의도된 용도에 기초하여 결정되므로 소프트웨어와 함께 제공되는 사용설명서에 근거를 둘 필요는 없다. 제조업체는 수정·변형된 소프트웨어가 밸리데이션을 통하여 그 수정사항 및 수정사항으로 인한 다른 부분/시스템에 미치는 영향을 구체적으로 검토하여야 한다.

밸리데이션에 필요한 정보가 내·외부로부터 가능하지 않은 SOUP[17]의 경우 소프트웨어의 코드는 알 수 없거나 액세스될 수 없다. 이런 경우 소프트웨어의 기능성만 다뤄지며 코드 수준의 밸리데이션은 안되므로, 통상적으로 제조업체는 소프트웨어가 "사용자 필요 및 사용 의도"를 확립하기 위한 충분한 "블랙박스(black box)" 시험을 실시하여 소프트웨어가 사용자 요구를 충족시키며 의도된 용도에 적합한지 유효성을 확인하여야 한다. 즉, 제조자는 소프트웨어 환경에서 시험을 시행하기 위해서 일련의 시험조건과 예상 입력을 고안하여 설치와 기능의 적절함에 대해서 소프트웨어를 평가하여야 하며, 이 경우 예상 입력은 공정오류의 가능성 등이다. 다양한 응용 프로그램에서는 블랙박스 검사 하나로는 충분하지 않을 수도 있다.

밸리데이션은 반드시 문서화된 계획(Protocol)에 따라 수행되어야 한다. 또한 소프트웨어의 밸리데이션 결과는 객관적인 입증이 가능하도록 반드시 품질기록으로 유지·관리하여야 한다. 밸리데이션 결과기록에는 대상 소프트웨어 및 버전(Version)의 식별, 밸리데이션 실시날짜, 합격/불합격 기준, 밸리데이션 방법 및 결과(시험 또는 측정값) 등이 포함되어야 한다.

17) Software Of Unknown Provenance(기원이 확인되지 않은 소프트웨어)의 약어로, 이미 개발되어 일반적으로 사용할 수 있거나, 의료기기에 채택할 목적으로 개발되지 않은 기성품(Off-the-shelf, OTS) 소프트웨어 또는 이전에 개발되어 개발 프로세스에 대한 적절한 기록이 없는 소프트웨어

소프트웨어는 시간의 경과에 따라 문제점을 해결하거나 기능의 업그레이드 필요성 때문에 변경될 수 있으므로 소프트웨어와 소프트웨어변경은 최초 사용 전에 공식적으로 검토되고 승인되어야 한다. 제조업체가 어떤 이유로, 예를 들어 용량을 늘리거나 문제점을 해결하기 위하여 밸리데이션된 소프트웨어를 변경하는 경우는 변경 사용 전에 변경된 소프트웨어의 유효성을 다시 확인하여 시스템 내에서 다른 소프트웨어 유닛에 대한 영향 및 제품 품질에 미치는 영향 등을 평가하고 판단하여야 한다. 변경의 특성은 문서화 해두며 밸리데이션 기록에는 변경사항과 그것이 미치는 영향을 구체적으로 언급되어야 한다.

변경되지 않은 소프트웨어일지라도 정기적인 유효성 재확인(Revalidation)이 필요하다. 이는 "생산 및 서비스 제공 프로세스의 유효성 재확인(Validation)[18]"과 운용의 궤를 같이 한다. 이 주기가 규정된 바는 없으나 통상적으로 1년을 적용한다.

컴퓨터 소프트웨어 및 관련 문서의 정본(master copies)은 품질시스템의 문서관리 절차에 따라 저장하고, 개정이력을 관리하여야 한다. 또한 변경 전 소프트웨어(old version)도 보존해 두는 것이 필요하다.

18) 의료기기 제조·수입 및 품질관리기준(식약청 고시 제2005-69호) 별표 1의 7.5.2.1 다. 5

Ⅳ. 참고 문헌

1. ISO 14971:2000 의료기기에 대한 위험관리의 적용
2. ISO/IEC TR 15846:1998 소프트웨어 수명주기 프로세스 - 형상관리
3. ISO 10007:1995 품질경영 - 형상관리에 대한 지침
4. ISO 90003:2004 소프트웨어 및 시스템 엔지니어링 - 컴퓨터 소프트웨어에 대한 ISO 9001:2000 적용을 위한 지침
5. IEC 62304/FDiS(2006) 의료기기 소프트웨어 수명주기 프로세스
6. IEC 60601-1-4:2000 의료용 전기기기-안전 일반 요구사항-부가규격: 프로그램 가능한 전기 의료시스템
7. IEC 60601-1-6:2004 의료용 전기기기 - 안전 일반 요구사항 - 보충 규격: 사용적합성
8. IEEE Std 828-1998 소프트웨어 형상 관리 계획 표준
9. IEEE Std 830-1993 소프트웨어 요구사양(SRS)에 대한 권고기준
10. IEEE STD 1012:1998 소프트웨어 검증 및 밸리데이션에 관한 IEEE 표준
11. IEEE STD 1012A:1998 소프트웨어 검증 및 밸리데이션에 대한 IEEE 표준 추가사항 : IEEE/EIA 12207.1-1997에 대한 내용
12. IEEE STD 1028:1998 소프트웨어 검토 및 심사(Audits)에 관한 IEEE 표준
13. FDA 지침 S/W 시판전 신고(2005) Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices
14. FDA 지침 SW Validation 원칙(2002) General Principles of Software Validation
15. 21 CFR PART 11 전자 기록; 전자 서명
16. 21 CFR Part 820 Quality System Regulatory
17. ANSI/AAMI SW68:2001 의료기기 소프트웨어 수명주기 프로세스
18. 93/42/EEC 유럽 의료기기 지침
19. MB-MED/2.2/Rec4 소프트웨어와 의료기기

[별첨 1]

국제규격화 동향

1. ISO/IEC 국제규격 분석

1990년도부터 시작된 소프트웨어 분야의 국제규격 활동은 국제규격화기구(ISO) 및 국제전기기술위원회(IEC)의 통합 기술 위원회인 JTC1/SC7 소프트웨어 엔지니어링에서 주도하고 있다. 현재 이 분야 WG 6에서의 국제규격화 작업은 소프트웨어 제품의 품질 특성 및 측정방법을 정의하는 ISO/IEC 9126 계열 및 14598 계열규격 제정을 완성하였으며, WG 10에서는 소프트웨어 개발업체의 능력과 성숙도를 평가하는 ISO/IEC TR 15504 계열규격(SPICE, Software Process Improvement and Capability dEtermination)를 제정하여 운영 중으로서, 이는 다음과 같이 소프트웨어 제품 평가, 프로세스 평가, 품질시스템 분야로 분리하여 고려할 수 있다.

- 소프트웨어 수명주기 관련 규격
  - ISO/IEC 12207:1995 (소프트웨어 수명주기 프로세스)
  - IEC 62304:2006[19] (의료기기 소프트웨어 수명주기[20] 프로세스)

- 소프트웨어 제품의 품질특성 및 매트릭스(Software Quality Characteristics and Metrics) 관련 ISO/IEC 9126 계열규격
  - ISO/IEC 9126-1 (소프트웨어 제품 품질-Part 1: 품질모델)
  - ISO/IEC 9126-2 (소프트웨어 제품 품질-Part 2: 외부 메트릭)
  - ISO/IEC 9126-3 (소프트웨어 제품 품질-Part 3: 내부 메트릭)
  - ISO/IEC 9126-4 (소프트웨어 제품 품질-Part 4: 메트릭 사용에서의 품질)

- 소프트웨어 제품의 평가(Software Product Evaluation) 관련 ISO/IEC 14598 계열규격
  - ISO/IEC 14598-1 (소프트웨어 제품 평가-Part 1: 개요)
  - ISO/IEC 14598-2 (소프트웨어 제품 평가-Part 2: 기획 및 관리)

---

19) IEC JTC(Joint Technical Committee) 1의 SC(Subcommittee) 62A Joint Working Group에서 2006년 5월 9일 발간되어 현재 FDIS 상태인 규격으로서, ISO/TC 210, ISO/IEC JTC 1/SC 7, CENELEC TC 62외에도 TC 62, SC 62B/C/D, TC 66, TC 76외 다수 TC 및 SC가 참여하여 합의한 규격

20) life cycle, 소프트웨어 제품의 개발, 운용 및 유지보수에 내포된 프로세스, 활동 및 업무를 포함하고, 시스템 수명을 시스템 요구사항 정의에서 시스템 사용의 종료까지 보는 구조 (framework)로, 소프트웨어 개발 수명주기는 요구사항의 정의에서부터 제조를 위한 릴리즈에 이르기까지 소프트웨어의 수명 전체를 포괄하는 개념적 구조.

- ISO/IEC 14598-3 (소프트웨어 제품 평가-Part 3: 개발자를 위한 프로세스)
- ISO/IEC 14598-4 (소프트웨어 제품 평가-Part 4: 취급자를 위한 프로세스)
- ISO/IEC 14598-5 (소프트웨어 제품 평가-Part 5: 평가자를 위한 프로세스)
- ISO/IEC 14598-6 (소프트웨어 제품 평가-Part 6: 평가모듈의 문서화)

• 소프트웨어 개발업체 능력과 성숙성 등의 프로세스 평가(Software Process Assessment) 관련 ISO/IEC TR15504 계열규격
- ISO/IEC TR15504-1 (소프트웨어 프로세스 평가-Part 1: 개요)
- ISO/IEC TR15504-2 (소프트웨어 프로세스 평가-Part 2: 프로세스 및 능력에 대한 모델)
- ISO/IEC TR15504-3 (소프트웨어 프로세스 평가-Part 3: 평가 수행)
- ISO/IEC TR15504-4 (소프트웨어 프로세스 평가-Part 4: 평가 지침)
- ISO/IEC TR15504-5 (소프트웨어 프로세스 평가-Part 5: 평가 모델 및 지표 지침)
- ISO/IEC TR15504-6 (소프트웨어 프로세스 평가-Part 6: 심사자의 능력 평가 지침)
- ISO/IEC TR15504-7 (소프트웨어 프로세스 평가-Part 7: 프로세스 개선 지침)
- ISO/IEC TR15504-8 (소프트웨어 프로세스 평가-Part 8: 공급자 프로세스 능력 측정 지침)

• 기타 관련 규격
- ISO/IEC 90003:2004 (컴퓨터 소프트웨어에 ISO 9001:2000 품질경영시스템(Quality Management System)적용을 위한 지침)
- ISO/IEC TR 15846:1998 (소프트웨어 수명주기 프로세스에서의 형상관리 (Configuration Management))

1995년 8월에 발간된 ISO/IEC 12207은 항공, 자동차, 의학 등 모든 사업 분야에서 사용할 수 있는 소프트웨어의 수명주기 프로세스, 활동 및 업무에 대하여 종합적으로 규정하고 있으며, ISO/IEC 12207에 확립된 소프트웨어 수명주기 프로세스를 위한 체제를 사용하여 그 의도를 전달하고 실현을 증명하기에 필요한 일련의 프로세스, 활동 및 업무를 설명하고 있다. 이 규격은 소프트웨어 수명주기 전체를 단계별로 구분하고 해당 단계별로 이행하여야 할 활동을 언급하여, 소프트웨어 수명주기 전체에서의 품질관리 활동을 수행하는 것이 가장 최선이며 효율적인 방법임을 규정하고 있다.

대규모화·복잡화되고 있는 소프트웨어의 개발 및 유지보수 활동에서 소프트웨어 품질은 수명주기의 각 단계에서 모두 영향을 미치고 있으며 특히 수명주기 초기단계의 오류일수록 최종산출물에 치명적인 영향을 미칠 수 있으므로 수명주기 초기단계부터 각 단계별로 품질활동을 실시하고 그 결과물을 다음 단계로 피드백하여 필요한 관리를

병행할 필요가 있다. 소프트웨어 개발에서의 품질관리 문제점은 개발이 완료된 후에 품질평가를 실시한다는 점이었다. 이는 소프트웨어의 성능과 품질을 저하시키는 요인이 되기 때문에, 고품질의 소프트웨어 확보를 위해서는 품질활동이 수명주기 전 단계에 걸쳐 실시되어야 하며 각 단계에서 소프트웨어의 활동 결과가 다음 단계에서의 입력이 되는 피드백과 수정과정이 반복적으로 이루어져야 함을 의미한다.

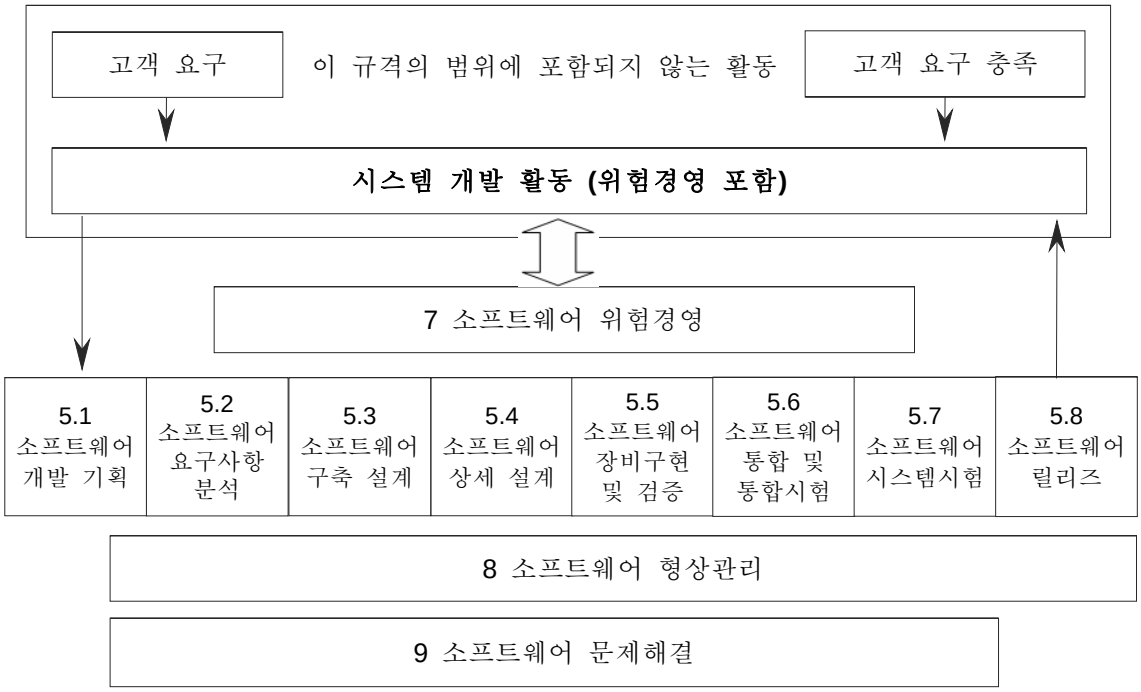
소프트웨어 수명주기 프로세스, 활동 및 업무에 대하여 ISO/IEC 12207은 모든 사업 분야에서 적용할 수 있는 규격임에 비하여 2006년 5월 9일 JTC 1 SC 62A JWG에서 발간한 IEC 62304규격은 현재 FDIS판이지만, 의료기기 분야에서 적용할 수 있는 소프트웨어 수명주기 프로세스[21] 규격으로 작성된 것이다. 규격 ISO/IEC 12207에서 소프트웨어 제품(software product)의 정의는 '컴퓨터 프로그램, 절차서, 그리고 가능한 관련 문서와 자료의 집합'으로 규정하고 있는데 비하여 IEC 62304에서는 '의료기기 소프트웨어(Medical Device Software)'로 한정하고 있으며 이에 대한 정의를 '개발 중인 의료기기에 채택할 목적으로 개발된 소프트웨어 시스템 또는 자체 권한으로 의료기기로 사용하기 위한 소프트웨어 시스템컴퓨터 프로그램'으로 규정하고 있다.

IEC 62304 규격은 의료기기 소프트웨어의 개발, 운용 및 유지보수에 내포된 프로세스, 활동 및 업무를 포함하고, 시스템의 수명을 요구사항의 정의에서부터 제조를 위한 릴리즈에 이르기까지 소프트웨어의 수명 전체를 포괄하는 개념적 구조(life cycle model)에서 의료기기 소프트웨어의 안전한 설계와 유지보수에 필요한 활동과 업무에 관련된 수명주기 프로세스의 체제를 제공한다.

IEC 62304 규격은 의료기기 소프트웨어 개발과 유지보수에 적용하며, 소프트웨어는 의료기기의 서브 시스템(Sub-system) 또는 그 자체가 의료기기로 간주한다. 수명주기 프로세스는 소프트웨어 개발 프로세스와 유지관리 프로세스로 크게 분류하며, 수명주기 프로세스에 대한 요구사항을 제시하며, 각각의 수명주기 프로세스를 일련의 활동으로 다시 분류하고, 각 활동을 일련의 업무로 분류하여, 품질경영시스템과 위험경영시스템의 범위 안에서 개발•유지하도록 규정하고 있는데, 이는 현장에서 발생하는 대부분의 사고는 부적합한 소프트웨어 갱신과 업그레이드를 포함해 의료기기 시스템의 정비나 유지보수에 관련되므로 소프트웨어 유지관리 프로세스는 소프트웨어 개발 프로세스만큼

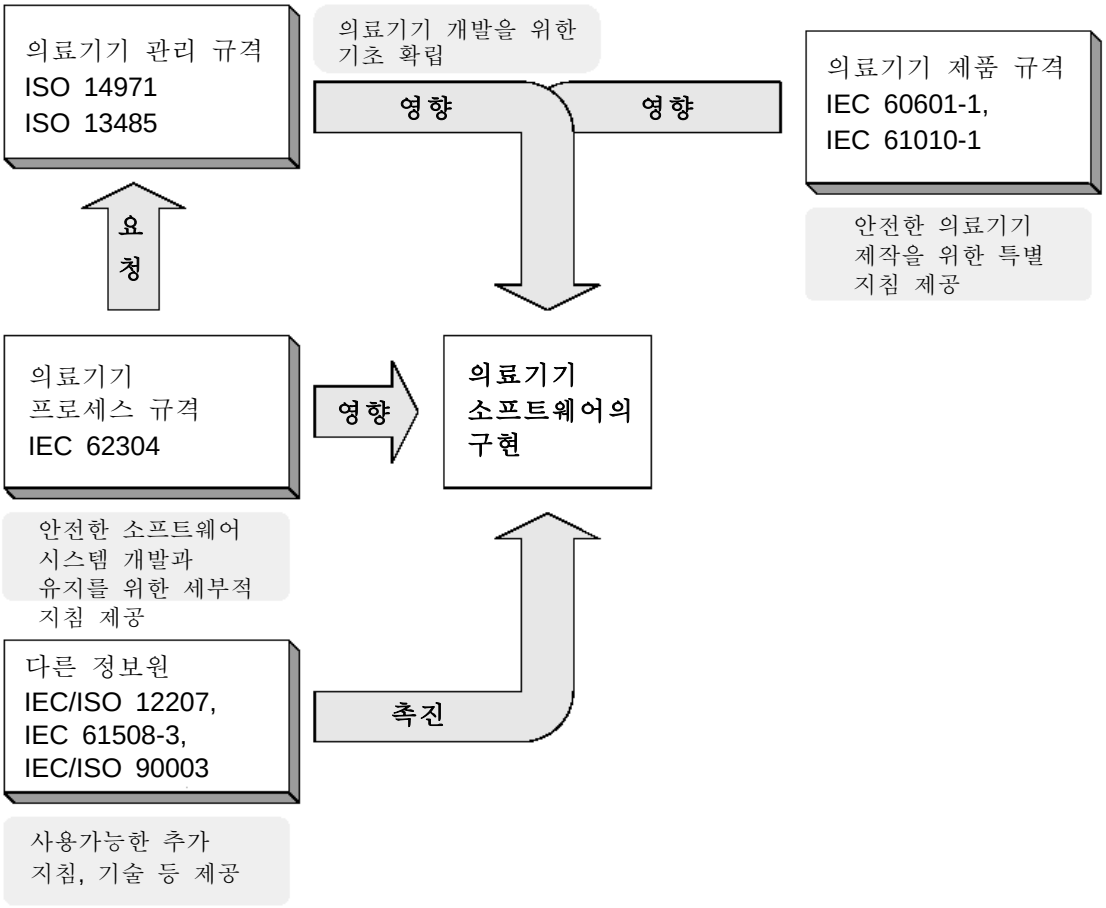
21) 규격에서의 수명주기 모델은 V모델로, 폭포수형 모델은 Pure Waterfall, Winston W. Royce가 처음 개발한 소프트웨어의 개발 수명주기 전략으로 프로젝트를 관리 가능한 단계(요구사항, 설계, 구현, 테스트 등)로 분류하고 마일스톤(Milestone)과 문서화 작업을 수립하며 각 단계의 종료마다 검토(review)하는 방법임. 즉, 폭포수형 모델은 문서중심의 모델로, 이는 현 단계에서 다음단계로 이동되는 주요 산출물이 문서임을 의미.

중요하게 취급해야 한다. 소프트웨어 유지보수 프로세스는 소프트웨어 개발 프로세스와 매우 유사하다. 개발 프로세스 활동에 대하여 그림 2-1에 표시되어 있다.



림 2-1] 소프트웨어 개발 프로세스 및 활동의 개요

이 규격과 관련하여 ISO 13485(품질경영시스템) 및 ISO 14971(위험경영시스템)과 같은 의료기기 국제규격은 제품을 개발하는 조직을 위한 기초를 형성하는 관리 환경을 제공한다. IEC 60601-1(의료용 전기기기의 기본적 안전성 및 필수 성능에 관한 공통 규격) 및 IEC 61010-1(측정, 제어 및 시험실용 전기기기의 안전에 대한 일반 요구사항)과 같은 안전 관련규격은 안전한 의료기기의 개발을 위하여 특별히 적용되는 지침을 제공한다. 소프트웨어가 이러한 의료기기의 일부일 경우, 이 규격은 안전한 의료기기 소프트웨어 개발과 유지관리에 필요한 보다 상세한 지시를 제공한다. ISO/IEC 12207(소프트웨어 수명주기 프로세스), IEC 61508-3(시스템 관련 전기·전자/프로그램이 가능한 전기적 안전의 기능적 안전성) 및 ISO/IEC 90003(컴퓨터 소프트웨어에 ISO 9001:2000 적용을 위한 지침)과 같은 많은 다른 국제규격은 이 규격의 요구사항 구현에 사용할 수 있는 방법, 도구 및 기술로서 간주할 수 있다. 그림 2-2에는 이러한 다른 해당 규격과의 관계를 표시한다.



[그림 2-2] IEC 62304와 다른 의료기기 국제규격과의 관계

IEC 62304는 ISO/IEC 12207과 상반되는 것이 아니다. 일반적인, 즉 의료기기에 제한되지 않는 소프트웨어 수명주기 프로세스에 대한 요구사항을 정의하는 ISO/IEC 12207 접근방식과 개념에서 도출된 규격이다. 이 규격은 의료기기 분야의 요구에 따라 안전성 측면 및 ISO 14971(의료기기 위험경영시스템)에 집중하고, 소프트웨어 개발이 품질 시스템에 포함된다는 사실을 고려하며, 규제 환경에 유용한 적절한 프로세스를 선택할 필요성에 때문에 ISO/IEC 12207에서의 변경이 발생한 것이다.

IEC 62304 규격에서 주로 다음과 관련된 부분이 ISO/IEC 12207과 다르다.

- 시스템 요구사항, 시스템 아키텍처 및 밸리데이션과 같은 시스템 관련 측면을 배제한다.
- 의료기기에 대해 다른 규격에 문서화되고 중복되는 활동이라고 간주되는 프로세스 일부를 생략한다.
- (안전성) 위험경영 프로세스와 소프트웨어 릴리즈 프로세스를 추가한다.
- 프로세스를 지원하는 문서와 검증을 개발과 유지보수 프로세스에 포함시킨다.

- 각 프로세스의 프로세스 구현과 기획을 개발과 유지보수 프로세스의 단일 활동으로 통합한다.
- 안전성 필요와 관련한 요구사항을 분류한다.
- 프로세스를 일차 또는 지원 프로세스로 명백히 분류하지 않으며, 프로세스를 그룹으로 구성하지 않는다.

표 2-1에 IEC 62304와 ISO/IEC 12207과의 관계를 표시한다.

ISO/IEC 62304 프로세스		ISO/IEC 12207 프로세스	
활동	업무	활동	업무
5 소프트웨어 개발 프로세스		5.3 개발 프로세스 6.1 문서화 프로세스 6.2 형상관리 프로세스 6.4 검증 프로세스 6.5 밸리데이션 프로세스 6.8 문제해결 프로 7.1 경영 프로세스	
5.1 소프트웨어 개발 기획		5.3.1 프로세스 구현 5.3.3 소프트웨어 구축 설계 5.3.7 소프트웨어 코딩 및 시험 5.3.8 소프트웨어 통합 5.3.9 소프트웨어 자격검증 시험 5.3.10 시스템 통합 6.1.1 프로세스 구현 6.2.1 프로세스 구현 6.2.2 형상 식별 6.4.1 프로세스 구현 6.5.1 프로세스 구현 6.8.1 프로세스 구현 7.1.2 기획 7.1.3 실행 및 관리 7.2.2 기반구조의 확립 7.2.3 기반구조의 유지보수	
	5.1.1 소프트웨어 개발 계획	5.3.1 프로세스 구현 7.1.2 기획	5.3.1.1 5.3.1.3 5.3.1.4 7.1.2.1
	5.1.2 소프트웨어 개발 계획 갱신 유지	7.1.3 실행 및 관리	7.1.3.3
	5.1.3 시스템 설계와 개발에 대한 소프트웨어 개발 계획 참조	5.3.3 시스템 구축 설계 5.3.10 시스템 통합 6.5.1 프로세스 구현	5.3.3.1 5.3.10.1 6.5.1.4

	5.1.4 소프트웨어 개발 규격, 방법 및 도구 기획	5.3.1 프로세스 구현	5.3.1.3 5.3.1.4
	5.1.5 소프트웨어 통합 및 통합 시험 기획	5.3.8 소프트웨어 통합	5.3.8.1
	5.1.6 소프트웨어 검증 기획	6.4.1 프로세스 구현 5.3.7 소프트웨어 코딩 및 시험 5.3.8 소프트웨어 통합 5.3.9 소프트웨어 자격검증 시험	6.4.1.4 6.4.1.5 5.3.7.5 5.3.8.5 5.3.9.3
	5.1.7 소프트웨어 위험경영 기획	개정 1:2002 – F 3.15 위험경영 프로세스	
	5.1.8 문서화 기획	6.1.1 프로세스 구현	6.1.1.1
	5.1.9 소프트웨어 형상관리 기획	6.2.1 프로세스 구현 6.8.1 프로세스 구현	6.2.1.1 6.8.1.1
	5.1.10 관리할 지원 항목	7.2.2 기반구조의 확립 7.2.3 기반구조의 유지보수	7.2.2.1 7.2.3.1
	5.1.11 검증 전 소프트웨어 형상 항목(item) 식별	6.2.2 형상 식별	6.2.2.1
5.2 소프트웨어 요구사항 분석		5.3.3 시스템 구축 설계 5.3.4 소프트웨어 요구사항 분석 6.4.2 검증	
	5.2.1 시스템 요구사항에서 소프트웨어 요구사항을 정의하고 문서화	5.3.3 시스템 구축 설계	5.3.3.1
	5.2.2 소프트웨어 요구사항 내용	5.3.4 소프트웨어 요구사항 분석	5.3.4.1
	5.2.3 소프트웨어 요구사항에 위험관리 방법 포함		
	5.2.4 의료기기 위험분석 재평가		없음
	5.2. 시스템 요구사항 갱신	5.3.4 소프트웨어 요구사항 분석	a) b)
	5.2.6 소프트웨어 요구사항 검증	5.3.4 소프트웨어 요구사항 분석 6.4.2 검증	5.3.4.2 6.4.2.3
5.3 소프트웨어 구축 설계		5.3.5 소프트웨어 구축 설계	
	5.3.1 소프트웨어 요구사항을 아키텍처로 변형	5.3.5 소프트웨어 구축 설계	5.3.5.1

	5.3.2 소프트웨어 항목의 연계성을 위한 아키텍처 개발		5.3.5.2
	5.3.3 SOUP 항목의 기능 및 성능 요구사항 명시		없음
	5.3.4 SOUP 항목에 필요한 시스템 하드웨어와 소프트웨어 명시		없음
	5.3.5 위험관리에 필요한 분리 식별		없음
	5.3.6 소프트웨어 아키텍처 검증	5.3.5 소프트웨어 구축 설계	5.3.5.6
5.4 소프트웨어 상세 설계		5.3.6 소프트웨어 상세 설계 6.4.2 검증	
	5.4.1 소프트웨어 아키텍처를 소프트웨어 단위로 개량	5.3.6 소프트웨어 상세 설계	5.3.6.1
	5.4.2 각 소프트웨어 단위에 대해 상세 설계 개발		
	5.4.3 연계성에 대한 상세 설계 개발		5.3.6.2
	5.4.4 상세 설계 검증	6.4.2 검증	5.3.6.7
5.5 소프트웨어 유닛 구현 및 확인		5.3.6 소프트웨어 상세 설계 5.3.7 소프트웨어 코딩 및 시험 6.4.2 검증	
	5.5.1 각 소프트웨어 단위 구현	5.3.7 소프트웨어 코딩 및 시험	5.3.7.1
	5.5.2 소프트웨어 단위 검증 프로세스 확립	5.3.6 소프트웨어 상세 설계 5.3.7 소프트웨어 코딩 및 시험	5.3.6.5 5.3.7.5
	5.5.3 소프트웨어 단위 인수 기준	5.3.7 소프트웨어 코딩 및 시험	5.3.7.5
	5.5.4 소프트웨어 단위 추가 인수 기준	5.3.7 소프트웨어 코딩 및 시험 6.4.2 검증	5.3.7.5 6.4.2.5
	5.5.5 소프트웨어 단위 검증	5.3.7 소프트웨어 코딩 및 시험	5.3.7.2
5.6 소프트웨어		5.3.8 소프트웨어 통합	

통합 및 통합 시험		5.3.9 소프트웨어 자격검증 시험 5.3.10 시스템 통합 6.4.1 프로세스 구현 6.4.2 검증	
	5.6.1 소프트웨어 단위 통합	5.3.8 소프트웨어 통합	5.3.8.2
	5.6.2 소프트웨어 통합 검증	5.3.8 소프트웨어 통합 5.3.10 시스템 통합	5.3.8.2 5.3.10.1
	5.6.3 통합 소프트웨어 시험	5.3.9 소프트웨어 자격검증 시험	5.3.9.1
	5.6.4 통합 시험 내용		5.3.9.3.
	5.6.5 통합 시험 절차 검증	6.4.2 검증	6.4.2.2
	5.6.6 회귀시험 수행	5.3.8 소프트웨어 통합	5.3.8.2
	5.6.7 통합시험 기록 내용	5.3.8 소프트웨어 통합	5.3.8.2
	5.6.8 소프트웨어 문제해결 프로세스의 사용	6.4.1 프로세스 구현	6.4.1.6
5.7 소프트웨어 시스템 시험		5.3.8 소프트웨어 통합 5.3.9 소프트웨어 자격검증 시험 6.4.1 프로세스 구현 6.4.2 검증 6.8.1 프로세스 구현	
	5.7.1 각 소프트웨어 요구사항에 대한 시험 확립	5.3.8 소프트웨어 통합 5.3.9 소프트웨어 자격검증 시험	5.3.8.4 5.3.9.1
	5.7.2 소프트웨어 문제해결 프로세스의 사용	6.4.1 프로세스 구현	6.4.1.6
	5.7.3 변경 후 재시험	6.8.1 프로세스 구현	6.8.1.1
	5.7.4 소프트웨어 시스템 시험 검증	6.4.2 검증 5.3.9 소프트웨어 자격검증 시험	6.4.2.2 5.3.9.3
	5.7.5 각 시험 소프트웨어 시스템 시험 기록 내용에 대한 데이터 문서화	5.3.9 소프트웨어 자격검증 시험	5.3.9.1
5.8 소프트웨어 릴리즈		5.3.9 소프트웨어 자격검증 시험 5.4.2 작동 시험 6.2.5 형상 평가 6.2.6 릴리즈 관리 및 인도	
	5.8.1 소프트웨어 검증의	5.4.2 작동 시험	5.4.2.1

	완벽성 확인(ensure)	6.2.6 릴리즈 관리 및 인도	5.4.2.2 6.2.6.1
	5.8.2 확인(known)된 잔류 비정상 상태 문서화	6.2.5 형상 평가 5.3.9 소프트웨어 자격검증 시험	6.2.5.1 5.3.9.3
	5.8.3 확인(known)된 잔류 비정상 상태 평가		
	5.8.4 릴리즈 버전의 문서화	6.2.6 릴리즈 관리 및 인도	6.2.6.1
	5.8.5 릴리즈한 소프트웨어를 작성한 방법 문서화		
	5.8.6 활동과 업무의 완결 확인(ensure)		
	5.8.7 소프트웨어 저장		
	5.8.8 소프트웨어 릴리즈의 반복 정밀도 보장		
6 소프트웨어 유지보수 프로세스		5.5 유지보수 프로세스 6.2 형상관리 프로세스	
6.1 소프트웨어 유지보수 계획 확립		5.5.1 프로세스 구현	5.5.1.1
6.2 문제 및 수정 분석		5.5.1 프로세스 구현 5.5.2 문제 및 코드화 분석 5.5.3 수정 구현 5.5.5 감소(마이그레이션)	
	6.2.1 피드백 기록 및 평가		
	6.2.1.1 피드백 모니터링	5.5.1 프로세스 구현	5.5.1.1 5.5.1.2
	6.2.1.2 피드백 문서화 및 평가		
	6.2.1.3 안전에 관한 문제 보고서 영향의 평가	5.5.2 문제 및 수정 분석	5.5.2.1 5.5.2.2 5.5.2.3 5.5.2.4
	6.2.2 소프트웨어 문제해결 프로세스의 사용	5.5.1 프로세스 구현	5.5.1.2
	6.2.3 변경 요청 분석	5.5.2 문제 및 수정 분석	5.5.2.1
	6.2.4 변경 요청 승인	5.5.2 문제 및 수정 분석	5.5.2.5

	6.2.5 사용자 및 규제자와의 의사소통	5.5.3 수정 구현 5.5.5 마이그레이션	5.5.3.1 5.5.5.3
6.3 수정 구현		5.5.3 수정 구현 6.2.6 릴리즈 관리 및 인도	
	6.3.1 확립된 프로세스를 사용해 수정 구현	5.5.3 수정 구현	5.5.3.2
	6.3.2 수정한 소프트웨어 시스템의 재 릴리즈	6.2.6 릴리즈 관리 및 인도	6.2.6.1
7 소프트웨어 위험경영 프로세스		개정 1:2002 – F 3. 15 위험경영 프로세스 62304의 프로세스는 개정 1에서 취급되지 않은 위험/위험요인 현안을 취급한다. 일부 공통점이 있지만(위험 측정 등) 분석의 초점은 전혀 다르다.	
8 소프트웨어 형상관리 프로세스		5.5 유지보수 프로세스 6.2 형상관리 프로세스	
8.1 형상 식별		6.2.2 형상 식별	
	8.1.1 형상 항목 식별을 위한 방법 확립	6.2.2 형상 식별	6.2.2.1
	8.1.2 SOUP 식별		없음
	8.1.3 시스템 형상 문서 식별	6.2.2 형상 식별	6.2.2.1
8.2 변경관리		5.5.3 수정 구현 6.2.3 형상관리	
	8.2.1 변경 요청 승인	6.2.3 형상관리	6.2.3.1
	8.2.2 변경 구현	5.5.3 수정 구현 6.2.3 형상관리	5.5.3.2 6.2.3.1
	8.2.3 변경 검증	6.2.3 형상관리	6.2.3.1
	8.2.4 변경 추적성에 대한 방법 제공		
8.3 형상 상태 설명		6.2.4 형상 상태 설명	6.2.4.1
9 소프트웨어 문제해결 프로세스		5.5 유지보수 프로세스 6.2 형상관리 6.8 문제해결 프로세스	
9.1 문제 보고서 작성		6.8.1 프로세스 구현 6.8.2 문제해결	6.8.1.1 b) 6.8.2.1

9.2 문제 조사		6.8.2 문제해결 6.8.1 프로세스 구현	6.8.2.1 6.8.1.1 b)
9.3 관련 당사자에게 통보		6.8.1 프로세스 구현	6.8.1.1 a)
9.4 변경관리 프로세스 사용		6.2.3 형상관리 5.5.3 수정 구현	
9.5 기록 유지		6.8.1 프로세스 구현	6.8.1.1 a)
9.6 문제 경향 분석		6.8.1 프로세스 구현 6.8.2 문제해결	6.8.1.1 b) 6.8.2.1
9.7 소프트웨어 문제해결 검증		6.8.1 프로세스 구현	6.8.1.1 d)
9.8 시험 문서의 내용			12207의 모든 시험 업무 문서화

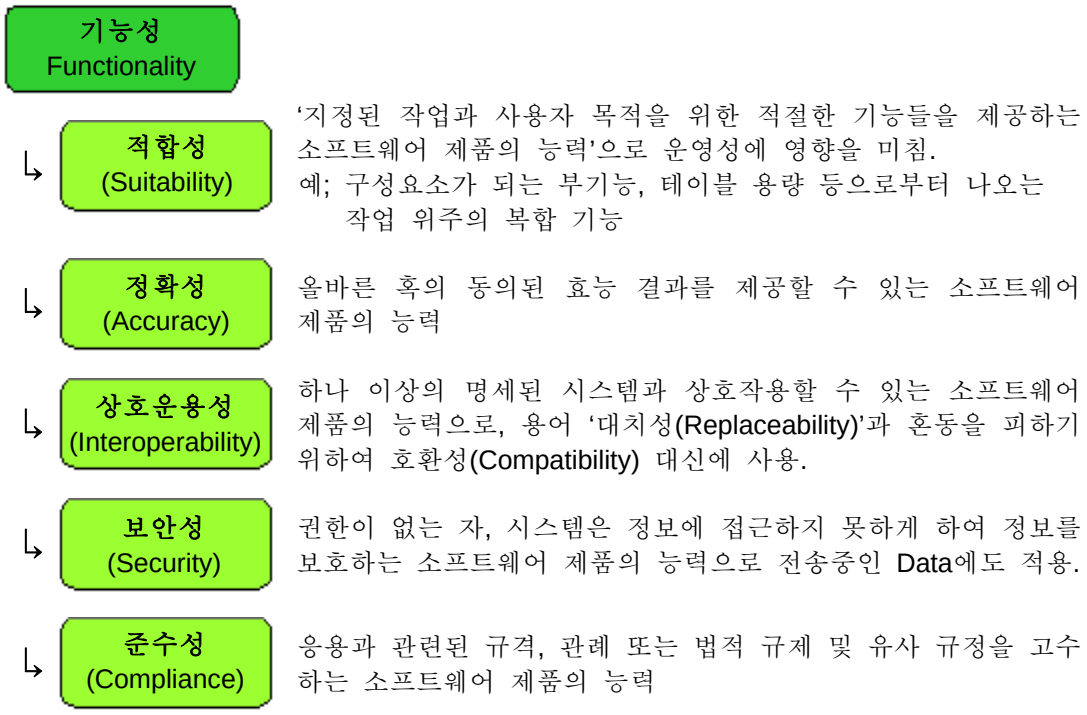
[표 2-1] IEC 62304와 ISO/IEC 12207과의 관계

1991년에 제정된 ISO/IEC 9126은 품질특성 및 메트릭을 정의하고 있는 국제규격으로서 소프트웨어 품질을 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성의 6가지 특성으로 표현하며, 각 품질특성별 세부 메트릭을 제시하고 있다. 메트릭은 소프트웨어 개발과정에서 개발자들이 적용할 수 있는 내부 메트릭과 소프트웨어 사용자들이 개발 초기나 개발 완료 후에 적용할 수 있는 외부 메트릭이 있다. ISO/IEC 9126은 소프트웨어 제품에 대한 품질 요구사항을 기술하는데 사용할 수 있으며 개발 중에 있거나 또는 개발 완료된 소프트웨어의 품질을 측정하는데 척도로 사용될 수 있다.

소프트웨어 품질특성은 공업제품의 품질특성보다 더욱 다양하고 복잡한 특성을 갖고 있기 때문에 여러 가지 품질특성을 고려하여 평가하여야 한다. 이러한 소프트웨어의 품질특성은 크게 외부특성과 내부특성으로 분류되고, 각 특성들은 다시 인자(요소)들로 구성되어 소프트웨어의 품질평가를 가능하게 한다. 어떤 제품을 비교 평가하는 경우, 동일 품질특성을 의미하는 어휘를 다른 의미로 이용하거나 동일특성에 대한 의미를 다른 어휘로 사용한다면 명확한 평가가 이루어질 수 없다. 소프트웨어 제품 품질평가에 관한 국제규격인 ISO/IEC 9126에서 정의하고 있는 품질특성은 기능성, 신뢰성, 사용성, 효율성, 유지보수성, 이식성으로 각 품질특성의 개념은 다음과 같다.

• 기능성

소프트웨어가 특정 조건에서 사용될 때, 명시된 요구와 내재된 요구를 만족하는 기능을 제공하는 소프트웨어 제품의 능력으로, 다른 특성들은 주로 소프트웨어가 언제, 어떻게 하는가에 관심을 두는 비해 이 특성은 요구를 충족하기 위해서 소프트웨어가 무엇을 하는가에 관심을 둔다. 이 특성의 명시된 요구와 내재된 요구에 대해서는 품질의 정의부분에 기술된 참고를 적용한다.



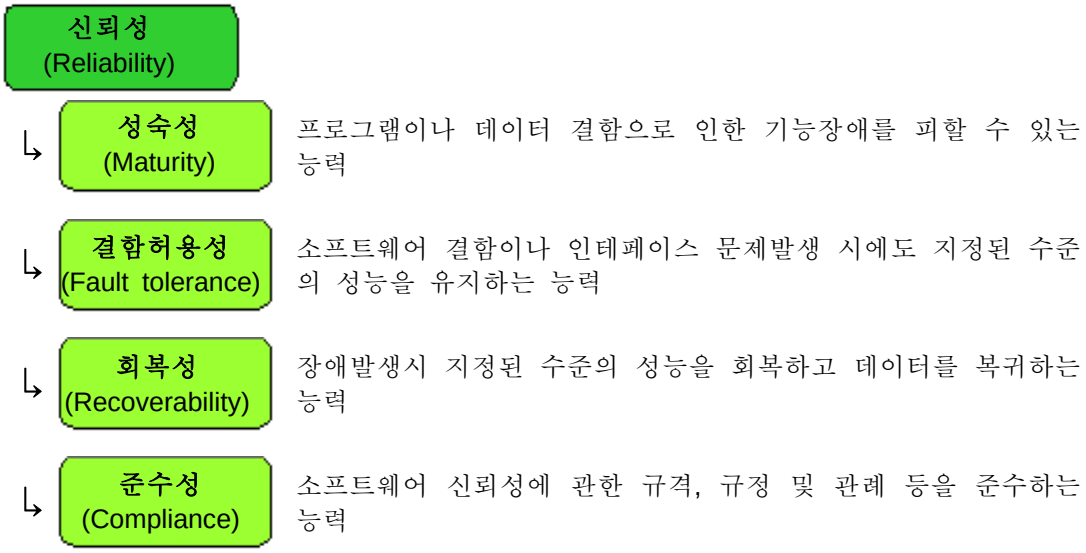
[그림 2-3기능성의 품질부특성

• 신뢰성

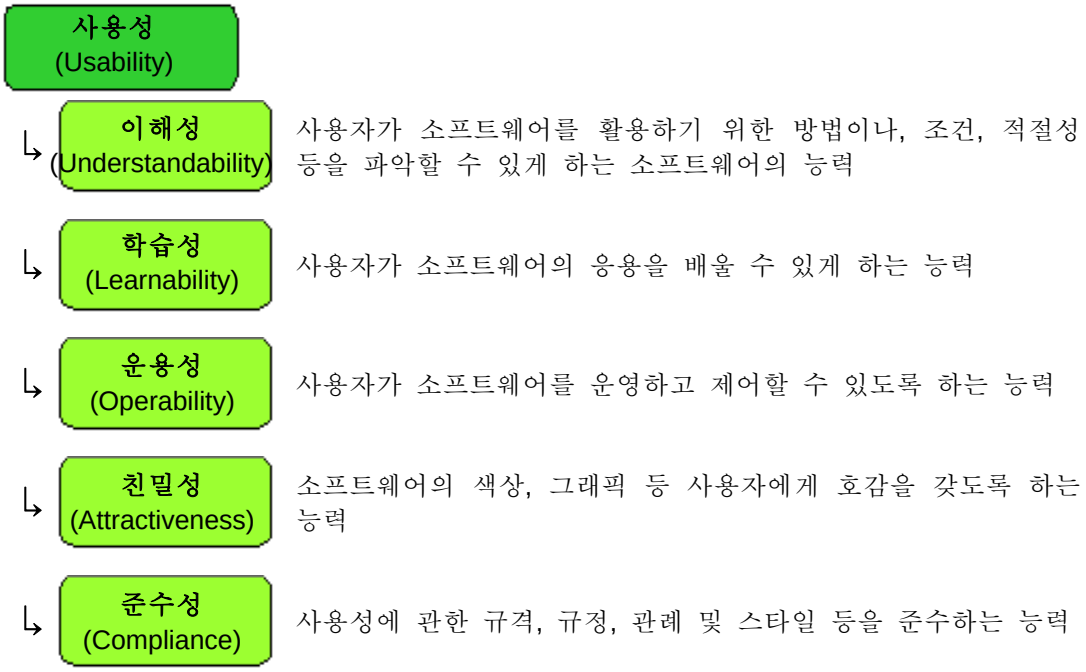
명세된 조건에서 소프트웨어를 사용할 경우, 성능을 유지할 수 있는 소프트웨어 제품의 능력으로 신뢰성의 한계는 요구사항, 설계 및 구현상의 결함에 기인한다. 결함으로 인한 고장은 사용 경과 시간보다는 소프트웨어 제품의 사용방법과 선정된 프로그램 선택사항에 따라 달라질 수 있다. (그림 2-4 참조)

• 사용성

소프트웨어를 규정된 조건에서 사용할 경우, 영향을 받거나 의존하는 운영자, 최종 사용자 및 간접 사용자가 이해, 학습, 사용하며 선호할 수 있는 소프트웨어 제품의 능력으로서 기능성, 신뢰성, 효율성 등의 몇몇 특징들은 사용성에 영향을 줄 수 있지만 ISO/IEC 9126의 목적상 사용성으로는 분류하지 않는다. (그림 2-5 참조)



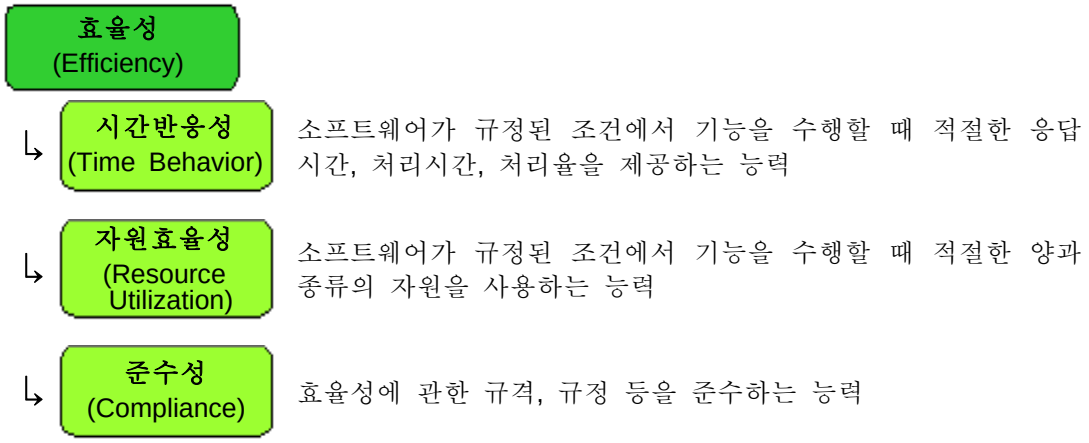
[그림 2-4신뢰성의 품질부특성



[그림 2-5] 사용성의 품질부특성

- 효율성

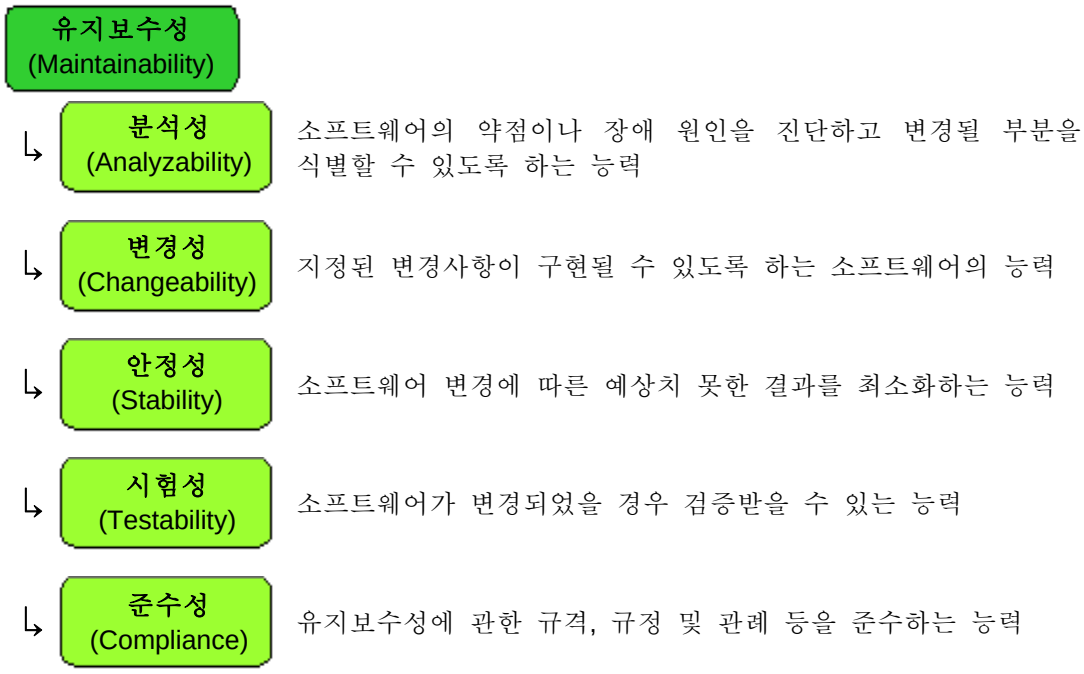
명시된 조건에서 사용되는 자원(다른 소프트웨어 제품, 하드웨어 장비, 재료-인쇄용지나 디스켓 등을 포함)의 양에 따라 요구된 성능을 제공하는 소프트웨어 제품의 능력으로서, 효율성의 품질부특성은 그림 2-6와 같다.



[그림 2-6] 효율성의 품질부특성

• 유지보수성

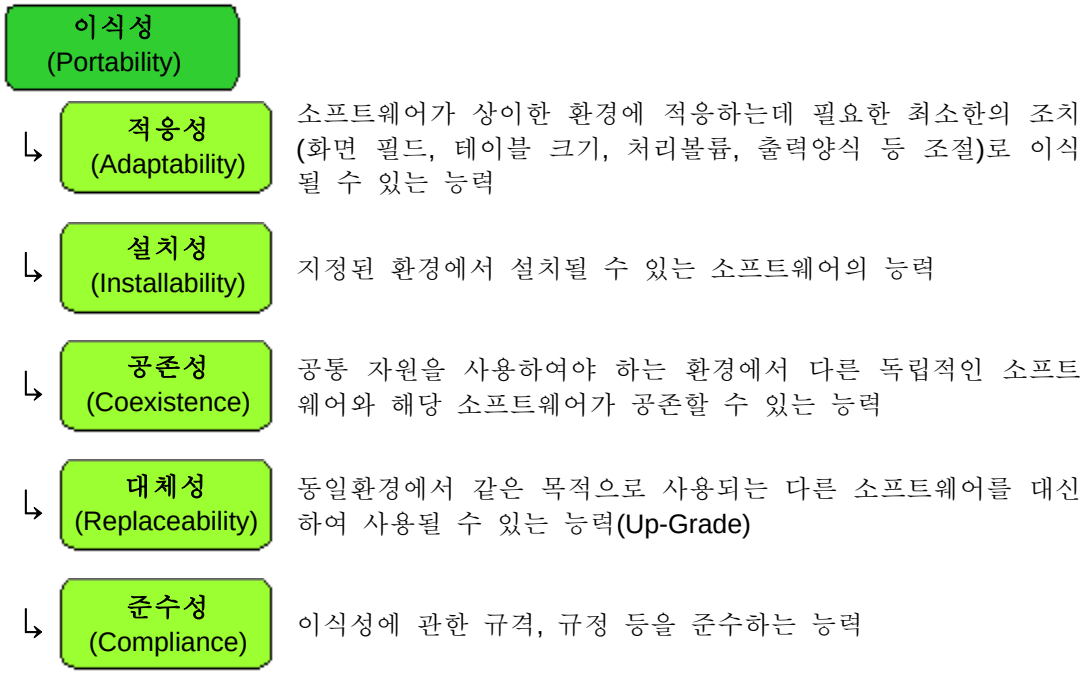
소프트웨어 제품이 변경되는 능력으로, 변경에는 환경과 요구사항 및 기능적 명세에 따른 소프트웨어의 수정, 개선 등이 포함된다.



[그림 2-7] 유지보수성의 품질부특성

• 이식성

조직, 하드웨어 혹은 소프트웨어를 포함하는 하나의 환경에서 다른 환경으로 전이될 수 있는 소프트웨어 제품의 능력



[그림 2-8] 이식성의 품질부특성

소프트웨어 제품의 품질을 측정하거나 평가하는데 필요한 방법과 절차를 정의하고 있는 ISO/IEC 14598은 최초 ISO/IEC 9126:1991에 품질 측정방법과 절차를 개략적으로 포함하여 규정되었으나 1994년부터 별도 규격으로 제정작업을 추진하여 현재는 6개의 계열규격에서 품질 평가주체를 소프트웨어 개발자, 구매자, 평가자로 구분하여 규격화 되었다.

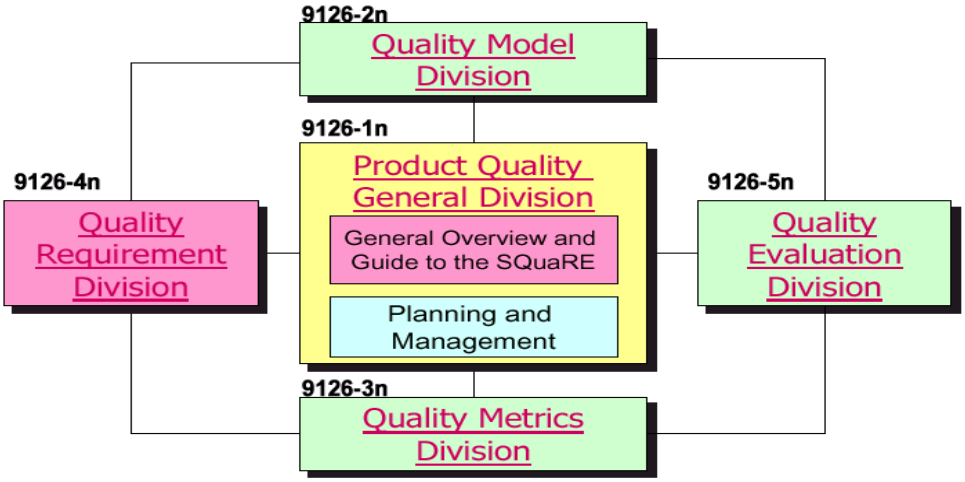
평가를 시행하는 절차로는 평가 요구사항 도출, 평가명세서 작성, 평가계획수립 평가수행 및 결과도출 등의 단계를 제시하고 있고 평가명세서를 작성할 때는 ISO/IEC 9126에 따른 내·외부 메트릭을 활용하도록 하고 있다.
메트릭을 적용하여 평가를 수행하는 과정에서 객관성 및 프로세스 확보를 위하여 각 메트릭 적용절차 및 기준 등을 명시한 평가모듈 라이브러리를 이 규격에서 제공하고 있기도 하다.

현재 ISO/IEC 14598과 ISO/IEC 9126 계열규격에 대한 전면적인 개정을 검토 중으로서, 이는 ISO 12119:1994(패키지 소프트웨어 품질요구사항 및 테스팅 지침) 규격의 제정 이후, 전 세계적으로 사용해 오면서 소프트웨어의 품질은 품질 모델에 기초하여 품질 요구사항에 대비되는 메트릭을 사용 평가해야 하므로 ISO/IEC 9126(소프트웨어 제품 품질 특성)과 이를 평가하는 ISO/IEC 14598을 보강·통합한 새로운 체계(Architecture)가

요구되었기 때문이다. 이러한 요구에 따라 ISO/IEC JTC1/SC7/WG6[22]은 새로운 소프트웨어 평가 모델인 SQuaRE (Software Quality Requirements and Evaluation)프로젝트를 진행 중이며, SQuaRE의 구축 후에는 ISO/IEC 9126과 ISO/IEC 14598 계열규격을 ISO/IEC 25000 계열규격으로 대체하는 것을 목표로 추진 중이다.

SQuaRE는 그림 2-9과 같이 □□4 + 1□□ 구조를 갖는다.

- 품질 모델(25010 Quality Model Division)
- 품질 메트릭(25020 Quality Metrics Division)
- 품질 요구사항(25030 Quality Requirement Division)
- 품질 평가(25040 Quality Evaluation Division)
- +(plus) 전체를 반영하는 부분(25000 Quality Management Division)



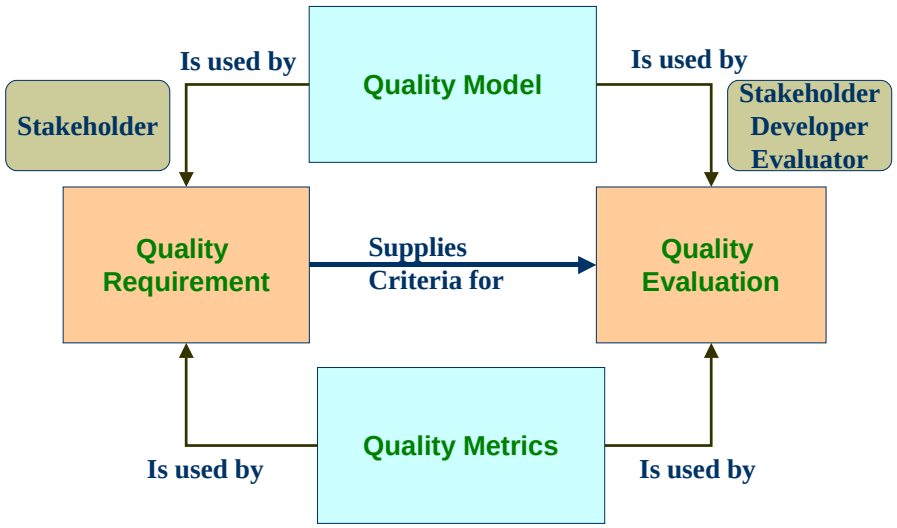
[그림 2-9] SQuaRE 체계(Architecture)

SQuaRE는 기존의 ISO/IEC 9126과 14598을 통합하는 과정에서 문서의 일관성 유지를 위하여 약간 또는 다수의 문서 내용을 변경(Minor/Major Editorial Revision)하고, 기술적으로 다수의 부분도 개정(Major Technical Revision)하였으며, 필요에 따라 새로운 하부 프로젝트를 도입(New Subproject Proposal)하였다. 기본적으로 ISO/IEC 9126 및 ISO/IEC 14598의 통합과정에서 개발된 SQuaRE 모델과 세부적으로 측정 개념(Concept)의 명확화, 측정 프리미티브(Measurement Primitives)의 정의, 기능적(Functional) 요구사항을 포함하는 품질 요구사항(Quality requirement) 등에 대한 내용이 변화의 근간을 이루고 있다.

22) ISO/IEC JTC1/SC7/WG6은 범용 소프트웨어 제품의 품질특성 및 평가, COTS 품질요구사항 및 제3자 평가에 관한 규격을 수립함에 그 목적을 두고 있음.

새로이 도입된 부분은 SQuaRE의 전체적인 개관(Overview)과 전체적 SQuaRE 체계 또는 구조모델(Architecture Model)에 대한 소개 'SQuaRE 개관 및 소개(General Overview and Guide to the SQuaRE)', 메트릭 참조모델에 대한 소개 '메트릭 참조모델과 소개(Metrics Reference Model and Guide)', 소프트웨어 개발 수명주기에서 공통 사용되는 메트릭에 대한 설명 '기본 메트릭(Base Metrics)', 소프트웨어 품질 요구사항과 추적, 타당성 및 관련성을 다루는 '품질 요구사항(Quality Requirement)' 등이다. 기술적으로 다수 개정된 내용은 '외부 메트릭(External Metrics)', '사용자 관점의 품질 메트릭(Quality In Use Metrics)' 등이며 기타 부분은 규격 문서의 일관성 유지를 위해 약간 또는 상당부분 편집한 것이다.

SQuaRE 모델과 기술 수준의 관계는 그림 2-10와 같이 소프트웨어 제품 관련자(Stakeholders)가 품질 요구사항을 제공하고 이러한 요구사항들이 품질평가(Product Quality Evaluation)를 어떤 범위에서 어떻게 수행할지 결정한다. 품질 측정(Product Quality Measurement)은 품질 요구사항과 품질 평가를 지원하는 역할을 하게 된다. 즉, 품질 모델과 품질 메트릭을 사용하여 품질 요구사항을 도출하고 도출된 요구사항은 품질평가를 위한 평가항목이나 기준으로 작용한다. 여기서 품질 모델과 품질 메트릭은 품질 평가를 위해서도 사용된다.



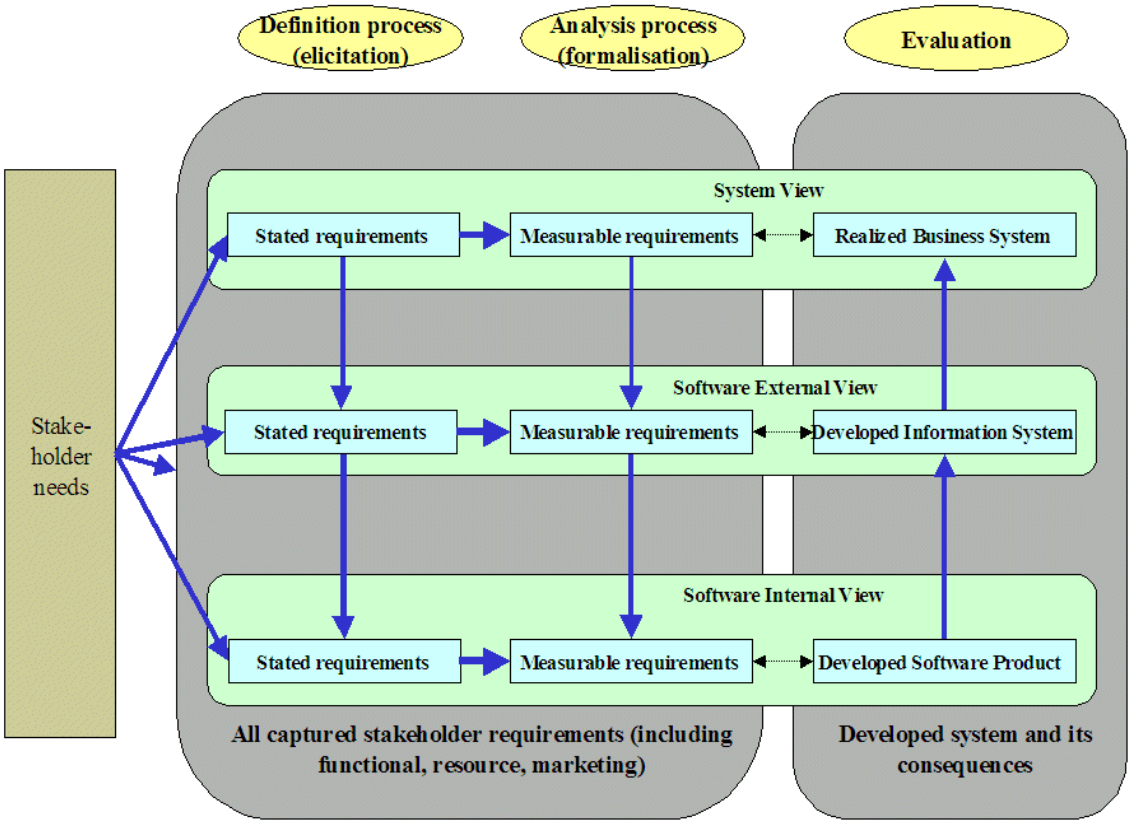
[그림 2-10] SQuaRE 모델을 기술 수준에서 본 관계 모델

측정 프리미티브(Measurement Primitives) 정의와 관련하여 SQuaRE에서의 메트릭 개념은 전체적으로 상호 연관관계를 갖는다. 품질 속성을 나타내는 측정 프리미티브가 측정과 데이터 수집을 통해 이루어지고, 품질 측정을 위하여 내·외부/QIU 품질 측정치가 필요하며 이는 품질부특성(Quality Sub-characteristic)을 나타낸다. 품질 측정치는 종합하여 품질 특성을 형성하게 되고 품질특성은 품질 평가에 필요하다. 이를 기반으로

품질평가보고서를 작성하여 소프트웨어 제품 품질을 평가하게 된다.

품질 요구사항(Quality Requirement)에서 요구명세서(Requirement Specification)는 기능적 요구사항, 비기능적 요구사항, 품질 요구사항을 포함하며, 품질 요구사항은 기능성 요구사항, 신뢰성 요구사항, 사용성 요구사항 등을 포함한다. 이러한 요구사항은 소프트웨어 관련자 (Stakeholders)의 요구(Needs)를 반영한 것으로 품질의 종류, 관련자의 종류, 개발 생명주기 등 여러 가지 측면에서 품질 요구사항을 볼 수 있다.

그림 2-11은 소프트웨어 내·외부 및 시스템 관점에서 요구사항과 평가와의 관계를 표시한 것으로서, 요구사항은 소프트웨어 제품 사용자, 판매자, 구매자 등의 관련자 (Stakeholders) 니즈를 정의하고 보다 상세하게 표현한 것이며, 이를 보다 형식적이고 구체적인 표현이 측정 가능한 요구사항(Measurable Requirement)이다.

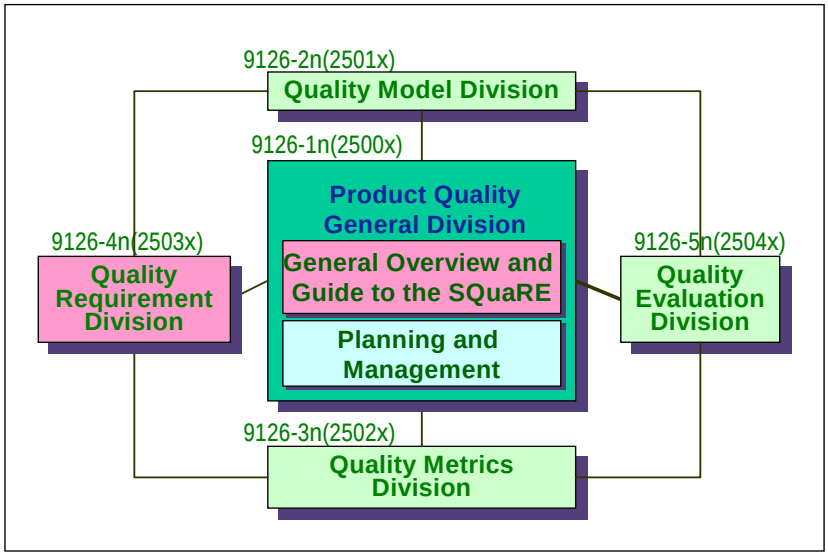


[그림 2-11] 소프트웨어 내/외부 및 시스템 관점에서의 요구사항과 평가와의 관계

측정 가능한 요구사항은 각각 소프트웨어 내부적 관점(Internal View), 외부적 관점(External View), 시스템 관점(System View)을 가질 수 있고, 이 관점들은 각각 개발된 소프트웨어 제품, 개발된 정보시스템, 현실화된(Realized) 비즈니스 시스템 평가를 위한 기본적인 재료로써 활용된다. 즉, 각 관점에서의 요구사항에 비춰 소프트웨어나 시스템이

얼마나 잘 구현되어 있는지 확인하는 것이 평가를 의미한다.

SQuaRE의 구조는 표 2-2 및 그림 2-12과 같이 구성되어 있다.



[그림 2-12] SQuaRE의 구조

규격	계열 규격	주요 내용
ISO/IEC 2500X	ISO/IEC 25000	SQuaRE에 대한 일반 개요 및 안내
	ISO/IEC 25001	계획과 관리(ISO/IEC 14598-2에 해당)
ISO/IEC 2501X	ISO/IEC 25010	품질 모델(ISO/IEC 9126-1에 해당) - 품질모델 및 품질모델 사용에 대한 안내
ISO/IEC 2502X	ISO/IEC 25020	메트릭 참조 모델과 안내
	ISO/IEC 25021	기본 메트릭(SQuaRE에서 새로 제정됨)
	ISO/IEC 25022	내부 메트릭(ISO/IEC 9126-3의 내부 메트릭)
	ISO/IEC 25023	외부 메트릭(ISO/IEC 9126-2의 외부 메트릭)
	ISO/IEC 25024	사용품질 메트릭(ISO/IEC 9126-4의 사용품질 메트릭)
	ISO/IEC 25025	평가모듈의 문서화(ISO/IEC 14598-6에 해당)
ISO/IEC 2503X	ISO/IEC 25030	품질요구 - 품질요구에 관한 일반적인 안내 - 품질요구를 위한 요구사항

		- 사용품질 요구사항 - 외부품질 요구사항 - 내부품질 요구사항
ISO/IEC 2504X	ISO/IEC 25040	평가 프로세스에 대한 개요
	ISO/IEC 25041	개발자의 프로세스(ISO/IEC 14598-3을 일부 변경)
	ISO/IEC 25042	구매자의 프로세스(ISO/IEC 14598-4를 일부 변경)
	ISO/IEC 25043	평가자의 프로세스(ISO/IEC 14598-5를 일부 변경)

[표 2-2] SQuaRE의 구조

ISO/IEC 15504 규격은 소프트웨어의 프로세서를 평가하고 개선함으로써 품질 및 생산성 향상을 위한 규격으로, 프로세스 평가 결과에 따른 적부 판정보다는 프로파일 형태의 능력 수준을 결정함으로써 점진적인 개선을 유도하고 있다는 점에서 소프트웨어 제품을 평가하는데 적용되는 ISO/IEC 9126, 14598과 차별화되며 보다 발전된 개념의 규격이라 할 수 있다.

ISO/IEC 15504에서는 소프트웨어 프로세서 영역을 ISO/IEC 12207(소프트웨어 수명주기 프로세스)에 준거하여 일차(Primary) 프로세스, 지원(Supporting) 프로세스, 조직(Organizational) 프로세스로 구분하여 각 프로세스 영역별로 프로세스 카테고리와 기본 프로세스를 정의하고, 이들 프로세스를 정립하여 수행하고 있는 수준에 따라 개발기관의 능력수준을 Incomplete, Performed, Managed Established, Predictable, Optimizing level 등 6단계로 구분하고 있다. 또한 이 규격에서는 소프트웨어 프로세스 및 능력수준에 대한 참조 모델을 제시함과 함께 참조모델을 토대로 실제로 소프트웨어 프로세스를 평가하기 위한 지침 및 심사원 자격 등에 관한 사항을 명시하고 있다.

프로세스 품질평가 SPICE(Software Process Improvement and Capability dEtermination)는 소프트웨어 개발업체의 프로세스 능력 평가를 위하여 제정된 ISO/IEC TR 15504계열 규격의 평가 모델을 실증적으로 검증하기 위한 목적으로 운영하고 있다. 현재 미국, 유럽, 캐나다, 멕시코, 북아시아(한국, 일본), 남아시아 등 5개 지역별 센터(LNC)에서 소프트웨어 프로세스 평가 및 심사자 양성업무를 수행하고 있으며 한국에는 KSPICE가 조직되어 운영 중에 있다. SPICE는 소프트웨어 수명주기 동안 조직의 능력 및 개발의 성숙성을 평가하는 내용으로 구성되어 있으며 표 2-3과 같이 6단계 수준으로 평가된다.

Level	프로세스 수준	평가 내용
0	불완전	프로세스가 구현되지 않거나 그 목적을 달성하지 못함
1	실행	정의된 목적은 달성하나 활동계획의 추적 불가능
2	관리	프로세스가 관리되고 있으며 산출물을 생산함
3	확립	조직 전반에 대한 규격화된 프로세스가 정의되어 있음
4	예측가능	프로세스별 정량적 데이터가 관리되고 일관성 있게 수행됨
5	최적화	프로세스가 최적하게 수행되며 지속적인 개선 절차가 확보됨

[표 2-3] SPICE 성숙도

소프트웨어에 대한 품질경영시스템 관련 규격인 ISO 9000-3:1991은 컴퓨터 소프트웨어 제조업체에서 소프트웨어의 개발, 공급, 설치 및 유지보수에 대한 ISO 9001:1994 (품질경영시스템) 적용을 위한 계열 지침(Guideline)이었으나 ISO 9001:1994 규격이 2000년에 전면 개정됨에 따라, ISO/TC 176/SC 2에서는 이 규격을 폐지하였다. 이후 ISO/IEC JTC 1의 SC 7(소프트웨어 및 시스템 엔지니어링)에서 ISO 90003: 2004 규격으로 새로이 제정하였다.

ISO 90003 규격은 조직에서 컴퓨터 소프트웨어의 획득, 지원, 개발, 작동 및 유지에 대한 ISO 9001(품질경영시스템)의 용이한 적용을 위한 지침으로, 이 규격에서는 ISO 9001 요구사항 내용과 순서를 일치시켜 기술적 사항, 수명주기 모델, 개발 절차, 활동 순서 및 조직적 구조 등을 언급하고 있다. 또한 JTC 1/SC 7에서 발행된 특성 때문에 소프트웨어 관련하여 ISO/IEC 12207, ISO/IEC 9126, ISO/IEC 12119, ISO/IEC 14598, ISO/IEC TR 15271, ISO/IEC 15504, ISO/IEC TR 15846, ISO/IEC 15939, ISO/IEC TR 16326, ISO 10007, ISO/IEC 6592, ISO/IEC 19761, ISO/IEC 20926, ISO/IEC 20968, ISO/IEC 14764, ISO/IEC 15026, ISO/IEC 15910, ISO/IEC 14102 규격들과 상호 연관되도록 구성하였다.

ISO 9001:2004 요구사항별 관련 소프트웨어 규격간의 유용성 관계를 표 2-4에 표시한다.

소프트웨어 개발 및 유지보수에 있어 형상관리(Configuration Management)가 많은 주목을 받고 있다. 형상관리는 1960년대에 등장하여 1970년대에 미 국방규격(Military Standard)의 일부로 규격화되었으며 이후 지속적으로 발전하다가 1990년대 들어 컴퓨팅 환경이 복잡해지고 품질기준이 엄격해지면서 본격적으로 적용되어 1998년 11월에 ISO/IEC JTC1 SC7에서 ISO/IEC TR 15846:1998(소프트웨어 프로세스에서의 형상관리)이 제정되었다.

[표 2-4] ISO/IEC JTC1/SC7, ISO/TC 176 관련 규격과 ISO 9001:2000간의 이행 관련 유용성

ISO 9001:2000	ISO/IEC 12207	ISO/IEC 9126	ISO/IEC 12119	ISO/IEC 14598	ISO/IEC TR 15271	ISO/IEC 15504	ISO/IEC TR 15846	ISO/IEC 15939	ISO/IEC TR 16326	ISO 10007	ISO/IEC 6592	ISO/IEC 19761, ISO/IEC 20926 & ISO/IEC 20968	ISO/IEC 14764	ISO/IEC 15026	ISO/IEC 15910	ISO/IEC 14102
4 품질경영시스템																
4.1 일반 요구사항	X				Annex C											
4.2 문서화 요구사항	6.1, F.2.1															
5 경영책임																
5.1 경영의지																
5.2 고객중심																
5.3 품질방침																
5.4 기 획																
5.4.1 품질목표						X										
5.4.2 품질경영시스템 기획																
5.5 책임, 권한 및 의사소통																
5.6 경영검토																
6 자원관리																
6.1 자원의 확보																
6.2 인적자원																
6.2.1 일반사항	F3.3.1, F3.32															

ISO 9001:2000	ISO/IEC 12207	ISO/IEC 9126	ISO/IEC 12119	ISO/IEC 14598	ISO/IEC TR 15271	ISO/IEC 15504	ISO/IEC TR 15846	ISO/IEC 15939	ISO/IEC TR 16326	ISO 10007	ISO/IEC 6592	ISO/IEC 19761, ISO/IEC 20926 & ISO/IEC 20968	ISO/IEC 14764	ISO/IEC 15026	ISO/IEC 15910	ISO/IEC 14102
6.2.2 능력, 인식 및 교육훈련																
6.3 기반구조	7.2, F.3.2			Pt 2, Pt 3												X
6.4 업무환경																
7 제품 실현																
7.1 제품 실현의 기획	5.2.4, 5.3.1, 6.1 to 6.8, F.2	Pt 1		Pt 2			6.2		6.2.2							
7.2 고객 관련 프로세스																
7.2.1 제품 관련 요구사항의 결정	5.3.2 to 5.3.4, F.1.3.1,2,4	Pt 1	X											X		
7.2.2 제품 관련 요구사항의 검토	5.2.1, 5.2.6, 6.4.2.1, 6.6, F.3.1.5															
7.2.3 고객과 의사소통	5.2.5.6, 5.2.6, 5.2.7, 6.6, F.1.4.2												6.6.1, 7.3.3, 8.2, 8.2.3			
7.3 설계 및 개발	F.1.3.4, F.1.3.5		X								X	X			X	
7.3.1 설계 및 개발 기획	5.2.4, 5.3.1								6.2.2							
7.3.2 설계 및 개발 입력		Pt 1														
7.3.3 설계 및 개발 출력	5.3.5 to 5.3.7															
7.3.4 설계 및 개발 검토	5.3.4.2, 5.3.5.6, 5.3.6.7, 6.6.3, F.2.6				Annex A											

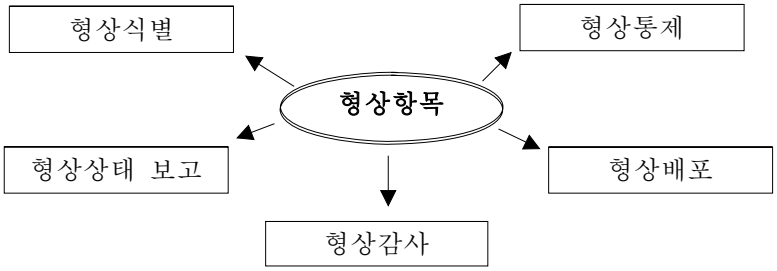
ISO 9001:2000	ISO/IEC 12207	ISO/IEC 9126	ISO/IEC 12119	ISO/IEC 14598	ISO/IEC TR 15271	ISO/IEC 15504	ISO/IEC TR 15846	ISO/IEC 15939	ISO/IEC TR 16326	ISO 10007	ISO/IEC 6592	ISO/IEC 19761, ISO/IEC 20926 & ISO/IEC 20968	ISO/IEC 14764	ISO/IEC 15026	ISO/IEC 15910	ISO/IEC 14102
7.3.5 설계 및 개발 검증	5.3, 6.5, F.1.3, F.2.4															
7.3.6 설계 및 개발 유효성 확인	5.3, 6.5, F.1.3, F.2.5			Pt 3, Pt 5												
7.3.7 설계 및 개발 변경 관리	5.5.2, 5.5.3, 6.1, 6.2, F.2.1, F.2.2															
7.4 구매		Pt 1		Pt 4								X				
7.4.1 구매 프로세스	5.1, F.1.1					Pt 3										
7.4.2 구매정보	5.1.2, F.1.1.1															
7.4.3 구매 제품의 검증	5.1.5, F.1.1.4															
7.5 생산 및 서비스 확보		Pt 1					X						X		X	
7.5.1 생산 및 서비스 확보 관리	5.3.12, 5.4.4, 5.5, 6.3.3, 6.8, F.1.3.11, F.1.4.2, F.1.5, F.2.8															
7.5.2 생산 및 서비스 확보 프로세스의 타당성 확인																
7.5.3 식별 및 추적성	6, F.2.2						7 to 12			X						
7.5.4 고객자산																
7.5.5 제품의 보존																
7.6 모니터링 및 측정장치 관리																

ISO 9001:2000	ISO/IEC 12207	ISO/IEC 9126	ISO/IEC 12119	ISO/IEC 14598	ISO/IEC TR 15271	ISO/IEC 15504	ISO/IEC TR 15846	ISO/IEC 15939	ISO/IEC TR 16326	ISO 10007	ISO/IEC 6592	ISO/IEC 19761, ISO/IEC 20926 & ISO/IEC 20968	ISO/IEC 14764	ISO/IEC 15026	ISO/IEC 15910	ISO/IEC 14102
8 측정, 분석 및 개선																
8.1 일반사항	7, F.3.3	Pt 2, Pt 3		Pt 2		Pt 1		5								
8.2 모니터링 및 측정																
8.2.1 고객만족		Pt 4														
8.2.2 내부감사	6.3, 6.7, F.2.3, F.2.7															
8.2.3 프로세스 모니터링 및 측정	7.3.2, 7.3.3, F.3.3.2					Pt 1, Pt 2		5								
8.2.4 제품 모니터링 및 측정	5.3, F.1.3	Pt 1		Pt 3, Pt 5												
8.3 부적합 제품의 관리	6.2, 6.8, F.2.2, F.2.8		X				X									
8.4 데이터의 분석								5.4				X				
8.5 개선																
8.5.1 지속적 개선	7.3, F.3.3					X										
8.5.2 시정조치	6.8, F.2.8															
8.5.3 예방조치	7.3.2, F.3.3.2					Pt 2										

형상관리란 불안정한 구성요소를 가진 시스템 즉, 변경이 일어나는 형태에서의 관리 기술을 의미하며, 형상 관리에 대해서 다수의 정의가 존재한다. IEEE Standard에서는 "형상 항목을 식별하여 기능적 물리적 특성을 문서화하고, 그 특성에 대한 변경을 제어하고, 변경 처리 상태를 기록·보고하고, 명시된 요구사항에 부합 여부를 확인하는 일련 사항에 대하여 기술 행정적 지침과 사후 관리를 적용하는 원칙(IEEE Standard Glossary of Software Engineering Terminology 1991)"로 정의하고 있다. ISO/IEC 12207에서는 "형상 항목(Configuration Item)은 특정의 기준점에서 고유하게 식별할 수 있는 실체"로 "버전(Version)은 형상 항목의 식별된 인스턴스(instance)로 규정하고 있다. 보편적으로 적용되는 또 다른 정의는 "전체 소프트웨어 공학 과정에 적용되는 '보호(Umbrella)' 활동이다. 변경은 언제라도 일어날 수 있기 때문에, 소프트웨어 형상관리(SCM, Software Configuration Management) 활동은 ①변경을 알아내기 위해 ②변경을 제어하기 위해 ③변경이 적절히 수행되고 있는 것을 확인하기 위해 ④변경에 관심을 가지고 있는 사람들에게 이것을 통보하는 것[23]"이다.

이와 같은 정의에서 파악할 수 있듯이 형상 관리는 크게 5가지로 구분할 수 있다.

- 형상 식별(configuration Identification): 형상 관리의 대상이 무엇인지 식별; 형상관리의 대상이 되는 형상 항목의 개체를 기록하고, 기준선(Base line)으로 설정하는 활동
- 형상통제(Configuration Control): 형상항목에 대한 변경사항을 체계적으로 관리하는 활동
- 형상 감사(configuration Audit): 형상관리 활동이 제대로 수행되는지를 조사하는 활동
- 상태 보고(status Reporting): 변경된 형상 항목을 기록하고, 보고하는 활동
- 형상배포(Configuration Release): 형상항목을 관련자 또는 고객에게 통보하는 활동



[그림 2-13] 형상관리 활동

ISO/IEC TR 15846 규격에서의 소프트웨어 형상관리(SCM)는 ISO/IEC 12207(소프트웨어 수명주기 프로세스)와 연계하여 소프트웨어 제품 수명주기에서의 CM 프로세스를 지원한다. 이 규격에서 요구하는 SCM는 소프트 제품들을 포함하여 컴퓨터에 저장될 수 있는 어떠한 정보도 제어가 가능하고, 또한 다른 장소에 저장되는 중요 소프트웨어

23) Roger's S Pressman, Software Engineering; :A Practitioner's Approach, 3rd Edition(1995)

항목에 대한 목록과 기록들을 관리할 수도 있으며, 인도 가능한 소프트 제품들을 만들거나, 유지하거나, 저장(archive)하거나, 복원하는 소프트웨어 환경에서 도구(tools)로서 사용되는 소프트웨어 제품은 어떠한 소프트웨어 타입일지라도 SCM에 의한 관리가 가능하도록 규정되어 있다. 다만 이 규격에서 SCM 프로세스에서의 활동과 작업들의 실행 방법을 지정하고 있는 것은 아니며, 운영, 유지보수 및 개발 프로세스의 전반에 걸쳐 지속성을 제공하기 위한 의도로 제정된 것이다.

ISO/IEC TR 15846의 SCM 요구사항은 운영, 유지보수 및 개발 프로세스 내에서 가시성(visibility)과 가측성(accountability)의 개선에 초점을 두고 있다. SCM 요구사항은 소프트웨어 제품 공급자에 작업을 주문하는 획득자(acquirer), 소프트웨어 제품의 인도에 책임을 지는 공급자 및 작업을 수행하는 외주업체 또는 소프트웨어 전문가로 구분될 수 있는 최소 3개의 연결고리에서 기인한 것으로, 에스크로(escrow)로 제3자 보관소 사용에 대하여 획득자와 공급자가 동의하는 경우에는 4개의 연결고리 관계가 존재할 수도 있다. 이러한 연결고리 하에서 SCM의 운영에 따른 이익은 다음과 같다.

- 수명주기 프로세스의 지원을 위한, 적절한 문서화 및 전자문서, 코드(Code), 인터페이스, 데이터베이스 등을 식별하고 수집하도록 하는 반복적인 체계(scheme)를 지원,
- 요구사항, 기준, 방침과 지침, 조직 및 경영철학에 부합되는 개발, 유지보수 또는 운영 가동 방법론을 지원,
- 기준선(baselines)의 상태, 변경, 릴리즈, 버전, 파일보관(archives) 등과 관련하여 관리 및 제품 정보를 생산,
- 관리될 중요 개개의 항목들(items) 수준에 따라 SCIs를 순차적(recursively)으로 정의,
- 그 상태 및 관련 정보와 함께 SCIs에 저장되는 라이브러리(libraries)를 제어,
- 형상의 완전성(integrity)을 보증하기 위한 ISO/IEC 12207 프로세스를 인용,
- 형상의 완전성 보증(예; 트래커(tracker), SCM 라이브러리 가디언, 릴리즈 빌더)과 그 도구들이 적용(예; 운영체계)되도록, 소프트웨어 제품의 개발 및 검증에 사용되는 소프트웨어 도구를 포함하여, 유용한 수명 전반에 걸쳐 소프트웨어 제품의 구성(configure) 및 재구성이 가능하도록 소프트웨어 환경을 제어,
- 소프트웨어 제품의 형상 및 개개의 SCIs에서의 변종/버그에 대한 정보를 저장 및 검색,
- 면허(licences)나 저작권과 같은 지적 재산권 고려를 위하여 소유권(ownership)을 기록

ISO/IEC TR 15846은 ISO/IEC 12207(소프트웨어 수명주기 프로세스) 및 ISO 10007(형상관리를 위한 품질경영 지침)과 일관성을 유지한다. 표 2-5에 이 규격과 ISO/IEC 12207 및 ISO 10007과의 관계를 표시한다.

ISO/IEC TR 15846		ISO/IEC 12207:1995		ISO 10007:1995	
6	SCM 프로세스 구현	6.2.1	프로세스 구현	5	형상관리 프로세스
				6	형상관리 조직
				6.1	개요
6.1	초기 및 범위설정	5.1.1	초기화	6.2	형상관리 구조
6.2	기획	5.1.2	제안에 대한 요청 준비	7.7	형상관리 계획
				Annex A	(권고) 형상관리 계획 구조 및 내용
		5.1.3	계약준비 및 업데이트		
		5.1.4	공급자 모니터링		
		5.2.4 6.2.1	기획 프로세스 구현		
		7.1.2.1 e)			
		7.2	기반구조 프로세스		
		7.3	개선 프로세스		
6.3	집행 통제		6.1.3.2, 6.1.4.1, 6.4.2.7 c)		
		6.8	문제해결 프로세스		
		7.1.3			
		7.4	훈련 프로세스		
6.4	SCM 프로세스 검토 및 평가	7.3	개선 프로세스		
6.5	종결	7.1.5	종결		
7	소프트웨어 형상식별(SCIs)	6.2.2	형상식별	5.2	형상식별
				7.2	형상식별 절차
7.1	SCIs의 식별	5.3.3.1			
7.2	형상기준선 식별				
7.3	소프트웨어 라이브러리 식별				
7.4	촉진상태				
8	소프트웨어 형상관리	5.3.1.2 b)		5.3	형상관리
		6.2.3	형상관리	7.4	형상관리 절차
8.1	제안 변경				

ISO/IEC TR 15846		ISO/IEC 12207:1995		ISO 10007:1995	
8.2	제안된 변경영향 평가				
8.3	변경의 이행				
8.4	처리 의사소통			7.3	
8.5	변경종결				
9	소프트웨어 형상상태 설명	6.2.4	형상상태 설명	5.4	형상상태 설명
				7.5	형상상태 설명 절차
9.1	식별 기록				
9.2	추적성 변경				
9.3	보고상태 설명 기록				
10	소프트웨어 형상평가	5.2.6	검토 및 평가	5.5	형상감사
		5.3.4.3		7.6	형상감사 절차
		5.3.9.5 b)			
		5.3.11.4b)		8	형상관리시스템 감사
		6.2.5	형상 평가		
		6.4	검증 프로세스		
		6.5	밸리데이션 프로세스		
		6.6.1.1			
		6.6.3.1 c)			
		6.7.1.1			
		7.1.5	종결		
		7.3	개선 프로세스		
11	소프트웨어 릴리즈 관리 및 인도	5.2.7	승인 및 완료		
		5.4.2.1			
		6.2.6	릴리즈 관리 및 인도		
11.1	취급	5.3.4.1 e)			
11.2	저장	5.5.6.1 b)			
11.3	복제				
11.4	포장	5.4.4.3 5.5.5.3			
11.5	인도				

[표 2-5] ISO/IEC TR 15846, ISO/IEC 12207, ISO 10007과의 관계

2. IEEE 규격 분석

컴퓨터 및 전자분야 전문가들의 참여로 구성된 IEEE의 규격들은 지나칠 정도로 상세하게 규정되어 있다. IEEE[24](Institute of Electrical and Electronics Engineers, Inc.)기관의 성격 및 많은 IEEE 규격들이 이미 미국의 ANSI 규격으로 제정되어 있어 이들 규격의 분석은 미국 동향으로 분류할 수도 있으나 본 연구에서는 국제기준에 준하는 것으로 가늠한다.

본 연구와 관련된 IEEE의 주요 규격들은 표 2-6와 같다.

IEEE Std 730은 소프트웨어의 품질보증계획서(Software Quality Assurance Plans)에 대한 규격으로, 소프트웨어의 개발, 검증 및 확인, 유지보수와 관련된 품질보증계획서(SQAP)의 준비와 내용에 대한 일관되고 수용 가능한 요구사항을 언급하고 있다. 이 규격에서 SQAP의 최소 문서화는 소프트웨어 요구사항 명세서(SRS), 소프트웨어 설계기술서(SDD), 소프트웨어검증 및 밸리데이션 계획서(SVVP), 소프트웨어 검증 및 밸리데이션 보고서(SVVR), 소프트웨어 형상관리 계획서(SCMP) 및 사용자 문서화 등을 요구하고 있다. SRS는 소프트웨어가 무엇을 할 것 인지에 대한 서술로 외부 인터페이스와 소프트웨어의 기본적 요구사항들(기능, 수행능력, 설계 제약, 속성) 각각에 대하여 명확하게 기술하고, 이 요구사항들을 시험·검사, 분석 등으로 객관적인 검증 및 밸리데이션이 가능하도록 정의하고 있다. SDD는 소프트웨어가 SRS의 요구사항을 만족시키기 위하여 어떻게 구성되어야 하는지 설명하는 문서이다. SDD는 데이터베이스와 내부 인터페이스를 포함하여 소프트웨어 설계의 구성요소와 하위 구성요소들에 대해서도 기술하여야 한다. 사용자 문서화에는 매뉴얼, 지침서와 같이 소프트웨어의 성공적인 실행을 위해 요구되는 자료와 제어입력, 입력 순서, 선택사항, 프로그램 한계 및 다른 조치나 항목을 명시하며, 모든 에러 메시지는 식별되고 시정조치를 설명하도록 규정되어 있다. 사용자와 직접적인 상호작용이 없는 내장형(embedded) 소프트웨어의 경우는 사용자 문서가 필요하지 않다. 이외에도 제조업체에서는 소프트웨어 개발계획서, 소프트웨어 유지보수 매뉴얼 등을 문서화할 필요가 있다.

24) 국제전기전자기술자협회, 1884년 전기전자 분야의 엔지니어 훈련을 위해 설립되었으며, 1963년에 AIEE(American Institute of Electrical Engineers, formed in 1884) 및 IRE(Institute of Radio Engineers, formed in 1912)가 통합되었다. 전기·전자, 컴퓨터, 전산 분야 등의 150개국 약 36만명의 엔지니어, 과학자 및 학생으로 구성된 단체이며 컴퓨터 및 전자분야에 대한 규격 제정, 세미나 및 회의 개최, 교육 등의 활동을 수행.

IEEE std	규 격 명
730-1998	Standard for Software Quality Assurance Plans.
730.1-1995	Guide for Software Quality Assurance Planning.
828-1998	Standard for Software Configuration Management Plans.
829-1983	Standard for Software Test Documentation
830-1993	Recommended Practice for Software Requirements Specifications
982.1-1988	Standard Dictionary of Measures to Produce Reliable Software.
982.2-1988	Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software
983-1986	Guide for Software Quality Assurance Planning.
1008-1987	Standard for Software Unit Testing.
1012-1998	Standard for Software Verification and Validation.
1012a-1998	Standard for Software Verification and Validation: Content Map to IEEE/EIA12207.1-1997
1016-1998	Recommended Practice for Software Design Descriptions.
1028-1988	Standard for Software Reviews and Audits.
1042-1987	Guide to Software Configuration Management.
1058a-1998	Standard for Software Project Management Plans: Content Map to IEEE/EIA 12207.1-1997
1063-1987	Standard for Software User Documentation.
1074-1997	Standard for Developing Software life cycle Processes.
1228:1994	Standard for Software Safety Plans
1233-1998	Guide for Developing System Requirements Specifications.

[표 2-6] 주요 IEEE 규격 목록

IEEE Std 828은 소프트웨어 형상관리 계획에 대한 규격으로 IEEE Std 730에서의 소프트웨어 형상관리 계획서(SCMP)와 일관된 사용을 가능하게 한다. 이와 관련하여 IEEE Std 1042(소프트웨어의 형상관리)는 미국 국가규격(ANSI)으로 이미 채택되었다. IEEE Std 828에서는 '형상(Configuration)'에 대하여 다음과 같이 정의[25]하고 있다.

(1) 컴퓨터 시스템의 배열 또는 본질, 수 그리고 기능 단위들의 주요한 특성들에 의해서 정의된 네트워크, 더 상세히 말하면 형상에는 하드웨어 형상과 소프트웨어 형상이 있다.

25) IEEE Std 828-1998의 3.1.4 참조

(2) 시스템 또는 시스템 성분의 특별한 설명을 정의하는 구현, 설계, 그리고 요구사항.
(3) 기술적인 문서화작업에 나타나고 생산에 수행되는 하드웨어와 소프트웨어의 기능적, 물리적 특성.

표 2-7은 IEEE Std 828(소프트웨어 형상관리 계획)과 IEEE Std 1042(소프트웨어의 형상관리)와의 상호관계를 표시한 것이다.

IEEE std 828-1998 해당 절	IEEE std 1042-1987 해당 절
1. 서론	1. 서론
—	2. 소프트웨어 관리에서 SCM원칙
4. SCM계획	3. 소프트웨어 형상관리 계획
4.1 서론	3.1 서론
4.2 SCM조직 및 책임	3.2 관리
4.3 SCM활동	3.3 SCM활동
4.3.1 형상 식별	3.3.1 형상 식별
4.3.2 형상 통제	3.3.2 형상 관리
4.3.3 형상 상태의 기록	3.3.3 형상 상태의 기록
4.3.4 형상 감사 및 검토	3.3.4 감사 및 검토
4.3.5 인터페이스 제어	3.2.3 인터페이스 제어
4.3.6 외주업체/판매자 관리	3.5 공급자 관리
4.4 SCM일정	3.2.4 SCM계획의 구현
4.5 SCM자원	3.4 도구, 기법 및 방법론
4.6 SCM계획의 유지보수	2.5 SCM의 계획
5. SCM 계획 적용	2.5 SCM의 계획
6. 규격에 대한 적합성	2.5 SCM의 계획

[표 2-7] IEEE Std 828과 IEEE Std 1042의 상호관계

IEEE 1012는 소프트웨어 검증 및 밸리데이션(V&V, Verification and Validation) 규격으로 ISO/IEC 12207(소프트웨어 수명주기 프로세스), IEEE 1074(수명주기 프로세스 개발), IEEE/ EIA 12207.0(수명주기 프로세스) 등의 규격과 일관성을 갖도록 V&V 프로세스에 대한 규격이다. 이러한 규격들은 모두 소프트웨어 검증 및 소프트웨어 밸리데이션 계획에 관한 요구사항을 규정하고, 이 요구사항들 간의 관계를 설명함으로써 모든 규격을 만족하는 소프트웨어를 생산하는 제조업체들이 사용할 수 있도록 하는 것이다.

IEEE STD 1012는 소프트웨어 검증 및 밸리데이션 계획(SVVP)에 대한 개념을 정의하고 특정 활동과 관련 작업에서의 검증 및 밸리데이션 활동을 규정하고 있다.
이 규격에서는 획득, 공급, 개발, 작동, 관리 프로세스 등을 포함하는 모든 소프트웨어 수명주기 프로세스를 뒷받침하는 V&V 프로세스 활동과 작업의 일반 골격(frames)을 구축하고, V&V 작업, 필요한 인풋과 아웃풋에 관한 개념을 정의하며, 소프트웨어 완전성 수준에 상응하는 V&V 최소 작업을 표 2-8에서와 같이 4단계로 구분하여 소프트웨어 V&V 계획(SVVP)의 개념을 정의한다.

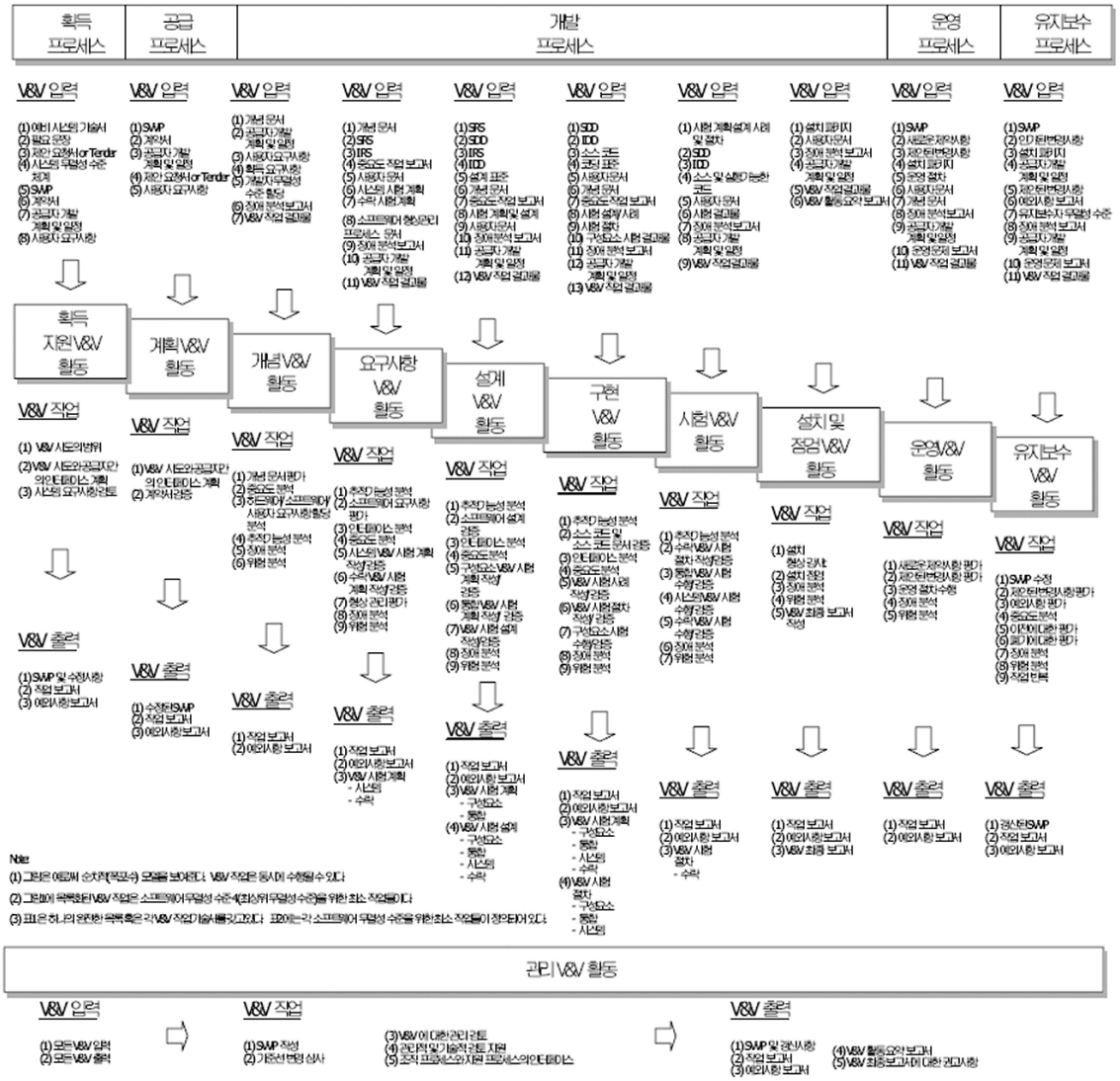
중요도	설명	수준
상(High)	선택된 기능은 관련 시스템의 주요 성능에 영향을 미친다.	4
중상(Major)	선택된 기능은 주요 시스템 성능에 영향을 미친다.	3
중(Moderate)	선택 기능은 시스템 성능에 영향을 미치지만 효과적인 전략을 이용하여 성능 상실을 보완한다.	2
하(Low)	선택된 기능은 시스템 성능에 현저한 영향을 미치지만 그 기능이 요구사항에 따라 이행되지 않으면 사용자에게 불편할 뿐이다.	1

[표 2-8] 소프트웨어 완전성 수준(Software integrity levels) 구분

이 소프트웨어 완전성 수준구분은 의료기기 위험경영시스템에서의 심각도 분류와 유사한 개념으로 다양한 소프트웨어 특성을 설명하기 위하여 소프트웨어 완전성(Software integrity)을 4단계의 수준으로 나누어 정의하고, 각 수준별로 필요한 V&V 최소한의 활동에 관하여 언급한 것이다.

소프트웨어는 의도한 목적(용도)와 주요하거나 또는 주요하지 않은 용도에 대한 시스템의 적용에 기초한 다른 중요도(criticality)를 갖고 있다. 일부 소프트웨어 시스템은 생명 유지용 시스템에 중대한 영향을 미치는 반면 다른 소프트웨어 시스템은 범용적, 독립적인 연구용 툴이다. 소프트웨어 중요도는 시스템 본래의 용도와 적용을 말해준다. 이 규격에서는 소프트웨어 완전성 수준을 이용하여 소프트웨어 중요도를 제한하고, 완전성 수준의 허용 한계 내에서 위험(Risk)을 관리하는데 필요한 소프트웨어 중요도 범위를 설정하였다. 소프트웨어 속성으로는 안전성, 복잡성, 성능, 신뢰도 또는 다른 특성이 포함된다. 중요도가 상(High)인 소프트웨어는 일반적으로 더 많고 더 정확한 V&V 작업의 적용을 필요로 한다. 또한 적용된 소프트웨어 완전성 수준 리스트 및 관련 V&V 최소 작업의 지도(Map)를 SVVP에 첨부하여 어떤 활동들을 수행할 것인지 나타내도록 한다. 다만, 규격에서 규정한 소프트웨어 완전성 수준 리스트의 이용을 의무화하지 않고 있다.

그림 2-14는 이러한 소프트웨어 V&V 활동에 대한 개요를 표시한 것이다.



[그림 2-14] 소프트웨어 V&V 개요

소프트웨어 V&V 목적은 소프트웨어 수명주기 동안 제조업체에서 소프트웨어 품질을 구축하는데 있다. 소프트웨어 V&V 프로세스에서 특정 범주의 개발 제품이 관련 요구사항을 따르고 있는지, 소프트웨어가 본래의 용도와 사용자 요구를 만족시키는지를 평가한다. 평가과정에서 소프트웨어 제품과 프로세스의 심사, 분석, 평가, 검토, 조사 및 시험 내용을 고려한다. 소프트웨어 V&V는 소프트웨어 개발 결정 시점이 아닌, 소프트웨어 개발 시점에서 동시에 수행한다.

V&V 작업에 적용되는 강도 및 정확도는 소프트웨어 완전성 수준에 따라 다르다. 여기서 강도란 모든 정상 및 비정상 시스템 작동 조건에서 보다 넓은 분석 범주를 포함한 것이며, 정확도란 더 형식적인 기술과 리코딩 절차를 포함한다. 정확성, 일관성, 완전성, 읽기 쉬움 및 시험 가능성에 관한 최소한의 기준을 포함하여 V&V 각 작업에 관한 기준에는 위해요인 분석(Hazard Analysis) 등의 평가를 포함한다.

V&V 프로세스를 통하여 수명주기의 소프트웨어 제품과 모든 프로세스를 객관적으로 평가하고 이를 통하여 소프트웨어 요구사항과 시스템 요구사항의 정확성, 완전성, 일관성, 시험가능 여부를 입증하여 다음과 같은 목적을 달성한다.

- 소프트웨어 결함을 초기에 검출하고 교정할 수 있도록 한다.
- 프로세스와 제품 리스크에 관한 경영적 식견을 높이도록 한다.
- 프로그램 성능, 스케줄, 예산 요구사항을 따르도록 수명주기 프로세스를 뒷받침한다.

검증 프로세스를 통해 소프트웨어와 관련 제품이 다음과 같음을 증명한다.

- 각 수명주기 프로세스(획득, 공급, 개발, 작동, 관리)에서 모든 수명주기 활동에 관한 요구사항(예: 정확성, 완전성, 일관성 등)을 준수한다.
- 수명주기 프로세스 중 규격, 사례, 및 협정을 만족시킨다.
- 각 수명주기 활동의 완료를 평가하고 다른 수명주기 활동을 개시하는 근거를 구축한다.

밸리데이션 프로세스를 통해 관련 소프트웨어가 소프트웨어에 할당된 시스템 요구사항을 만족시키고 문제를 바르게 해결하는지의 증거에 대해서는 소프트웨어 검증 및 밸리데이션 보고서(SVVR)에 기록한다.

표 2-9에 소프트웨어 검증 및 밸리데이션 계획의 예제를 명시한다.

소프트웨어 V&V 플랜 개요 (견본)
1. 목적 2. 참고 문헌 3. 개념 정의 4. V&V 개요 4.1 조직 4.2 마스터 스케줄 4.3 소프트웨어 무결성 수준 일람표 4.4 자원 요약 4.5 의무 4.6 툴, 기술, 그리고 방법 5. V&V 프로세스 5.1 프로세스: 경영 5.1.1 활동: V&V의 경영 5.2 프로세스: 취득 5.2.1 활동: 취득 지원 V&V 5.3 프로세스: 공급 5.3.1 활동: V&V 계획 5.4 프로세스: 개발 5.4.1 활동: 개념 V&V 5.4.2 활동: 요구사항 V&V 5.4.3 활동: 설계 V&V 5.4.4 활동: 이행 V&V 5.4.5 활동: 테스트 V&V 5.4.6 활동: 설치 및 체크아웃 V&V 5.5 프로세스: 작동 5.5.1 활동: 작동 V&V 5.6 프로세스: 관리 5.6.1 활동: 관리 V&V 6. V&V 보고에 관한 요구사항 7. V&V 관리에 관한 요구사항 7.1 예외 해결과 보고 7.2 작업 반복 정책 7.3 일탈 정책 7.4 관리 절차 7.5 규격, 사례, 그리고 협정 8. V&V 기록에 관한 요구사항

[표 2-9] SVVP 개요의 견본

1995년에 IEEE의 SESC[26](Software Engineering Standards Committee)는 ISO/IEC 12207(소프트웨어 수명주기 프로세스) 국제규격 평가하고 그 규격이 IEEE 내의 수명주기 과정에 대한 근간으로 채택되도록 결정하였다. ISO/IEC 12207을 IEEE에서 채택한 것이 IEEE/EIA 12207.0 규격이다. 이 규격은 ISO/IEC 12207 규격 요구사항을 포함하여 개선시킨 적합성 접근법, 수명 주기 프로세스의 구현성의 고려, 수명주기 데이터

26) 컴퓨터 학회 소트프웨어 공학 기준규격 위원회

등과 같은 사항을 추가하여 IEEE/EIA 12207.1로 개정되었다.

IEEE/EIA 12207.1은 지침 규격이지만 구체적인 적합성 요구사항을 갖는 기준 규격으로서 적용에 필요한 내용도 포함하고 있다. 표 2-10은 이 규격과 ISO/IEC 12207의 관계를 나타낸 것이다.

IEEE STD 1228 규격은 소프트웨어 안전성 계획서에 관한 규격으로, 안전이 중요한 소프트웨어의 안전성 개선을 위한 과정 및 활동을 처리하기 위하여 소프트웨어 안전성 계획서의 내용에 대한 최소 허용 가능한 요구사항을 규정하고 있다. 이 규격의 취지는 소프트웨어는 시스템의 일부분이며, 시스템의 다른 부분으로 컴퓨터 하드웨어, 기타 장치(기계적, 전기적, 화학적 또는 방사선 관련 장치를 포함) 및 사람들이 포함되는데, 소프트웨어 자체만으로는 안전성 문제가 있을 수 없고, 보다 더 큰 범위의 시스템이란 맥락 하에서만 문제가 된다. 따라서 소프트웨어 안전성은 시스템이란 큰 범위에서 시작하여야 하며 소프트웨어 안전성은 반드시 관련된 하드웨어, 환경 및 조작자라는 맥락에서 고려할 것이다.

이 기준규격은 예를 들면, 고장이 나게 되면 생명 손실, 중상을 일으키거나 또는 부정적 사회 충격을 확산시킬 수 있는 소프트웨어 제품과 같이 안전성이 중요시되는 소프트웨어의 개발, 구매, 보전 및 폐기에 사용되는 계획서에 적용한다. 이 기준규격은 계획서가 시스템 안전성 프로그램의 맥락 이내에서 작성될 것을 요구한다.

1012 V&V 활동들	IEEE Std 12207 소프트웨어 수명주기	
	프로세스	활동들
취득 지원 V&V	취득	- 개시 - 제안(텐더) 요청서 준비 - 계약서 작성 업데이트 - 공급자 모니터링 - 승인과 완료
계획 V&V	공급	- 개시 - 답변 준비 - 계약 - 계획 - 실행과 관리 - 검토와 평가 - 릴리즈와 완료
개념 V&V	개발	- 프로세스 이행 - 시스템 요구사항 분석 - 시스템 구조 설계
요구사항 V&V	개발	- 소프트웨어 요구사항 분석
설계 V&V	개발	- 소프트웨어 구조 설계 - 소프트웨어 세부 설계
이행 V&V	개발	- 소프트웨어 코드화와 테스트
테스트 V&V	개발	- 소프트웨어 통합 - 소프트웨어 승인 지원
설치와 체크아웃 V&V	개발	- 소프트웨어 설치 - 소프트웨어 승인 지원
작동 V&V	작동	- 프로세스 이행 - 작동 테스트 - 시스템 작동 - 사용자 지원
관리 V&V	관리	- 프로세스 이행 - 문제와 변경 내용 분석 - 변경 이행 - 관리 검토/승인 - 이동 - 소프트웨어 요구사항
V&V의 경영	모든 프로세스	- 모든 활동들

[표 2-10] ISO/IEC 12207과 IEEE Std 1012 V&V 활동들과의 관계

[별첨 2]

ISO/IEC TR 15846:1998
정보기술 - 소프트웨어 수명주기 프로세스 - 형상관리
(Medical electrical equipment - Part 1-6: General requirement for safety-Collateral Standard: Usability)

1. 적용범위

이 기술 보고서에는 개발, 유지보수 및 운영을 위한 컴퓨터소프트웨어 형상관리의 성능에 대한 요구조건이 규정되어 있다. 이 기술 보고서는 ISO/IEC 12207의 형상관리(CM) 프로세스(이하 소프트웨어 형상관리(SCM) 프로세스라 칭함)를 기초로 작성되었다.

이 기술 보고서는 다음에 적용된다.

- 모든 형태의 소프트웨어
- 소프트웨어 제품 수명주기 및 개별 개발의 경우 그러한 수명 주기 중 유지보수와 운영 프로젝트, 하청업체나 벤더가 공급하는 소프트웨어
- 소프트웨어 제품의 공급업체 및 구매자

이 기술 보고서는 당사자가 둘인 상황에 적용되며, 두 당사자가 동일한 조직에 속할 경우 균등하게 적용된다. 상황은 비공식 합의에서 공식 계약에 이르기까지 다양할 수 있다. 이 기술 보고서는 자체 부과 임무로서 단일 당사자가 사용하거나 기성품에 적용할 수 있다.

1.1 이 기술 보고서의 조정

일부 소프트웨어나 소프트웨어 수명주기에는 다른 해당 표준이나 계약에 규정한 요구조건이 적용될 수 있으며, 현지 실천요강이 채택될 수도 있다. SCM 프로세스는 요구조건을 추가로 적용함으로써 조정할 수 있다.

또한 SCM 프로세스는 특정 요구조건을 적용할 필요가 없다고 판단할 경우 이 기술 보고서의 요구조건을 무시하도록 조정할 수도 있다. ISO/IEC 12207에 따른 이 기술 보고서는 이 기술 보고서에 제공된 매핑을사용하면 쉽게 조정할 수 있다.(부속문서 A 참조)

1.2 프로세스 역할

이 기술 보고서의 사용자는 구매자와 공급업체의 역할을 수행한다(그림 1 참조).

ISO/IEC 12207에 규정한 유지보수와 개발 프로세스를 수행하는 소프트웨어 제품의 공급업체는 SCM의 구매자이다.

운영 프로세스의 경우 소프트웨어 제품의 인수 후 구매자는 최종 고객이나 소비자에 대한 SCM 공급업체의 역할을 수행한다.

SCM 프로세스(이하 "SCM 프로세스"라 칭함)의 공급업체는 하청계약 또는 벤더 제품 구매자의 역할을 수행한다.

2. 준수성

해당 없음

3. 규격상 참고문헌

아래의 규격상 참고문헌에는 이 문서내용을 통해 이 기술 보고서의 규정을 구성하는 규정이 수록된다. 일자가 명기된 참고문헌의 경우 그러한 발간물의 후속 수정이나 개정은 적용되지 않는다.

단, 이 기술 보고서에 기초한 합의에 대한 당사자는 아래에 표시한규격상 참고문헌의 최신 버전을 적용할 수 있는 가능성을 모색해야 한다. ISO와 IEC 구성원은 현재 유효한 기술 보고서의 등록부를 유지한다.

- ISO/IEC 12207:1995, 정보기술 – 소프트웨어 수명주기 프로세스
- ISO/IEC 2382-20:1990, 정보기술 – 용어 – 제20부: 시스템 개발
- ISO/IEC 2382-1:1993, 정보기술 – 용어, 제1부: 기본 용어

참조용 참고문헌은 부속문서 B에 수록되어 있다.

4. 정의

이 기술 보고서의 목적을위해 ISO/IEC 12207에 규정한 정의와 다음과 같은 정의가 적용된다.

4.1 승인 받은 수정

SCI에 대한 변경을 인가하는, 하나 이상으로 제안된 변경의 처리

참고) '제안된 변경'과 '승인된 수정' 사이에는 다양한 관계가 있을 수 있다. 제안된 변경에 의해 몇몇 SCI에 수정이 발생할 수 있다(코드와 시험 사례에만 해당되는 경우에도). 수정이 진행 중인 동안 동시에 또는 시간이 경과하면서 승인된 몇 가지 제안 변경에서 수정이 발생할 수 있다.

4.2 변경 권한

ISO 10007의 "형상 보드"

참고) 처리는 "변경/형상관리 보드"라고 그 동안 불렸던 지정 변경 권한에 의해 수행된다. 이 권한에 따라 제안된 변경을 승인해 승인된 수정으로 변환하거나 제안된 변경을 승인하지 않거나 결정을 유보할 수 있다.

4.3 제안된 변경

아이디어를 기록한 시점부터 지정 변경 권한에 의한 처리까지 요구되거나 권고되는 강화가 포함되는 변경 보고서

참고 1) 처리는 거부, 후속 분석을 위한 유예 또는 승인된다. 승인된 제안 변경은 허가 수정이 된다.

2) 제안된 변경과 허가된 수정 사이에는 일대일, 일 대 다자, 또는 다자간관계가 있을 수 있다.

4.4 소프트웨어 형상관리(SCM)

SCI의 완벽성과 정확성 확보를 위해 소프트웨어 수명주기 전반에 걸쳐 형상관리를 적용하는 프로세스(ISO 10007 참조)

4.5 소프트웨어 라이브러리

개발, 운영 및 유지보수 지원을 위해 제어된 방식으로 수집한SCI 집합체

4.6 소프트웨어 툴

소프트웨어 수명주기 임무를 위한 자동 지원을 제공하는 소프트웨어 제품

참고) 소프트웨어 툴에는 작성자의 지원 여부에 관계없이 벤더 소프트웨어 및 자체 개발 툴이 포함된다. 툴에는 운영체제가 실행하는 소프트웨어와 운영체제 자체가 포함된다. 또한 툴에는 매크로, 시험 스크립트 또는 빌드 설명서와 같은 해석한 프로그램이 포함된다.

5. 기호(약어 포함)

5.1 약어와 두문자

아래의 약어와 두문자가 이 기술보고서에 사용된다.

- CI : 형상 항목
- CM : 형상관리
- SCI : 소프트웨어 형상 항목
- SCM : 소프트웨어 형상관리

6. SCM 프로세스 구현

ISO/IEC 12207: 1995

6.2.1 프로세스 구현. 이 활동은 다음과 같은 업무로 구성된다.

6.2.1.1 형상관리 계획을 개발해야 한다. 계획에는 다음에 대한 설명을 포함해야 하다. 형상관리 활동, 그러한 활동의 수행을 위한 절차와 일정, 활동의 수행을 담당하는 조직, 그러한 조직과 소프트웨어 개발이나 유지보수를 담당하는 다른 조직과의 관계. 계획을 문서화하고 구현해야 한다.

참고) 계획은 시스템 형상관리 계획의 일부일 수도 있다.

SCM 프로세스는 운영, 유지보수 또는 개발 프로세스를 위한 소프트웨어 수명주기 전체 또는 일부에 적용해 구현된다.

6.1 개시 및 범위의 정의

6.1.1 SCM 프로세스에 대한 입력의 정의

SCM 프로세스는 SCM 요구조건을 입력으로서 확보하고, SCM 요구조건이 완벽하며 쉽게 이해할 수 있는 것인지 확인해야 한다. 이러한 SCM 요구조건은 다음과 같다.

a) SCM 프로세스의 일부가 되는 소프트웨어 제품

SCM 프로세스가 SCM 계획에 규정한 것처럼 수행되는지 여부의 입증이나 보증

b) SCM 프로세스의 소프트웨어 환경

소프트웨어 제품에 구매, 고객 공급, 하청계약에 따른 개발 또는 벤더 SCI가 있을 경우 SCM 프로세스는 외부 개발 SCI의 식별, 변경 제어, 상태 설명 및 형상평가("형상 심사"라고도 함)를 수행해야 한다.

6.1.2 SCM 프로세스에 대하 자원과 제약사항의 정의

SCM 프로세스는 SCM 활동이 다음의 정의에 의해 구현되는 기술적 및 관리적 측면의 조직 상황을 구축해야 한다.

a) 기준으로 지정되는 SCI의 영향을 받으며, SCM 활동에 참여하거나 그러한 활동을 담당하는 조직 단위
b) 그러한 조직 단위에 대한 SCM 역할과 책임
c) 조직 단위, 구매자 및 공급업체 사이의 관계

SCM 프로세스는 조직, 활동, 업무, 절차, 정보와 보고서 계획입안을 위한 형식 및 자원을규정하는 문서를 확립해 유지해야 한다.

SCM 프로세스는 SCM 절차, 표준, 용어 및 관련 문서에 대한 참조를 식별해야 한다.

6.1.3 책임과 권한의 할당

SCM 프로세스는 SCM에 필요한 자원을 계획하고 획득하며 채택해야 한다.

SCM 프로세스는 다음을 수행할 수 있는 권한과 능력이 있는 조직 단위에 SCM 활동을 할당해야 한다.

a) 기준의 확립
b) 기준의 변경에 대한 승인/비승인
c) 소프트웨어 제품의 릴리즈
d) SCM 요구조건과의 편차에 대한 승인/비승인

SCM 프로세스는 통합 연락처를 식별하고 지명해야 한다.
SCM 프로세스는 기준 변경의 승인을 취득하기 위한 기준을 결정해야 한다.
SCM 프로세스는 권한의 변경을 식별하고 권한의 범위를 할당해야 한다.

6.1.4 SCI 선정을 위한 기준

SCM 프로세스는 소프트웨어 제품에 기초해 SCI를 선정하기 위한 다음과 같은 기준을 확립해야 한다.

a) 필수
b) 소프트웨어 환경에 의한 사용
c) 유도를 위한 툴의 설명과 매개변수를 포함해 릴리즈의 유도에 사용

SCM 프로세스는 SCI의 성능 매개변수와 물리적 특성의 관리에 충분한 SCI 선정을 위한 기준을 정의해야 한다.

참고) SCM 프로세스는 너무 많은 SCI를 선정해 관리의 가시성을 훼손하고 비용을 증가시키지 않도록 해야 한다.

6.1.5 SCM 프로세스 결과의 정의

필요한 경우 SCM 프로세스는 다음의 결과의 전달을 계획해야 한다.

a) 소프트웨어 환경의 운영을 위한 정보
b) SCI 식별 계획
c) SCI의 재작성을 위한 툴과 소프트웨어 환경
d) SCI 버전 제어를 위한 계획
e) SCI 구조를 입증하는 문서
f) SCI 상태의 의미
g) SCI의 상태
h) SCI 상태의 완전성
I) SCI

6.2 계획입안

특정 소프트웨어 제품의 경우 SCM 프로세스는 SCM 구현에 영향을 주는 소프트웨어 수명주기 주요일정이나 현안(예를 들어 인터페이스 제어의 도입)에 대한 SCM 활동의 의존성을 계획해야 한다.

SCM 활동을 수행하거나 상호작용하는 조직의 대표는 SCM 계획입안 정보를 검토하고 승인해야 한다.

필요한 경우 SCM 프로세스는 SCM 프로세스를 전달하고 SCM 소프트웨어 제품을 개선해야 한다.

SCM 프로세스는 필요한 경우 SCM 활동의 중단을 계획해야 한다.

SCM 프로세스는 변경의 반영을 위해 SCM 계획입안 정보를 갱신해야 한다. SCM 프로세스는 변경된 SCM 업무를 수행하기 전 그러한 업무를 검토하고 해당 관계자의 승인을 받아야 한다.

아래의 정보를 SCM 계획에 포함하거나 참조해야 한다.

a) 계약의 식별
b) 지정한 소프트웨어 수명주기 프로세스에 대한 SCM 지원의 범위
c) 소프트웨어 제품이 인도되었는지 확인
d) 후속 유지보수에 필요하거나 위 c)에 식별한 항목의 완전성에 영향을 주는 다른 소프트웨어 제품의식별
e) 조직 정의 및 상호관계
f) 역할과 책임
g) 필요한 자원의 목록 및 자원이 필요한 시점
h) 하청업체 SCM 제어를 포함한활동의 인터페이스를 위한 하드웨어나 시스템 CM 및 절차와 SCM과의 관계
i) 형식, 일정 및 배포를 포함해 상태 보고의 절차
j) 변경 제안 및 SCI 변경과 개선을 위해 할당한 권한을 포함해 변경을제어하는 절 차
k) 유지해야 할 버전의 번호를 포함해 이전 버전 지원에 대한 정책
l) 다중 버전으로 개발 고객 지원
m) 기준 확인의 검토
n) 적용할 SCM 프로세스의 완전성 확인을 위한 심사
o) SCM 프로세스가 필요한 업무를 제공할 수 없어 SCM 업무의 원가, 일정 또는 성능에 영향을 주는 위험
p) 릴리즈 관리와 인도 절차
q) 인터페이스 제어

6.3 실행 제어

SCM 프로세스는 담당자가 SCM 계획에 규정한 SCM 업무를 수행하기에 충분한 시간으로 적절한 툴, 장치 및 훈련을 소프트웨어 환경에 제공해야 한다. SCM 프로세스는 SCM 계획에 기록된 SCM 업무를 수행해야 한다.

6.4 SCM 프로세스의 검토와 평가

SCM 프로세스는 SCM 업무가 SCM 계획에 부합하는지 확인해야 한다. SCM 프로세스는 문제해결과 프로세스 개선 프로세스와 같은 프로세스를 수행해 SCM 계획과의 차이를 해결해야 한다.

6.5 마감

SCM 프로세스는 필요한 경우 SCM 활동을 종료해야 한다.

7. 소프트웨어 형상식별

ISO/IEC 12207: 1995

6.2.2 형상 식별 이 활동은 다음과 같은 업무로 구성된다.

6.2.2.1 프로젝트를 위해 제어해야 할 소프트웨어 항목 및 버전의 식별을 위한 계획을 확립해야 한다. 각 소프트웨어 항목과 버전의 경우 다음의 내용을 식별해야 한다. 기준을 확립하는 문서, 버전 참조번호 및 기타 식별 내용이 그것이다.

SCM 프로세스는 SCI와 기준으로서 제어되는 소프트웨어 제품에 대한 식별 계획을 확립해야 한다.

7.1 SCI 식별

SCM 프로세스는 각 SCI에 고유 식별번호를 부여해야 한다. SCM 프로세스는 SCI 사이의 관계를 문서화해야 한다.

SCM 프로세스는 SCI 개발, 제어, 빌드, 확인, 부하 및 재작성에 사용하는 도구에 고유 식별번호를 부여해야 한다.

7.2 소프트웨어 형상 기준의 식별

SCM 프로세스는 다음과 같은 관점에서 각 기준을 식별해야 한다.

a) 모든 기성제품 및 공급업자에게 사용권은 있지만 소유권은 없는 독점적 항목을 포함해 각 기준에서 제어되는 SCI
b) SCI를 기준에 입력하기 위해 사용하는 절차
c) 기준을 완벽하게 구성하고 확립하기 위한 절차
d) 기준 정의에 필요한 소프트웨어 제품과 기록
e) 기준 승인에 필요한 절차
f) 기준 승인에 필요한 권한
g) 기준 구축을 위한 툴

참고) 위의 b)와 c) 절차에는 확인 프로세스를 사용해야 한다.

7.3 소프트웨어 라이브러리의 식별

SCM 프로세스는 다음을 포함해, 고유의 이름이 부여되고 제어되는 소프트웨어 라이브러리를 식별해야 한다.

a) 위치

b) 각 라이브러리에 대한 매체
c) 동등한 내용 유지를 위한 동일한 라이브러리와 메커니즘의 수
d) SCI 관점에서의 내용
e) SCI 상태 관점에서의 내용
f) 소프트웨어 라이브러리 내용과 호환되는 최소한의 상태를 포함해 SCI를 입력할 수 있는 조건
g) 효과적인 복구 절차와 함께 악의적 및 우발적 위해와 기능저하를 방지하기 위한 규정
h) SCI의 검색, 복사나 제거를 하지 않은 상태의 사용과 복사를 한 상태의 사용 사이의 구별 및 SCI 제거를 위한 조건
i) 각 라이브러리에 SCI 입력, 라이브러리에 수록된 목록과 내용의 확인, 각 라이브러리의 SCI 평가 복사 및 삭제라는 각 기능을 발휘하도록 허가된 담당자 그룹이나 유형에 대한 접근에 대한 규정

7.4 개선 상태

SCM 프로세스는 각 SCI 및 기준에 대한 상태를 확립해야 한다.

SCM 프로세스는 형상이 통제되는 각 SCI 및 기준의 개선을 명시해야 한다.

참고) SCM 프로세스는 특히 "개방"과 "폐쇄"라는 의미에 있어 제안된 변경의 상태를 식별해야 한다.

8. 소프트웨어 형상통제

ISO/IEC 12207: 1995

6.2.3 형상통제: 이 활동은 다음과 같은 업무로 구성된다.

6.2.3.1 다음을 수행해야 한다. 변경 요청의 식별과기록, 변경 분석과 평가, 요청의 승인이나 비승인 및 수정된 소프트웨어 항목의 구현, 확인과 릴리즈가 그것이다. 각 수정, 수정의 이유 및 수정의 승인을 추적할수 있도록 심사 결과가있어야 한다. 안전과 보안에 중요한기능을 취급하는 제어된 소프트웨어 항목에 대한 모든 접근을 통제하고 심사해야 한다.

SCM 프로세스는 SCI 완전성, 기준 및 SCI와 기준을 정의하는 소프트웨어 제품의 유지를 위한 절차를 확립하고 유지하며 운영해야 한다.

8.1 변경 제안

SCM 프로세스는 변경 제안을 접수하고 기준 SCI에 대한 변경을 처리해야 한다.

형상 통제에 따라 SCI 및 기준에 대해 제안한 변경을 식별하고 기록하며, 승인/비승인하고 추적해야 한다.

8.2 제안된 변경 영향의 평가

SCM 프로세스는 예를 들어 유지보수 프로세스를 사용해, 제안된 변경의 영향 평가를 지원해야 한다.

SCM 프로세스는 다음을 식별해야 한다.

a) 제안된 변경의 영향을 받는 SCI 및 관련 기준

b) 식별한 SCI나 기준에 영향을 주는 승인된 수정

SCM 프로세스는 문제해결 조치를 위한 프로그램의 복제나 확인을 위해 SCI를 구성해야 한다.

SCM 프로세스는 문제해결이나 프로세스 개선 프로세스를 사용해 강화에 대한 편차, 오해 또는 제안의 발생 원인을 추적하고 기록해야 한다.

8.3 변경의 구현

SCM 프로세스는 승인된 각 수정에 대한 활동과업무의 순서를 기록해야 한다.

SCM 프로세스는 승인된 수정만 기준에 포함되는지 확인해야 한다.

8.4 처리의 통보

SCM 프로세스는 처리에 영향을 받는 모든 관계자에게 통보함으로써 제안된 각 변경의 승인, 비승인 또는 유예에 대해 지정된 변경권한 의사결정을 지원해야 한다.

주 1. 결정이 유예될 경우 SCM 프로세스는 발의자에게 다시 검토를 진행하는 방법을 알려주어야 한다. 변경이 승인된 경우 SCM 프로세스는 승인된 변경과 유예된 변경을 해당 SCI를 사용하는 관계자에게 통보해야 한다.

주 2. 승인되지 않는 경우 SCM 프로세스는 관계자에게 제안된 변경을 무시할 것을 통보해야 한다.

8.5 변경의 마감

SCM 프로세스는 승인된 수정에 따라 새로운 기준이 발생하는지 확인해야 한다.

9. 소프트웨어 형상 상태 설명

ISO/IEC 12207: 1995

6.2.4 형상상태 설명. 이 활동은 다음과 같은 업무로 구성된다.

6.2.4.1 기준을 포함해 제어된 소프트웨어 항목의 상태와 이력을 나타내는 관리 기록과 상태 보고서를 작성해야 한다. 상태 보고서에는 프로젝트에 대한 변경의 숫자, 최신 소프트웨어 항목 버전, 릴리즈 식별자, 릴리즈의 숫자 및 릴리즈의 비교를 포함해야 한다. 구성 상태 설명 활동은 다른 SCM 프로세스 활동의 기록을 수행한다.

9.1 식별번호의 기록

SCM 프로세스는 새 SCI 및 수정한 SCI의 식별번호와 상태를 기록해야 한다.

SCI가 형상 통제될 경우 SCM 프로세스는 각 후속 개선에 있어 버전과 상태를 유지해야 한다.

9.2 변경 추적

SCM 프로세스는 제안된 변경의 상태와 승인된 수정의 구현 상태를 추적, 기록 및 보고해야 한다.

참고 1. 제안된 변경의 추적은 비승인의 공식적 통보나 하나 이상의 SCI 수정을 위한 요구조건의 발표 또는 발의자에 의한 철회를 통해 아이디어를 처음 통보한 시점부터 시작해야 한다.

2. 승인된 수정의 추적은 기준에 포함시키도록 하나 이상의 SCI를 수정해야 하는 요구조건 발표 후 시작해야 한다.

9.3 상태 설명 기록의 보고

SCM 프로세스는 다음을 보고해야 한다.

a) 소프트웨어 제품의 구조

b) 수취인에 대한 중요도 수준에서 각 SCI의 상태

c) 제안된 변경의 상태

d) 승인된 수정 및 기준 버전

e) 릴리즈의 식별번호

SCM 프로세스는 기준의 현재 및 과거 버전에대한 상태를 보고해야 한다.

SCI에 확인된 편차가 있을 경우 SCM 프로세스는 다음을 수행해야 한다.

a) 그러한 조건을 보고한다.
b) SCI를 식별한다.
c) 사후영향을 설명한다.
d) 일시적인 해결책을 제공한다.

필요한 경우 SCM 프로세스는 해결되지 않은 수정, 편차 또는 포기를 보고해야 한다.

참고. 소프트웨어 제품이 문제해결 프로세스를 사용해 구매자에게 양보, 편차 또는 포기의허용을 요청할 경우 SCM 프로세스는 그 중 어떤 양보가 허용되는지 보고해야 한다.
소프트웨어 제품에 구매하거나 구매자가 공급한 제품이 포함될 경우 SCM 프로세스는 소유권의 추적성(예를 들어 라이선스 판매와 저작권)을 보고해야 한다.

10. 소프트웨어 형상평가

ISO/IEC 12207: 1995

6.2.5 형상평가 이 활동은 다음과 같은 업무로 구성된다.

6.2.5.1 다음을 판단하고 확인해야 한다. 소프트웨어 시스템의 요구조건과 물리적 완벽성과 비교한 소프트웨어 항목의 기능적 완벽성(설계와 코드가 최신 기술을 반영하는지의 여부)

SCM 형상평가는 다음을 결정한다.

a) SCM 기록에 해당되는 제어된 라이브러리에 SCI가 저장되어 있는지의 여부
b) SCI의 누적 상태 및 SCI가 구축되는 승인된수정과 관련해 소프트웨어 제품이 완벽하고 사용할 수 있다.
c) 기준 SCI가 관련 SCI 및 승인된 각 수정으로 구성된다.

SCM 프로세스는 확인 및 심사 프로세스를 지원해평가 중인 SCI, 기준 및 소프트웨어 제품의 완벽성을 확인해야 한다.

SCM 프로세스는 형상평가를 수행해 기준을 구성하는 SCI가 안전하게 저장되었는지 판단해야 한다.

SCM 프로세스는 형상평가 결과를 보고해야 한다.

편차가 파악될 경우 SCM 프로세스는 문제해결 또는 프로세스 개선 프로세스를 수행해야 한다.

11. 소프트웨어 릴리즈 관리 및 인도

ISO/IEC 12207: 1995
6.2.6 소프트웨어 릴리즈 관리 및 인도. 이 활동은 다음과 같은 업무로 구성된다.
6.2.6.1 소프트웨어 제품과 문서의 릴리즈와 인도는 공식적으로 통제해야 한다. 코드와 설명서의 마스터복사본은 소프트웨어 제품 수명 동안 유지해야 한다. 안전 및 보안에 중요한기능이 수록된 코드와 설명서는 관련 조직의 정책에 따라 취급, 보관, 포장 및 인도해야 한다.

SCM 프로세스는 승인된 복수의 수정을 조정하고 완벽성과 정확성을 확립하며, SCI를 다시 구성하고 소프트웨어 제품을 전달하기 위한 절차를 확립, 유지 및 수행해야 한다.

11.1 취급

SCM 프로세스는 릴리즈 관리 및 인도에 대한 모든 입력과 출력을 제어해야 한다.

SCM 프로세스는 기준 라이브러리에서 발표된 SCI를 이전 버전이 필요한수준으로 유지되는 범위까지 미래에 다시 구성할 수 있는지확인해야 한다.

SCM 프로세스는 소프트웨어 환경을 다시 구축할 수 있어야한다.

참고. 기준을 도출하는 절차는 기준의 일부로 간주해야 한다. 소프트웨어 툴의 운영이나 소프트웨어 툴에 대한 수정을 위한 설명과 매개변수는 소프트웨어 툴의 일부로 다시 확립해야 한다.

SCM 프로세스는 기준 소프트웨어 라이브러리 및 소프트웨어 환경을 유지해야 한다.

11.2 보관

SCM 프로세스는 사용 매체나 라이브러리에 관계 없이 보관된 SCI의 완전성을 다음을 수행함으로써 확인해야 한다.

a) 재발 오류나 성능저하의 최소화를 위한 저장 매체를 선택한다.
b) 매체의 저장 수명에 적합한 주기로 저장된 SCI를 활용하고 새로 고친다.
c) 손실 위험의 최소화를 위해 제어된 위치에 복사본을 이중으로 저장한다.

재사용 SCI의 경우 SCM 프로세스는 다음을 수행해야 한다.
a) 공공 이름을 사용한다.
b) 저장 방법, 위치 및 시점을 지정한다.
c) SCI의 사용을 담당하는 조직을 식별한다.

11.3 복제

복제는 복사본을 만들어 소프트웨어를 제조하는 단계이다.

SCM 프로세스는 일관성 있고 완벽한 복제의 보증을 위한 절차를 확립해야 한다.

SCM 프로세스는 릴리즈를 위한 매체에 이질적 항목(예를 들어 소프트웨어 바이러스나 데모에 부적합한 시험 데이터)이 없는지 확인해야 한다.

SCM 프로세스는 적합한 매체를 사용해 소프트웨어 제품이 제대로 복제되는지 확인해야 한다. 매체는 공급품의 예상 수명 동안 내용의 완전성을 보존할 수 있도록 선정해야 한다.

11.4 포장

SCM 프로세스는 인도된 매체가 승인된 절차에 의해 준비된 것인지 확인해야 한다.

SCM 프로세스는 구매자가 확인할 수 있는 릴리즈의 식별번호를 명확히 표시해야 한다.

참고 1. 물리적 매체(예를 들어 CD-ROM이나 자기 테이프)를 사용해 릴리즈할 경우 마크는 릴리즈가 포함된 매체에 표시해야 한다(영구 용기 포함). 전자 릴리즈의 경우(예를 들어 운영 라이브러리에 다운로드) 마크는 릴리스 안에 표시해야 한다.

2. SCM 프로세스는 소프트웨어 제품과 함께 포장하는 다른 자료(예를 들어 라이선스 계약서와 저작권 명세서)를 포함해야 하며 복사본을 SCI에 저장해야 한다.

11.5 인도

SCM 프로세스는 인도 절차를 따라야 한다.

12. 인터페이스 제어

SCM 프로세스는 인터페이스(예를 들어 하드웨어, 시스템 소프트웨어, 지원 소프트웨어, 기성제품 및 병렬/동시 개발로 제작한 소프트웨어) 문서를 식별하고 제어해야 한다. 인터페이스는 상호 합의로 조정할 수 있으며(예를 들어 소프트웨어 통합자와 하청업체 소프트웨어 개발자) 어느 한 당사자가 지정할 수 있다(예를 들어 라이선스 사용자가 제품을 복제할 수 있는 기성 소프트웨어의 공급업체).

SCM 프로세스는 다음을 식별해야 한다.

a) 인터페이스의 목적
b) 인터페이스에서의 요구조건
c) 해당 조직
d) 인터페이스 되어 제어해야 할 문서
e) 인터페이스에 영향을 주는 제안된 변경을 다른 관계자에게 통보하고, 합동으로 또는 별도로 인터페이스 전체의 영향 평가를 수행하는 절차
f) 인터페이스 변경 권한을 포함해 승인, 변경 및 릴리즈할 인터페이스 문서의 절차
g) 인터페이스 문서의 변경을다른 SCI에 대한 변경으로 변환하기 위한 절차
h) 역할과 책임

ISO 9001:2000	ISO/IEC 12207	ISO/IEC 9126	ISO/IEC 12119	ISO/IEC 14598	ISO/IEC TR 15271	ISO/IEC 15504	ISO/IEC TR 15846	ISO/IEC 15939	ISO/IEC TR 16326	ISO 10007	ISO/IEC 6592	ISO/IEC 19761, ISO/IEC 20926 & ISO/IEC 20968	ISO/IEC 14764	ISO/IEC 15026	ISO/IEC 15910	ISO/IEC 14102
4 품질경영시스템																
4.1 일반 요구사항	X				Annex C											
4.2 문서화 요구사항	6.1, F.2.1															
5 경영책임																
5.1 경영의지																
5.2 고객중심																
5.3 품질방침																
5.4 기 획																
5.4.1 품질목표						X										
5.4.2 품질경영시스템 기획																
5.5 책임, 권한 및 의사소통																
5.6 경영검토																
6 자원관리																
6.1 자원의 확보																
6.2 인적자원																
6.2.1 일반사항	F3.3.1, F3.32															

[표 2-4] ISO/IEC JTC1/SC7, ISO/TC 176 관련 규격과 ISO 9001:2000간의 이행 관련 유용성

ISO 9001:2000	ISO/IEC 12207	ISO/IEC 9126	ISO/IEC 12119	ISO/IEC 14598	ISO/IEC TR 15271	ISO/IEC 15504	ISO/IEC TR 15846	ISO/IEC 15939	ISO/IEC TR 16326	ISO 10007	ISO/IEC 6592	ISO/IEC 19761, ISO/IEC 20926 & ISO/IEC 20968	ISO/IEC 14764	ISO/IEC 15026	ISO/IEC 15910	ISO/IEC 14102
6.2.2 능력, 인식 및 교육훈련																
6.3 기반구조	7.2, F.3.2			Pt 2, Pt 3												X
6.4 업무환경																
7 제품 실현																
7.1 제품 실현의 기획	5.2.4, 5.3.1, 6.1 to 6.8, F.2	Pt 1		Pt 2			6.2		6.2.2							
7.2 고객 관련 프로세스																
7.2.1 제품 관련 요구사항의 결정	5.3.2 to 5.3.4, F.1.3.1,2,4	Pt 1	X											X		
7.2.2 제품 관련 요구사항의 검토	5.2.1, 5.2.6, 6.4.2.1, 6.6, F.3.1.5															
7.2.3 고객과 의사소통	5.2.5.6, 5.2.6, 5.2.7, 6.6, F.1.4.2												6.6.1, 7.3.3, 8.2, 8.2.3			
7.3 설계 및 개발	F.1.3.4, F.1.3.5		X								X	X			X	
7.3.1 설계 및 개발 기획	5.2.4, 5.3.1								6.2.2							
7.3.2 설계 및 개발 입력		Pt 1														
7.3.3 설계 및 개발 출력	5.3.5 to 5.3.7															
7.3.4 설계 및 개발 검토	5.3.4.2, 5.3.5.6, 5.3.6.7, 6.6.3, F.2.6				Annex A											

ISO 9001:2000	ISO/IEC 12207	ISO/IEC 9126	ISO/IEC 12119	ISO/IEC 14598	ISO/IEC TR 15271	ISO/IEC 15504	ISO/IEC TR 15846	ISO/IEC 15939	ISO/IEC TR 16326	ISO 10007	ISO/IEC 6592	ISO/IEC 19761, ISO/IEC 20926 & ISO/IEC 20968	ISO/IEC 14764	ISO/IEC 15026	ISO/IEC 15910	ISO/IEC 14102
7.3.5 설계 및 개발 검증	5.3, 6.5, F.1.3, F.2.4															
7.3.6 설계 및 개발 유효성 확인	5.3, 6.5, F.1.3, F.2.5			Pt 3, Pt 5												
7.3.7 설계 및 개발 변경 관리	5.5.2, 5.5.3, 6.1, 6.2, F.2.1, F.2.2															
7.4 구매		Pt 1		Pt 4								X				
7.4.1 구매 프로세스	5.1, F.1.1					Pt 3										
7.4.2 구매정보	5.1.2, F.1.1.1															
7.4.3 구매 제품의 검증	5.1.5, F.1.1.4															
7.5 생산 및 서비스 확보		Pt 1					X						X		X	
7.5.1 생산 및 서비스 확보 관리	5.3.12, 5.4.4, 5.5, 6.3.3, 6.8, F.1.3.11, F.1.4.2, F.1.5, F.2.8															
7.5.2 생산 및 서비스 확보 프로세스의 타당성 확인																
7.5.3 식별 및 추적성	6, F.2.2						7 to 12			X						
7.5.4 고객자산																
7.5.5 제품의 보존																
7.6 모니터링 및 측정장치 관리																

ISO 9001:2000	ISO/IEC 12207	ISO/IEC 9126	ISO/IEC 12119	ISO/IEC 14598	ISO/IEC TR 15271	ISO/IEC 15504	ISO/IEC TR 15846	ISO/IEC 15939	ISO/IEC TR 16326	ISO 10007	ISO/IEC 6592	ISO/IEC 19761, ISO/IEC 20926 & ISO/IEC 20968	ISO/IEC 14764	ISO/IEC 15026	ISO/IEC 15910	ISO/IEC 14102
8 측정, 분석 및 개선																
8.1 일반사항	7, F.3.3	Pt 2, Pt 3		Pt 2		Pt 1		5								
8.2 모니터링 및 측정																
8.2.1 고객만족		Pt 4														
8.2.2 내부감사	6.3, 6.7, F.2.3, F.2.7															
8.2.3 프로세스 모니터링 및 측정	7.3.2, 7.3.3, F.3.3.2					Pt 1, Pt 2		5								
8.2.4 제품 모니터링 및 측정	5.3, F.1.3	Pt 1		Pt 3, Pt 5												
8.3 부적합 제품의 관리	6.2, 6.8, F.2.2, F.2.8		X				X									
8.4 데이터의 분석								5.4				X				
8.5 개선																
8.5.1 지속적 개선	7.3, F.3.3					X										
8.5.2 시정조치	6.8, F.2.8															
8.5.3 예방조치	7.3.2, F.3.3.2					Pt 2										

[별첨 3]

의료기기에 포함된 소프트웨어의 시판 전 신고 지침 (Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices)

FDA CDRH
2005. 5. 11 발행

이 문서는 의료기기에 포함된 소프트웨어의 시판전 신고에 대한 지침(1998.5.29 발행)을 대체하며, 혈액처리(Blood Establishment) 컴퓨터 소프트웨어의 시판전 신고에 대한 지침(1997.1.13 발행)의 재검토이다.

CDRH에서 규제하는 의료기기와 관련하여 본 문서에 의문점이 있으면 (301) 443-8517의 David S. Buckles에게, CBER에서 규제하는 의료기기에 관한 의문점이 있으면 (301) 827-6136의 Linda Weir에게 연락하기 바란다.

서 문

이 지침은 독립적으로 사용(stand-alone)되는 소프트웨어 및 하드웨어에 적용 사용되는 소프트웨어를 포함한 소프트웨어 기기에 대하여 시판전 신고에 제출 권고되는 문서와 관련하여 산업계에 정보를 제공하는데 목적이 있다.

이 문서는 FDA의 권고사항을 더욱 명확히 하고 기술 발전에 따른 현재의 시류를 보증하기 위한 노력의 결과이다. 또한, 이 문서는 이전 2종 지침에서의 권고사항을 단일 지침으로 통합한다.

최소 부담 접근방법

이 지침에 명기된 사항(issues)은 의료기기의 시판 전에 수행되어야 함을 나타낸다. 지침 제정과정 동안 FDA는 감독기관(Agency)의 의사결정 관련 법규들을 신중하게 검토했다. 또한 이 지침 및 지침 명기사항에 따라 제조업체에서 발생될 수 있는 부담도 고려했다. FDA는 이 지침 명기사항의 해결을 위하여 가장 효율적인 접근방법(the least burdensome approach)으로 고려하였다. 그러나 명기 사항들에 보다 효율적인 접근법이 있다면, 다음 지침에 기술된 절차에 따라야 한다.

A Suggested Approach to Resolving Least Burdensome Issues(지침),
http://www.fda.gov/cdrh/modact/leastburdensome.html.

이 지침을 포함한 FDA의 지침들은 법률적으로 시행 가능한 책임을 확정하지 않는다. 다만 지침은 주제에 관한 현재 감독기관의 생각을 기술하고 오직 권고사항으로 검토하여야 한다. 그러나 특정 규제사항 또는 법적 요구사항은 소환된다. 감독기관 지침에서 단어 사용은 제안 또는 권고를 의미하지만, 강제사항은 아니다.

적용범위

이 문서의 목적상, 다음 사항과 같이 하나 이상의 소프트웨어 부품, 구성품 또는 부속품을 포함하거나 또는 소프트웨어로만으로 구성된 것을 "소프트웨어 기기(software devices)"로 간주한다.

- 펌웨어 및 소프트웨어에 근거한 의료기기의 제어를 위한 다른 수단
- 독립적 사용(stand-alone) 소프트웨어의 적용
- 일반 컴퓨터에 설치하도록 의도된 소프트웨어
- 의료기기 목적(dedicated)의 하드웨어/소프트웨어
- 소프트웨어를 구성하거나 포함되는 의료기기 부속품(accessories)

이 지침은 소프트웨어가 제3자 판매자(vendor)에 의한 설치, 제조소 설치, 현장 설치 또는 업그레이드 간에 최종 사용자에게 인도되는 방법에 관계없이 소프트웨어 기기에 적용한다.

이 지침에서 제외하는 소프트웨어는 의료기기로서의 목적이 아닌 소프트웨어 및 제조 또는 공정관리용으로 디자인된 소프트웨어를 포함한다. 보다 세부적인 정보 또는 자사의 의료기기에 적용 해당여부를 명확히 알기를 원하는 경우 FDA 관련 검토부서에 문의하기 바란다.

이 지침은 다음과 같은 소프트웨어 기기에 대한 시판 전 신고의 모든 종류에 적용한다.

- 전통적(Traditional), 특별(Special) 및 약식(Abbreviated) 시판전 신고(510(k))
- 시판전 승인(PMA)
- 임상시험 의료기기의 적용면제(IDE)
- 보상을 포함한 인도적 의료기기의 적용면제(HDE)

다른 문서와의 관계

FDA 지침들

본 문서는 소프트웨어 관련 권고사항을 제공하는 기존의 다른 지침을 보충하려는 의도이다. 예를 들어, 본 문서에서는 소프트웨어 기기(단독 사용기기 또는 기기의 부품, 구성품, 부속품을 포함) 관련 권고사항으로 "소프트웨어 밸리데이션의 일반 원칙 지침"을

참조하도록 권한다. 의료기기가 OTS 소프트웨어를 사용하는 경우 “의료기기의 OTS 소프트웨어 사용에 대한 지침”을 참조하도록 권한다.

소프트웨어 기기 제조업자들은 QSR(21 CFR part 820)의 요구사항에 일치하는 소프트웨어 관련 문서를 수립 유지하여야 한다. 다른 FDA의 지침들 및 본 지침의 권고사항에 따르는 것은 QSR을 준수하는 것으로 대체할 수 없다는 것에 주의 하여야 한다.

소프트웨어 관련 합의규격(Consensus Standards)

소프트웨어 관련 합의규격의 출현은 위험관리 및 위험평가와 같은 정밀한 활동 측면에서 소프트웨어의 개발품질, 문서화 및 일관성 향상에 기여하였다. 본 문서에서 용어와 권고사항은 ISO 14971 및 AAMI SW68과 같은 소프트웨어 관련 합의규격과 조화되도록 하였다.

전문용어

검증 및 밸리데이션

이 문서에서는 QSR에서의 정의와 같이 “검증 및 밸리데이션”(V&V) 용어를 사용한다. 검증이란 “특정 요구사항에 충족된다는 객관적 증거 제시 및 검사에 의한 확인”을 의미한다. 소프트웨어 개발환경에서 소프트웨어 검증은 개발의 특정단계에서 출력이 그 단계의 모든 입력요건을 충족한다는 확인이다. 소프트웨어 시험은 소프트웨어 개발 출력이 그 입력요건을 충족함을 확인하기 위한 여러 검증활동의 하나이다. 다른 검증활동으로 다음 사항들을 포함한다.

- 워크스루(walk-through)
- 다양한 정적(static), 동적(dynamic) 분석
- 코드와 문서 조사
- 모듈 레벨 시험
- 통합 시험

설계 밸리데이션은 “의료기기 규격이 사용자 요구 및 사용목적에 일치함을 객관적인 증거로 입증하는 것”을 의미한다. 본 문서에서 밸리데이션은 “설계 밸리데이션”으로 제한하며 21 CFR 820.3(z)(1)에서 정의한 “공정 밸리데이션”은 포함하지 않는다.

설계 밸리데이션의 한 요소가 소프트웨어 밸리데이션이다. 소프트웨어 밸리데이션은 소프트웨어가 사용된 기능에 대한 사용자 요구와 사용목적이 소프트웨어와 일치함을 객관적 증거로 입증하는 것이다. 소프트웨어의 밸리데이션은 완성된 의료기기의 설계

밸리데이션의 일부분이다. 이것은 의료기기에 소프트웨어를 결합시킨 후 실제 또는 모의 사용 환경에서 소프트웨어의 적절한 작동에 대하여 점검하는 것을 포함한다. 소프트웨어의 밸리데이션은 포괄적인 소프트웨어 시험과 소프트웨어 개발 수명주기의 각 단계에서 이미 완료한 다른 검증 작업에 상당히 의존한다. 기획, 검증, 추적성, 형상관리 및 양호한 소프트웨어 엔지니어링의 다양한 요소들은 소프트웨어가 유효하다는 결론도출에 도움을 주는 중요한 활동이다.

경미하고 심각한 상해

본 문서에서의 용어 "경미한 상해"는 21 CFR 803.3(bb)(1)의 정의와 같이 심각한 상해(serious injury)에 해당되지 않는 것을 의미한다. 규정에서 심각한 상해 또는 질병은 다음과 같이 정의한다.

- 생명에 위협적이다.
- 인체기능의 영구적 장애 또는 인체구조의 손상을 초래한다.
- 인체기능의 영구적 장애 또는 인체구조의 손상을 배제하기 위한 의학적 또는 수술이 필요하다.

본 문서에서 용어 "영구(permanent)"는 "사소한 약화나 손상을 제외한 인체구조 또는 기능에 대한 회복가능성이 없는 장애 또는 손상"으로 정의한다. 21 CFR 803.3(bb)(2).

심각도

서문

시판전 신고에 포함될 문서는 일반적으로 의료기기의 심각도에 의존한다. 심각도는 의료기기의 고장, 설계 결함 또는 의료기기 사용목적에 따른 단순 사용의 결과로 환자나 조작자에게 직접 또는 간접적으로 가해질 수 있는 상해의 심각성 평가를 의미한다. 환자 또는 조작자에게 상해를 입힐 수 있는 위험요인의 발생, 관리 및/또는 감소에 대한 소프트웨어 역할을 명확히 기술하도록 권고한다. 또한 이것들은 자사 의료기기의 해당 심각도를 결정하는 요인이다.

소프트웨어 기기에 대한 시판 전 제출 문서의 범위는 이러한 심각도에 비례한다. 심각도는 이러한 배경에서 정의하며, 의료기기의 등급분류(1,2,3등급) 또는 본질적인 위험요인이나 위험분석에 관계하지 않는다.

심각도의 상, 중, 하

소프트웨어 기기에 해당되는 심각도 수준과 각 수준별 심각도에 따라 신고하는 문서에

대한 권장사항을 설명한다. 관련 위험 요인을 감소시키기 전에 심각도를 결정하도록 권한다. 즉, 심각도는 개별 위험요인에 대한 경감 영향에 관계없이, 경감 작업없이 위험요인 분석으로 다루어져야 한다. FDA는 소프트웨어 기기에 대하여 결정된 심각도를 제출문서에 명기하도록 권한다. 심각도는 다음과 같이 "상, 중, 하"가 있다. 또한 어떻게 심각도를 결정하였는지에 대하여 설명하도록 권한다. 심각도는 의료기기 기능에 연관된 소프트웨어 운영이 환자나 조작자에 미치는 영향에 기초하여 결정한다. 영향은 직접적 또는 간접적일 수도 있다.

상 (Major)

고장이나 잠재적인 결함이 직접적인 원인으로 환자나 조작자에게 사망 또는 중상을 입힌 경우 심각도는 '상'에 해당된다. 고장이나 잠재적 결함이 간접적인 원인으로 작용하여 부정확하거나 지연된 정보로 환자나 조작자에게 사망 또는 중상을 입힌 경우에도 심각도는 '상'에 해당한다.

중(Moderate)

고장이나 잠재적 설계 결함이 직접적 원인이 되어 환자나 조작자에게 경상을 입힌 경우 심각도는 '중'이며, 고장이나 잠재적 결함이 간접적 원인으로 작용하여 부정확하거나 지연된 정보로 환자나 조작자에게 경상을 입힌 경우 심각도는 '중'에 해당한다.

하(Minor)

고장이나 잠재적인 설계, 결함으로 환자나 조작자에게 어떤 부상도 발생할 가능성이 없는 경우 심각도는 '하'에 해당한다.

심각도 수준의 결정

심각도 결정을 위하여 다음과 같은 핵심 질문을 제시한다. FDA는 위험요인 경감을 시도하기 전에 심각도를 평가하도록 권장한다. 즉, 위험이 감소되지 않은 상태에서 이 질문들에 응답하여 소프트웨어 기기를 평가하여야 한다.

질문에 대한 응답이 부정일 경우, 다음 질문으로 넘어간다. 나중에 상세히 설명하겠지만, 시판전 제출 문서에 심각도를 결정한 근거를 포함하도록 권한다. 어떤 경우에도 소프트웨어 기기의 고장으로 발생하는, 최악의 가능성, 합리적 예상, 임상적 결과의 관점에서 심각도를 평가하도록 권한다.

사용하는 소프트웨어 기기의 심각도에 의문이 있는 경우 언제라도 FDA의 검토부서에 문의하기 바란다. 심각도가 '상'으로 판단되는 소프트웨어 기기에 대한 시판전 신고를 아직 제출하지 않은 경우, 제출 전에 FDA의 관련 부서에서 해당 소프트웨어 기기에 관하여 협의할 것을 권한다.

소프트웨어 관련 문서화

시판 전 신고에서 제출할 소프트웨어 관련 문서는 소프트웨어 기기의 사용목적, 심각도 및 제출 형태에 일치하여야 한다. 본 장에서는 심각도에 따라 시판전 신고에 포함할 문서들을 언급한다.(표 3) 그러나 의료기기에 대한 특정 지침이 있는 경우 그 권고사항에 따른다. 일반적으로 제출할 문서들은 다음과 같다.

질문의 답이 하나라도 yes인 경우 소프트웨어 기기의 심각도는 '상'일 가능성이 높다.
소프트웨어 기기가 혈액처리(Blood Establishment) 컴퓨터 소프트웨어처럼 중요한가? (혈액처리 컴퓨터 소프트웨어는 혈액과 혈액 성분의 제조에 사용하는 소프트웨어 제품 또는 혈액처리 담당자가 헌혈자의 적합성을 판단하거나 수혈 또는 후속 제조를 위한 혈액이나 혈액 성분 공급의 판단에 사용하는 소프트웨어 제품으로 정의된다.)
소프트웨어 기기가 의약품이나 생물제재와 조합하여 사용하는가?
소프트웨어 기기는 심각도가 '상'인 의료기기의 부속품인가?
위험요인을 경감시키기 전, 소프트웨어 기기의 고장으로 환자나 의료기기 사용자의 사망이나 중상이 발생될 수 있는가? 이러한 예는 다음과 같다.
소프트웨어 기기가 생명보조(life supporting) 또는 생명유지 기능을 제어하는가?
소프트웨어 기기가 방사선 치료 시스템, 제세동기 및 절제 발생기(ablation generators)처럼 사망이나 중상을 유발할 수 있는 잠재적으로 유해한 에너지의 전달을 제어하는가?
소프트웨어 기기가 오류나 오작동(malfunction)이 발생할 경우 사망이나 중상을 유발할 수 있는 치료나 요법(therapy)을 제어하는가?
잘못 적용할 경우 중상이나 사망을 유발할 수 있는 치료나 요법의 결정에 직접적인 영향을 주는 진단(diagnostic) 정보를 소프트웨어 기기가 제공하는가?
소프트웨어 기기는 의학적 시술(intervention)이 필요한 잠재적인 생명 위급 상황에서 생명유지 신호(vital signs)의 모니터링 및 경고(alarms) 기능을 제공하는가?

[표 1] 상(Major) 심각도

소프트웨어 기기의 심각도가 '상'이 아니면서, 다음 질문에 하나라도 yes인 경우 심각도는 '중'에 해당된다.
소프트웨어 기기는 심각도가 '중'인 의료기기의 부속품인가?
위험요인을 경감시키기 전, 소프트웨어 기기의 고장으로 환자나 의료기기 사용자의 경상이 발생될 수 있는가?
소프트웨어 기기의 오작동 또는 잠재적 설계 결함으로 진단에 오류가 발생하거나 적절한 의학적 진료(care)가 지연되어 경상을 유발할 가능성이 있는가?

[표 2] 중(Moderate) 심각도

표 1과 2의 모든 질문에 대한 답이 'No'인 경우, 그 심각도는 '하'에 해당된다.

- 의료기기 설계(design) 설명
- 설계가 어떻게 이행되었는지에 대한 문서
- 설계에 따라 생산된 기기가 어떻게 시험되었는지의 입증
- 적절한 위험요인 식별 및 효과적인 위험관리의 증명
- 설계, 이행, 시험 및 위험관리와 추적성의 연결

기기의 심각도를 핵심으로 하는, 제출하는 문서의 종류와 범위는 표 3에 요약한다. 이 권고사항은 설계관리를 포함한 효과적인 이행 및 QSR의 관리에 근거한다.
제출 권장 문서들은 소프트웨어 기기의 개발과정에서 일반적으로 생성되는 문서이다. 그러므로 소프트웨어 개발환경에서 적절히 관리되고 문서화된 제출문서는 자사 제품 개발 문서의 복사본일 수도 있다.

FDA가 제출을 권장하는 문서는 표 3의 뒤에서 설명하기로 한다. 심각도에 대해 권장되는 문서는 신고서의 일부를 구성하는 경우도 있다. 소프트웨어 요구사항 규격(SRS)과 같은 문서는 신청서에 복사하여 제출할 수도 있다.

심각도 수준

소프트웨어 기기의 심각도는 어떠한 경감의 영향이 있기전에 결정하도록 권한다. 심각도의 3수준 중 자사 의료기기의 해당 수준을 명시하고 그 수준을 결정한 이론적 근거를 관련문서에 포함시킬 것을 권한다. 또한 자사의 의사결정 과정(decision-making process)을 문서화하도록 권장한다.

소프트웨어 설명서

소프트웨어가 제어하는 의료기기 특성(features)의 포괄적인 개요와 운영환경을 설명하는 문서를 제출하도록 권한다. 일반적으로 정보는 단락 형식(paragraph format)으로 작성하고, 중요하거나 운영에서 의미가 큰 소프트웨어의 특성을 언급하도록 권한다.

소프트웨어 설명서에는 다음의 정보를 포함하여야 한다.

- 프로그래밍 언어
- 하드웨어 플랫폼(platform)
- 해당되는 경우, 운영 시스템(operating system)
- 해당되는 경우, OTS 소프트웨어 사용

의료기기에 기성품 소프트웨어를 사용할 경우 FDA의 “Guidance for Off-the-Shelf Software Use in Medical Devices." 지침을 참조하기 바란다.

이 정보가 소프트웨어 요구사항 규격(SRS)과 같이 다른 문서에 포함되는 경우, 제출 문서에는 이 정보의 위치를 나타내는 주석(annotation)과 참고사항(reference)을 명시하여야 한다.

의료기기 위험요인 분석

모든 소프트웨어 기기에 대하여 의료기기 위험요인 분석을 제출하도록 권한다. 의료기기 위험요인 분석은 하드웨어와 소프트웨어 위험요인을 포함하여 의료기기 사용목적과 관련된 모든 의료기기 위험요인을 고려하여야 한다. 식별된 각 위험요인에 대한 정보를 표 형식(tabular form)으로 제출하도록 권한다. 이 문서는 ISO 14971에 명기된 위험관리 요약서(Risk Management Summary)와 같이 포괄적인 위험관리 문서에서 소프트웨어 관련 항목의 발췌 형식일 수도 있다. 이러한 형식인 경우, 다음과 같은 정보를 포함하여야 한다.

- 위험스러운 사상(事象, event)의 식별
- 위험요인의 심각도

[표 3] 심각도에 근거한 문서화

소프트웨어 문서	하(Minor)	중(Moderate)	상(Major)
심각도	심각도의 표시 및 해당 심각도에 대한 이유를 기술		
소프트웨어 설명서	개략적인 특징(features) 및 소프트웨어 운영환경(operating environment.) 요약		
의료기기 위험요인 분석	심각성 평가 및 감소를 포함시킨 하드웨어 및 소프트웨어의 위험요인 식별표(Tabular)		
소프트웨어 요구사항 규격 (SRS)	SRS에서의 기능별(functional) 요구사항 요약	완전한 SRS 문서	
구조 설계도	제출 필요 문서 없음	기능별 유닛 및 소프트웨어 모듈에 대한 상세한 설명. 흐름도와 같은 diagrams 포함 가능	
소프트웨어 설계 규격 (SDS)	제출 필요 문서 없음	완전한 SRS(Software design specification) 문서	
추적성 분석	요구사항, 설계 규격, 식별된 위험요인, 위험감소, 검증 및 밸리데이션 시험간의 추적성.		
소프트웨어 개발 환경 설명서	제출 필요 문서 없음	형상관리 및 유지활동 요약서를 포함한 소프트웨어 개발 수명주기 계획 요약서.	소프트웨어 개발 수명주기 계획 요약서. 형상관리 및 유지계획 문서를 포함하여 개발과정에서 작성된 문서들
검증 및 밸리데이션 문서	소프트웨어 기능별 시험 계획 합부판정기준 및 결과	유닛, 통합 및 시스템 레벨에서 VV&T 활동에 대한 설명서. 합부판정기준, 시험결과를 포함한 시스템 레벨의 시험계획(protocols)	유닛, 통합 및 시스템 레벨에서 VV&T 활동에 대한 설명서. 합부판정기준, 시험보고서, 요약 및 시험결과를 포함한 유니트, 통합 및 시스템 레벨의 시험계획(protocols) 프로토콜
개정판 이력	릴리즈(release) 버전번호와 일자를 포함한 개정이력 일지(log)		
미해결된 변종 (버그 및 결함)	제출 필요 문서 없음	잔존하는 소프트웨어 변종 목록, 조작자 사용 및 인체요소(human factors)를 포함하여 안전 또는 효율성에 대한 영향을 설명한 주석(annotate).	

- 위험요인의 원인
- 관리방법(예: 경고(alarm), 하드웨어 설계)
- 의료기기 설계/요구사항 측면에서의 설명을 포함하여, 위험스러운 사상의 제거, 감소 또는 경고(warn)와 같이 취해진 시정조치
- 관리 방법의 올바른 이행여부에 대한 검증

위험요인 분석을 수행하는 경우, 의료기기의 의도적이거나 부주의한 오용(misuse)으로 발생하는 위험요인을 포함하여 예상 가능한 모든 위험요인을 다루도록 권한다.

소프트웨어 요구사항 규격

소프트웨어 요구사항 규격(SRS)에는 소프트웨어 요구사항이 규정되어 있다. 이 규격에는 전형적으로 소프트웨어의 기능, 성능, 인터페이스, 설계 및 개발요구사항들이 포함된다. 실제 이 문서로 소프트웨어 기기가 무엇을 하는지에 대하여 기술한다. SRS에 포함될 전형적인 요구사항의 예는 다음과 같다. 심각도가 '하'인 소프트웨어 기기는 OTS 소프트웨어의 식별을 포함하여 SRS에 기능 요구사항만을 제시하여도 충분하다. '중'과 '상'의 심각도인 소프트웨어 기기는 완벽한 SRS를 제시하도록 권한다.

하드웨어 요구사항

일반적으로 포함된 하드웨어의 요구사항:

- 마이크로 프로세서(microprocessors)
- 메모리 장치
- 센서
- 에너지 원(sources)
- 안전성
- 커뮤니케이션

프로그램밍 언어 요구사항

메모리 렉(memory leaks) 관리에 대한 정보와 제한 또는 프로그램 크기(size) 요구사항을 포함한 프로그래밍 언어 요구사항

인터페이스 요구사항

시스템 구성요소들 간의 커뮤니케이션 및 프린터, 모니터, 키보드, 마우스와 같은 유저(user)와의 커뮤니케이션 모두를 포함한 인터페이스 요구사항

소프트웨어 성능과 기능 요구사항

소프트웨어 성능 및 기능 요구사항에는 치료, 진단, 감시, 경고, 분석을 위한 알고리즘이나 제어 특성 및 필요한 경우 전체 텍스트 참고문헌(references)이나 보조(supporting) 임상자료의 해석이 포함된다. 소프트웨어 성능과 기능 요구사항에는 다음과 같은 내용이 포함될 수 있다.

- 소프트웨어로 인한 기기의 제한적 조건(limitations)
- 내부 소프트웨어 시험 및 확인
- 오류와 인터럽트(interrupt) 처리
- 결함 감지, 허용차(tolerance) 및 복구(recovery)특성
- 안전 요구사항
- 타이밍(timing)과 메모리요구사항
- 해당되는 경우 OTS 소프트웨어의 식별

구조 설계도(Architecture Design Chart)

이 문서는 통상 네트워크 연결(networking)과 같이 하드웨어와 데이터 흐름과의 관계를 포함하여 소프트웨어 기기내의 주요 기능별 유닛(units)간 관계에 대한 흐름도(flowchart) 또는 이와 유사한 설명이다. 일반적으로 이 문서에 모든 기능 및 모듈을 포함시킬 필요는 없다. 단, 소프트웨어 기기의 기능 및 사용 목적에 관련된 소프트웨어의 구조를 검토하기에 충분한 정보가 포함되어야 한다. 심각도가 '중'과 '상'인 의료기기의 경우, 도해(diagrams)와 같이 소프트웨어 기능별 유닛간의 관계를 명확히 묘사한 상세한 정보가 유용할 수 있다. 구조 설계도가 SRS처럼 다른 문서에 포함되어 있다면, 구조 설계도의 위치에 대한 참고(reference)와 결과(effect)에 대한 설명(statement)을 제출문서에 포함하여야 한다.

소프트웨어 설계 규격(Software Design Specification)

소프트웨어 설계 규격(SDS)은 소프트웨어 기기에 대한 요구사항의 실현(implementation)을 설명한 것이다. SRS 및 SDS간의 관계에서, SRS는 '소프트웨어 기기가 무엇을 하는가?'인 반면 SDS는 'SRS에 기술된 요구사항을 어떻게 실현하는가?'로 정의된다. SDS에 기술된 정보는 소프트웨어 기기를 만든 소프트웨어 엔지니어가 수행하는 작업이 임시적인 설계 결정이 없이 분명하고 명확함을 보증하기에 충분하여야 한다. SDS에는 상세한 소프트웨어 규격처럼 다른 문서와의 참조도 포함될 수 있다. 단, 제출문서에는 사용 목적, 기능성, 안전성 및 활용성과 같은 소프트웨어 요구사항에 대한 실현 계획을 검토하기에 적합한 정보가 포함되어야 한다.

추적성(Traceability) 분석

추적성 분석은 제품 설계 요구사항, 설계 규격 및 시험 요구사항을 상호 연계한다. 또한 식별된 위험요인의 감소에 대한 이행 및 시험을 연결하는 방법을 제공한다. 효과적인 제품 개발에 필수적임과 동시에 제품 설계, 개발, 시험 및 위험요인 감소여부를 파악하는데 필수적인 이러한 활동 및 관련 문서는 명확한 추적성 검토를 위해 제출하도록 권한다. 일반적으로 추적성 분석은 요구사항, 규격 및 시험에 대한 항목과 위험요인 감소를 암시(pointers)하는 매트릭스로 구성된다. 공통번호 체계(common numbering scheme)를 가진 분할된 조직구조(shared organizational structure)로서 문서 추적성을 용이하게 할 수 있다. 그러나 제출된 정보로 FDA 검토자에게 길잡이가 될 수 있는 매트릭스와 같은 구성(mechanism)을 포함하도록 권한다.

소프트웨어 개발환경 설명서

심각도가 '중'과 '상'인 소프트웨어 기기의 경우, 제출문서에는 소프트웨어 개발 life cycle 계획에 대한 요약서를 포함하여야 한다. 요약서에는 소프트웨어 개발 life cycle 및 다양한 life cycle 활동들을 관리하기 위한 과정을 설명하여야 한다. 또한 심각도가 '상'인 소프트웨어 기기의 경우에는 소프트웨어 개발 과정동안 발생된 관리/기준(control/baseline) 문서의 주석판(annotated) 목록 및 소프트웨어 코딩기준(coding standards)의 목록이나 설명을 포함하여야 한다.

형상 또는 변경관리는 소프트웨어 개발에 있어 매우 중요한 요소이다. 최초 시판후 소프트웨어 기기의 변경은, 필요한 경우 명확한 규격 및 잘 정의된(well-defined) 회귀시험(regression testing)을 포함한 시험 계획으로 확실하게 관리하여야 한다. 개발환경 설명서에는 형상관리(configuration management) 및 소프트웨어 개발 life cycle 측면에서의 유지 계획에 관한 정보가 포함되어야 한다. 심각도가 '상'인 의료기기는 형상관리와 유지 계획을 완전히 파악할 수 있는 충분한 상세 설명을 제공하여야 한다. 심각도가 '중'인 의료기기는 형상관리와 유지 계획에 대한 요약서(summary)를 제시하면 된다.

검증 및 밸리데이션 문서

이 지침에서 "검증"과 "밸리데이션"에 대해 소프트웨어 기기 시험의 두 단계와 관련하여 언급한 바 있다. 이 장에서는 소프트웨어 기기의 시판전 신고에 포함되는 시험 문서(documentation)의 종류(type)에 관하여 언급한다.

심각도가 '하'인 기기

심각도가 '하'인 기기의 경우, 시스템이나 기기의 레벨 시험 및 해당되는 경우 통합

(integration) 시험에 관한 문서를 제출하여야 한다. 제출문서에는 시스템이나 기기의 레벨시험 합부판정기준(pass/fail criteria) 및 시험결과 요약서를 포함하여야 한다.

심각도가 '중'인 기기

심각도가 '중'인 기기의 경우, 검증 및 밸리데이션 활동의 요약 목록, 이러한 활동의 결과 및 합부판정기준을 제출하여야 한다. 추적성 분석은 이러한 활동과 결과가 설계 요구사항 및 규격에 효과적으로 연계됨을 보장하여야 한다.

심각도가 '상'인 기기

심각도가 '상'인 기기의 경우 심각도가 '중'인 의료기기에서 요구되는 정보와 불합격된 시험에 대한 설명을 제출하여야 한다. 또한 불합격된 시험에 대응하기 위한 변경(modification)과 그러한 변경이 효과적이었음을 입증하는 결과 문서를 포함시키도록 권한다. 제출될 문서에는 유닛 통합 시험(unit integration testing)의 예시(examples) 및 결과 요약서를 포함하여야 한다.

개정판(Revision Level) 이력

제출문서에는 제품 개발과정에서 발생한 소프트웨어의 개정(revisions) 이력을 포함하여야 한다. 일반적으로 이력은 날짜, 버전 및 이전 버전과 개정 버전의 주요 변경 비교 설명을 포함하여 개발 주기 중 소프트웨어의 주요 변경항목을 표 형식(tabulation)으로 작성한다. 목록의 최종 기재내용은 출하된 기기에 채택된 최종 버전이어야 한다. 또한, 기재내용에는 시험한 소프트웨어 버전과 출하된 버전간의 차이점 및 기기의 안전성과 유효성 차이로 인한 잠재적 영향에 대하여 평가한 것을 포함하여야 한다.

미해결된 변종(버그 또는 결함)

심각도가 '중'과 '상'인 소프트웨어 기기의 제출문서에는 미해결된 모든 소프트웨어 변종(anomalies) 목록을 포함하여야 한다. 각 변종에 대하여 다음 사항을 언급하도록 권한다.

- 문제점
- 의료기기 성능에 미치는 영향
- 해당되는 경우, 문제 해결을 위한 계획이나 일정(time frames)

조작자 사용과 인체공학 문제를 포함한 의료기기 안전성이나 유효성에 대한 변종의 영향 설명을 각 항목별로 주석(annotate)을 달도록 권한다. 통상 이 목록은 미해결된 소프트웨어 변종의 평가와 처리를 위하여 변경관리위원회(change control board) 또는 유사한 조직(mechanism)에서 도출된 결과로 생성된다. 해당하는 경우 의료기기의 적절

한 작동에 지원하도록, 이 목록을 최종 사용자(end user)에게 알릴 것을 권한다. 실행이 가능한 모든 사례에서, 미해결된 변종을 감소시키거나 가능한 회피방법(work-arounds)을 포함하여야 한다: 특히 이 권고사항은 혈액처리 컴퓨터 소프트웨어에 적용된다.

특별(Special) 510(k) 프로그램

특별 510(k) 프로그램에서 적격성을 확인(qualify)하는 시판전 신고대상 의료기기는 이전에 받은 510(k) 의료기기의 수정(modification)한 의료기기가 되어야 한다. 단, 이러한 수정으로 의료기기의 사용목적이나 근본적인 과학기술이 변경되어서는 안된다. 특별 510(k)에서, 제출문서는 이 지침에 따라야 하지만 신고의 원인이 된 수정과 관련된 문서만을 제출하면 된다. 예를 들어 특별 510(k)에서의 요구사항 및 규격 제출문서는 수정된 부분에 초점을 맞추며, 의료기기의 모든 요구사항과 규격을 포함시킬 필요는 없다.

수정을 검증, 유효성 확인하기 위하여 수행하는 회귀시험(regression testing)을 제출하도록 권한다. 시험 자료보다도 오히려 시험 계획, 합부판정기준 및 결과 요약서를 제출하도록 권한다. 어떠한 경우라도 제시할 소프트웨어 관련 문서의 형태 및 세부적 수준은 수정과 관련된 의료기기 심각도 에 적절하여야 한다. 특별 510(k) 신고는 설계관리에 대한 자사의 적합성 선언(declaration of conformance)에 좌우되므로, 시험 또는 선언에 따른 기타 활동을 완료하기 전까지는 특별 510(k) 신고를 적절히 제출할 수 없을 것이다. (연방 FD&C법 514(c)(1)(B)항(21 U.S.C. 360d(c)(1)(B) 참조).

약식(Abbreviated) 510(k) 프로그램

약식 510(k) 신고는 21 CFR 807.87에 명시된 요구 요소(required elements)를 포함하여야 한다. 약식 510(k)의 경우 이 지침에서 권고한 문서 내용은 21 CFR 807.87(f)나 (g)의 의미를 적절하게 입증하는 자료일 것으로 간주한다. 그러므로 이 지침에서 설명한 문서를 제출하도록 권한다.

의료기기 설계나 시험에서 FDA가 인정한 표준(standard)에 따른 경우, 다음 중 하나를 포함하면 된다.

- 제품 시판 전에 시험을 실시하고 시험이 규정된 허용기준(acceptance criteria)을 충족할 것이라는 진술서(statement)
- 기준에 대한 적합성 선언

적합성 선언은 시험 결과에 근거하기 때문에, 표준에서 규정한 시험이 완료되기 전에 적합성 선언을 제출할 수 없다. 세부적인 사항은 법 514(c)(1)(B)항 및 FDA “Use of Standards in Substantial Equivalence Determinations" 지침을 참조하기 바란다.

소프트웨어 기기에 대해 권고되는 특정 시험이나 시험 방법으로 적합성을 선언하는 경우, 적합성 선언과 함께 합부판정기준 및 관련 시험 결과를 같이 제출하도록 권한다. 또한 표준에서 규정된 시험과 시험 방법과의 차이(deviations)를 기록하고, 소프트웨어 기기의 안전성과 유효성에 미치는 영향의 관점에서 이러한 차이를 설명하도록 권한다. FDA가 인정한 합의표준(consensus standards) 목록은 CDRH 웹사이트에 있다.

추가적인 주제(Additional Topics)

위험평가 및 관리

배경

부적절하거나 부적합한 소프트웨어 개발 life cycle 및 위험관리 활동, 소프트웨어 기기의 부적합한 사용 또는 작동상 오류는 다양한 잠재적인 고장이나 설계 결함(flaws)의 결과일 수 있다. 이러한 고장이나 결함에는 안전하지 않거나 유효하지 않은 에너지, 약품 및 생명 보조나 생명 유지 기능의 전달이 포함된다. 또한 오진 또는 잘못된 치료나 요법(therapy)의 선택을 야기하는 부정확, 불완전한 정보의 전달은 특정 소프트웨어 기기와 관련된 잠재적인 고장이다. 따라서 잠재적인 고장이나 설계 결함과 관련된 위험은 소프트웨어 기기의 검토과정에서 관심의 대상이다.

위험 평가 및 심각도

이미 언급한 바와 같이, 소프트웨어 기기와 관련된 위험 평가는 해당 심각도의 결정에 도움이 된다. 또한 동일한 유형(type)이나 사용 목적을 가진 다른 의료기기들의 심각도를 고려하도록 권한다. 자사의 의료기기에 해당하는 심각도 수준이 다르다고 판단한 경우, 그 이론적 근거의 상세 설명을 제출하도록 권한다.

위험관리(Risk Management)

소프트웨어 기기와 관련된 위험은 ‘무시’에서부터 ‘매우 심각’까지 다양하다. 일반적으로 위험은 상해의 심각성과 발생 가능성의 산물로 간주한다. 그러나 소프트웨어 고장은 특성상 시스템적인 사항이며, 이에 따라 발생 가능성은 전형적인 통계적 방법을 사용하여 결정할 수는 없다. 따라서 소프트웨어 기기에 대한 위험 추정(estimation)은 고장 발생으로 야기되는 위험요인의 심각성에 근거하도록 권한다. 또한 ISO 14971과 같은 합의표준에 명기된 위험식별 및 관리기법을 사용하도록 권한다.

소프트웨어 변경관리

소프트웨어 수정판(revisions)의 설계, 개발, 시험 및 버전관리는 시판전 신고에서 검토되었던 소프트웨어의 개발 및 시험만큼이나 중요하다. 소프트웨어 관련 의료기기의 리콜을 포함하여 현장에서 발생하는 소프트웨어 관련 의료기기의 대다수의 문제는 시판전 검토 이후 수정된 소프트웨어를 구현하는 의료기기에서 발생한다는 것이 FDA의 판단이다. FDA의 검토가 요구되지 않는 수정판도 부작용(adverse events)과 리콜을 초래하였다. 이는 소프트웨어의 수정에 세심한 관리가 필요함을 나타내는 것이다.

혈액처리 컴퓨터 소프트웨어(Blood Establishment Computer Software)

혈액처리 컴퓨터 소프트웨어에 대한 시판전 신고의 경우, 모든 제한사항의 설명을 포함해 사용자에게 제공되는 사용자 설명서의 완전한 사본을 제출하여야 한다. 또한 사용자 설명서에 언급하지 않은 모든 중요한 변종(anomalies) 또는 소프트웨어 결함을 적절한 해결책(workarounds)과 함께 사용자게 제공할 경우 그러한 문서도 제출하여야 한다.

근원(Pedigree)을 알 수 없는 소프트웨어(SOUP)

소프트웨어 기기에 포함되는 소프트웨어의 일부 또는 전부는 제3자가 개발한 것일 수 있다. 이러한 소프트웨어에 수반되는 문서의 형태와 품질은 매우 다양할 수 있다. 적절한 문서의 확보(obtain)가 어려운 소프트웨어를 '근원을 알 수 없는 소프트웨어' 또는 SOUP라고 한다.

SOUP에 대한 해당 설계 문서를 확보하거나 만들거나 재구성(reconstruct)하기는 어려울 수 있다. 따라서 소프트웨어의 출처(origin)와 소프트웨어 문서의 주변 상황(circumstances surrounding)을 설명하도록 권한다. 추가적으로 위험요인 분석에는 잘못되거나 불완전한 문서 또는 시험전(prior testing) 문서의 누락과 관련하여 SOUP 연관 위험을 포함하여야 한다. 다만, 의료기기의 적절한 시험 및 적절한 소프트웨어 시험계획과 결론 문서의 제공에 대한 책임은 신고인에게 있다.

바이러스 방지 소프트웨어

소프트웨어 기기를 포함한 정보 시스템을 유해하거나 악의적인 코드(예; 바이러스, 웜 등)로부터 보호하도록 설계된 소프트웨어 응용프로그램은, 의료기기의 상호연결 및 이에 따른 외부정보환경으로의 노출 가능성 증가에 따라, 보다 상용화되고 있다. 바이러스 방지 소프트웨어의 설치 및 시험 관련 문제는 이 지침의 범위를 벗어난다. 이 주제에 관한 세부 사항은 CDRH OC(Office of Compliance)에 문의하기 바란다.

인터페이스, 네트워크 연결(Networking) 및 네트워크 기반구조

이미 언급한 바와 같이 소프트웨어 기기는 특정 의료기기와 특정 데이터의 교환을 위하여 지점간(point-to-point) 인터페이스, 근거리(local)통신망과 광역통신망(wide area networks) 및 인터넷과의 접속으로 상호 연결이 계속 증가하고 있다. 전화, 근거리통신망 및 광대역 연결과 같은 데이터 교환 및 통신 기반구조는 의료기기처럼 규제받지는 않으나 이러한 전송매체(carriers)로 인하여 소프트웨어 기기의 운영에 영향을 미친다. 예를 들면 근거리통신망에 연결된 소프트웨어 기기는 네트워크 인터페이스에 문제가 발생할 경우 원활한 운영이 중단된다. 소프트웨어의 설계에서 의료기기에 제공되는 인터페이스의 용량(capabilities) 및 부하(liabilities) 모두를 고려하도록 권한다. 특히 위험요인 분석과 감소 활동에는 이러한 문제(issues)를 포함하여야 한다.

조합제품(Combination Products)

일반적으로 이 지침의 권고사항은 의료기기 구성부품이 소프트웨어 기기의 정의에 부합할 경우, 조합 제품(예; 의약품-의료기기 및 생물제재-의료기기 조합)의 구성부품에 적용될 것이다. 자세한 사항은 OCP(Office of Combination Products) 또는 조합 제품에 대한 검토를 안내하는 FDA 검토 부서에 문의하기 바란다.

참고문헌

ⅰ. 이 문서는 1998년 5월 29일에 발행된 "의료기기에 포함된 소프트웨어의 시판전 신고 지침"의 대체판 및 1997년 1월 13일 발행된 혈액처리(Blood Establishment) 컴퓨터 소프트웨어의 시판전 신고 지침"에 대하여 재검토한 것을 조합한 것이다.

ii. "General Principles of Software Validation," http://www.fda.gov/cdrh/comp/guidance/938.html

iii. "Guidance for Off-the-Shelf Software Use in Medical Devices" http://www.fda.gov/cdrh/ode/guidance/ 585.pdf

iv. 21 CFR 820.30 Subpart C – Design Controls of the Quality System Regulation

v. ISO 14971-1; Medical devices - Risk management - Part 1: Application of risk analysis

vi. AAMI SW68:2001; Medical device software - Software life cycle processes.

vii. See "The New 510(k) Paradigm – Alternate Approaches to Demonstrating Substantial Equivalence in Premarket Notifications – Final Guidance," available on the FDA Web site at http://www.fda.gov/ cdrh/ode/parad510.html

viii. For more information see Device Advice, "How to Prepare an Abbreviated 510(k)," http://www. fda.gov/cdrh/devadvice/3145.html, in particular the section titled "Information Required in an Abbreviated 510(k)."

ix. See "Required Elements for a Declaration of Conformity to a Recognized Standard (Screening Checklist for All Premarket Notification [510(K)] Submissions)," http://www.fda.gov/cdrh/ode/ reqrecstand.html

x. See "Use of Standards in Substantial Equivalence Determinations," http://www.fda.gov/cdrh/ ode/guidance/1131.html

xi. http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfStandards/search.cfm.

xii. For information on determining when revisions to software should result in a new premarket submission, you should consult the relevant FDA guidances such as "Deciding When to Submit a 510(k) for a Change to an Existing Device," http://www.fda.gov/cdrh/ode/510kmod. html. See also 21 CFR 807.81(a)(3)

[별첨 4]

소프트웨어 수명주기 프로세스(IEC 62304/FDiS(2006))
(Medical device software – Software life cycle processes)

서 문

의료기기 기술에 소프트웨어가 중요한 부분이 되는 경우가 많다. 소프트웨어가 포함된 의료기기의 안전성과 효과성 확립에는 소프트웨어 사용 의도 및 소프트웨어 사용에 의해 허용할 수 없는 위험의 발생 없이 그러한 의도가 충족된다는 사실의 증명이 필요하다.

이 규격은 의료기기 소프트웨어의 안전한 설계와 유지보수에 필요한 활동과 업무에 관련된 수명주기 프로세스의 체제를 제공한다. 이 규격은 각 수명주기 프로세스에 대한 요구사항을 제공한다. 각 수명주기 프로세스는 일련의 활동으로 다시 분류되며, 각 활동은 일련의 업무로 분류된다.

기본적으로 의료기기 소프트웨어는 품질경영시스템(4.1 참조)과 위험관리시스템(4.2 참조) 범위 안에서 개발•유지된다. 위험관리 프로세스는 이미 국제규격 ISO 14971에 규정되어 있다. 따라서 IEC 62304는 단순히 ISO 14971 규정을 참조함으로써 이 장점을 활용한다. 특히 위험요인에 관련해 영향을 주는 소프트웨어 요인의 식별 영역의 경우 일부 사소한 위기경영 요구사항이 소프트웨어에 추가로 필요하다. 이러한 요구사항은 제7장에 IEC 위험관리 프로세스에 요약 설명되어 있다.

소프트웨어가 위험요인에 대한 기여 요소인지의 여부는 위험관리 프로세스의 위험요인 식별 활동 중 판단된다. 소프트웨어에 의해 간접적으로 발생할 수 있는 위험요인(예; 잘못된 정보를 제공해 치료가 부적절하게 수행되는 경우)은 소프트웨어가 발생 요인인지 여부를 결정할 때 고려하여야 한다. 위험 제어에 소프트웨어를 사용할 것인지의 여부는 위험관리 프로세스의 위험제어 활동 중 결정된다. 이 표준에 필요한 소프트웨어 위험관리 프로세스는 ISO 14971에 따라 의료기기 위험관리 프로세스에 내재되어야 한다.

소프트웨어 개발 프로세스는 몇 가지 활동으로 구성된다. 이들 활동은 그림 1에 표시되어 있으며, 제5조에 설명된다. 현장에서 발생하는 대부분의 사고는 부적합한 소프트웨어 갱신과 업그레이드를 포함해 의료기기 시스템의 정비나 유지보수에 관련되므로 소프트웨어 유지관리 프로세스는 소프트웨어 개발 프로세스만큼 중요하게 취급해야 한다. 소프트웨어 유지보수 프로세스는 소프트웨어 개발 프로세스와 매우 유사하다. 이 프로세스는 그림 2에 표시되어 있으며, 제6조에 설명된다.

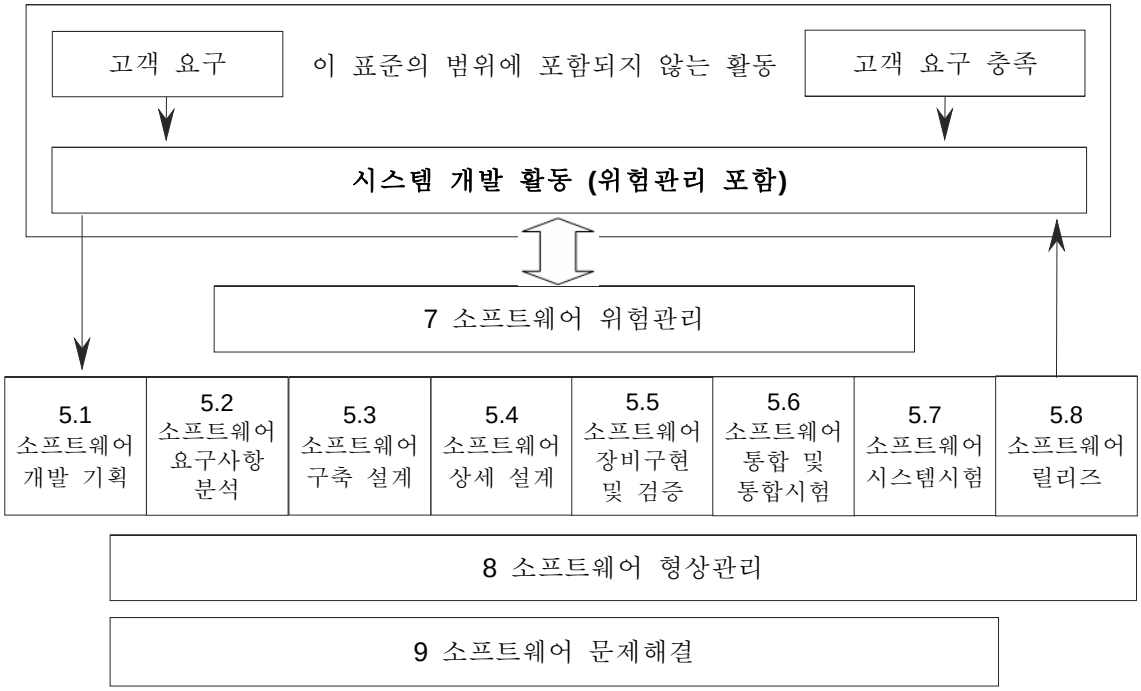
이 표준은 안전한 의료기기 소프트웨어 개발에 필수적인 것으로 간주되는 두 가지 추가 프로세스를 식별한다. 즉, 형상관리 프로세스(제8조) 및 소프트웨어 문제해결 프로세스(제9조)가 그것이다.

이 표준은 제조업체에 대한 조직 구조 또는 조직의 어떤 부서가 어떤 프로세스, 활동 또는 업무를 수행할 것인지 규정하지 않는다. 이 표준에는 이 표준의 준수성을 확보할 수 있도록 프로세스, 활동 또는 업무가 완료되어야 한다는 사실만 요구된다.

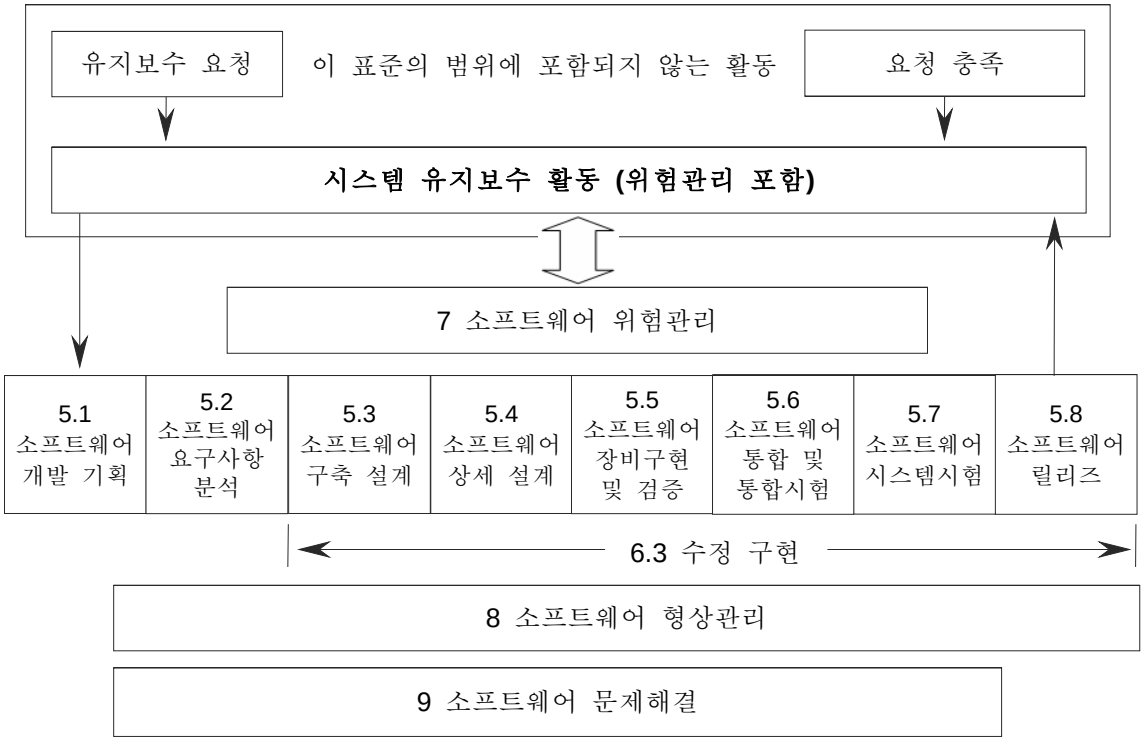
이 표준은 제작할 문서의 명칭, 형식 또는 명확한 내용을 규정하지 않는다. 이 표준에 따라 업무의 문서화가 필요하지만, 이 문서를 구성하는 방법은 표준의 사용자가 결정한다.

이 표준은 특정 수명주기 모델을 규정하지 않는다. 이 표준 사용자는 소프트웨어 프로젝트에 대한 수명주기 모델을 선택하고, 그 모델에 이 표준의 프로세스, 활동 및 업무를 매핑할 책임이 있다.

부록 A에는 이 표준의 조항에 대한 이론적 근거가 제공된다. 부록 B에는 이 표준의 규정에 대한 지침이 제공된다.



[그림 1] 소프트웨어 개발 프로세스 및 활동의 개요



[그림 2] 소프트웨어 유지보수 프로세스 및 활동의 개요

이 표준의 목적을 위해 아래 용어는 다음과 같이 정의된다.

- “shall”은 요구사항의 준수가 표준 준수에 의무적임을 의미한다.
- “should”은 요구사항의 준수가 표준 준수에 권장되지만 의무적은 아님을 의미한

다.

- “may”는 요구사항 준수를 위해 허용되는 방법의 설명에 사용된다.
- “establish”는 정의, 문서화 및 구현을 의미한다.
- 이 표준에서 “as appropriate”라는 용어를 요구되는 프로세스, 활동, 임무 또는 산출과 연계해 사용할 경우 그 사용 목적은 제조업체가 그러한 프로세스, 활동, 업무 또는 산출을 사용하지 않아도 되는 근거를 문서화할 수 없는한 제조업체가 그러한 프로세스, 활동, 업무 또는 산출을 사용해야 한다는 것이다.

1. 범위

1.1 목적

이 표준에는 의료기기 소프트웨어에 대한 수명주기 요구사항이 정의된다. 이 표준에 설명한 프로세스, 활동 및 업무는 의료기기 소프트웨어 수명주기 프로세스에 대한 공통적인 체제를 확립한다.

1.2 적용분야

이 표준은 의료기기 소프트웨어 개발과 유지보수에 적용된다.

이 표준은 소프트웨어 자체가 의료기기이거나, 소프트웨어가 최종 의료기기에 내장되거나 구성 부분인 경우 의료기기 소프트웨어 개발과 유지보수에 적용된다.

이 표준은 의료기기가 소프트웨어로만 구성된 경우에도 의료기기의 밸리데이션과 최종 릴리스는 취급하지 않는다.

1.3 다른 표준과의 관계

이 의료기기 소프트웨어 수명주기 표준은 의료기기를 개발할 때 다른 해당 표준과 함께 사용된다. 부록 C에는 이 표준 및 기타 관련 표준과의 관계가 표시되어 있다.

1.4 준수성

이 표준의 준수성은 소프트웨어 안전 등급에 따라 이 표준에서 확인(identified)된 모든 프로세스, 활동 및 업무를 구현하는 것으로 정의된다.

주 : 각 요구사항에 할당된 소프트웨어 안전 등급은 요구사항 뒤의 규정 텍스트에 설명된다.

준수성은 위험관리 파일 및 소프트웨어 안전 등급에 필요한 프로세스, 활동 및 업무의 평가를 포함해, 이 표준이 요구하는 모든 문서화의 검사에 의해 결정된다. 부록 D 참조.

주 1 : 이 평가는 내부감사나 외부감사로 수행할 수 있다.
주 2 : 지정 프로세스, 활동 및 업무를 수행하는 경우에도 그러한 프로세스의 구현과 그러한 활동과 업무 수행의 방법에는 융통성이 있다.
주 3 : 요구사항에 "as appropriate"라는 용어가 포함되어 있지만 수행되지 않은 경우 이 평가에는 그 정당화에 대한 문서화가 필요하다.
주 4 : 일치성이라는 용어는 ISO/IEC 12207에 사용되며, 이는 이 표준에 사용되는 준수성이라는 용어에 해당된다.

2. 규격 참고문헌

아래에 인용한 문서는 이 문서의 적용에 필수불가결이다. 날짜가 명시된 참고문헌의 경우 해당 판만 적용된다. 일자가 명기되지 않은 참고문헌의 경우 인용한 문서 최신판(수정 포함)이 적용된다.
ISO 14971, 의료기기 - 의료기기 위험관리의 적용

3. 용어 정의

이 문서에는 다음과 같은 용어와 정의가 적용된다.

3.1 활동(ACTIVITY)

하나 이상의 상호관련 또는 상호작용 업무

3.2 비정상 상태(ANOMALY)

요구사항 시방서, 설계문서, 표준 등에 기초한 예상치 또는 인식이나 경험에서 벗어나는 조건. 비정상 상태는 소프트웨어 제품 또는 해당 문서의 검토, 시험, 분석, 작성 또는 사용 중 확인(be found)할 수 있다. [IEEE 1044:1993, 정의 3.1]

3.3 아키텍처(ARCHITECTURE) :

시스템이나 구성부품의 조직 구조. [IEEE 610.12:1990]

3.4 변경 요청(CHANGE REQUEST)

소프트웨어 제품에 이루어지는 변경을 문서화한 시방서

3.5 형상 항목(CONFIGURATION ITEM)

특정 기준점에서 고유하게 확인(identified)할 수 있는 실체

주 : ISO/IEC 12207:1995, 정의 3.6에 기초.

3.6 산출물(DELIVERABLE)

활동이나 업무에서 요구되는 결과 또는 산출(문서화 포함)

3.7 평가(EVALUATION)

실체가 지정 기준을 충족하는 범위에 대한 체계적인 판단. [ISO/IEC 12207:1995, 정의 3.9]

3.8 위해(HARM)

사람의 보건에 대한 신체적 상해나 피해 또는 양쪽 모두. 또는 재산이나 환경에 대한 피해 [ISO/IEC Guide 51:1999, 정의 3.1]

3.9 위험요인(HAZARD)

위해의 잠재적 발생원 [ISO/IEC Guide 51:1999, 정의 3.5]

3.10 제조업체(MANUFACTURER)

의료기기의 설계, 제조, 포장 또는 라벨링, 시스템 조립 또는 해당 관계인이나 해당 관계인을 대리한 제3자가 운영을 수행하는지의 여부에 관계 없이 시판 및 사용 전 의료기기의 채택을 담당하는 자연인이나 법인. [ISO 14971:2000, 정의 2.6]

3.11 의료기기(MEDICAL DEVICE)

다음과 같은 특정 목적 중 하나 또는 그 이상을 위해 인간에 대해 단독으로 또는 조합으로 사용되는, 제조업체의 기구, 장비, 도구, 기계, 기구, 이식 조직편, 체외 시약 또는 측정기, 소프트웨어, 자재 또는 기타 유사하거나 관련된 품목

- 질병의 진단, 예방, 모니터링, 치료 또는 경감
- 상해의 진단, 모니터링, 치료, 경감 또는 보상
- 비정상 상태 또는 생리학적 프로세스의 조사, 대체, 수정 또는 지지
- 생명 지지 또는 유지

- 임신관리
- 의료기기의 소독
- 인체에서 채취한 표본의 체외 검사를 사용해 의료 목적을 위한 정보 제공 또한 약리학, 면역학 또는 신진대사적 수단으로 인체에 의도된 주된 조치를 달성하는 것이 아니라 기능이 그러한 수단의 지원을 받을 수 있는 것

주 1 : 이 정의는 의료기기 국제조화기기(GHTF)가 개발한 것이다. 참고문헌[15] (ISO 13485:2003) 참조. [ISO 13485:2003, 정의 3.7]

주 2 : 각국의 법규에서 사용하는 정의에 어느 정도 차이가 발생할 수 있다.

3.12 의료기기 소프트웨어(MEDICAL DEVICE SOFTWARE)

개발 중인 의료기기에 채택할 목적으로 개발된 소프트웨어 시스템 또는 자체 권한으로 의료기기로서 사용하기 위한 소프트웨어 시스템

3.13 문제 보고서(PROBLEM REPORT)

사용자나 다른 이해관계 당사자가 의도된 사용에 불안전하고 적절하지 않거나 시방서와 다르다고 생각하는 소프트웨어 제품의 실제 또는 잠재적 작동상태의 기록

주 1 : 이 표준에 다르면 모든 문제 보고서에 따라 소프트웨어 제품을 변경할 필요는 없다. 제조업체는 문제 보고서를 오해, 오류 또는 사소한 사상(事象)으로 간주해 배척할 수 있다.

주 2 : 문제 보고서는 이미 릴리즈된 소프트웨어 제품 도는 현재 개발 중인 소프트웨어 제품에 관련될 수 있다.

주 3 : 이 표준에 따라 규제 조치를 식별하고 구현할 수 있도록 이미 릴리즈된 제품에 관련된 문제 보고서에 대해 추가 의사결정 단계(제6조 참조)를 수행해야 한다.

3.14 프로세스(PROCESS)

입력을 출력으로 변환하는 일련의 상호관련 또는 상호작용 활동 [ISO 9000:2000, 정의 3.4.1]

주 : "활동"이란 용어에는 자원의 사용도 포함된다.

3.15 회귀시험(REGRESSION TESTING)

시스템 구성부품에 대한 변경이 기능성, 신뢰성 또는 성능에 부정적인 영향을 미치지 않으며 추가 결함이 발생하지 않았다고 판단하기에 필요한 시험 [ISO/IEC 90003:2004, 정의 3.11]

3.16 위험(RISK)

위해 및 위해 심각도 발생 가능성의 조합 [ISO/IEC Guide 51:1999 정의 3.2]

3.17 위험 분석(RISK ANALYSIS)

위해를 식별하고 위험을 추정하기 위한 가용 정보의 체계적인 사용 [ISO/IEC Guide 51:1999 정의 3.10]

3.18 위험관리(RISK CONTROL)

의사결정이 이루어지며 위험이 지정 수준으로 감소하거나 지정 수준 안에 유지되는 프로세스 [ISO 14971:2000 정의 2.16]

3.19 위험관리(RISK MANAGEMENT)

위험 분석, 평가 및 관리 업무에 대한 경영정책, 절차 및 방법의 체계적인 적용[ISO 14971:2000 정의 2.18]

3.20 위험관리 파일(RISK MANAGEMENT FILE)

위험관리 프로세스에서 생성되는 기록과 기타 문서. 단, 연속적일 필요는 없다. [ISO 14971:2000 정의 2.19]

3.21 안전성(SAFETY)

허용할 수 없는 위험의 부재 [ISO/IEC Guide 51:1999 정의 3.1]

3.22 보안(SECURITY)

미승인 관계자나 시스템이 정보와 데이터를 읽을 수 없게 만들며, 승인 관계자나 시스템이 정보와 데이터에 대한 액세스가 거부되지 않도록 정보와 데이터를 보호하는 활동. [ISO/IEC 12207:1995 정의 3.25]

3.23 중상(SERIOUS INJURY)

직접적 또는 간접적으로 다음과 같은 부상이나 질병

a) 생명을 위협한다.
b) 신체 기능에 대한 영구적 훼손이나 신체 구조에 대한 영구적 피해가 발생한다.
c) 신체 기능에 대한 영구적 훼손이나 신체 구조에 대한 영구적 피해 배제에 의료적 또는 수술적 개입이 필요하다.

주 : 영구적 훼손은 사소한 훼손이나 피해를 제외하고, 신체 구조나 기능에 대한 회복 불가능한 훼손이나 피해를 의미한다.

3.24 소프트웨어 개발 수명주기 모델(SOFTWARE DEVELOPMENT LIFE CYCLE MODEL)

요구사항의 정의에서부터 제조를 위한 릴리즈에 이르기까지 소프트웨어의 수명 전체를 포괄하는 개념적 구조로서,

- 소프트웨어 제품 개발에 관련된 프로세스, 활동 및 업무를 확인(identify)한다.
- 활동과 업무 사이의 순서와 의존성을 설명한다.
- 특정 산출물의 완벽성을 검증할 수 있는 일정을 확인(identify)한다.

주 : ISO/IEC 12207:1995, 정의 3.11에 기초.

3.25 소프트웨어 항목(SOFTWARE ITEM)

컴퓨터 프로그램의 확인 가능한 부분(identifiable part)[ISO/IEC 90003:2004, 정의 3.14]

주 : 세 가지 용어가 소프트웨어 분해(decomposition)를 구분한다. 최상위 수준(Level)은 소프트웨어 시스템이다. 더 이상 분해할 수 없는 최하위 수준이 소프트웨어 유닛이다. 최상위와 최하위가 포함된 모든 수준의 구성을 소프트웨어 항목(item)이라고 부를 수 있다. 즉, 소프트웨어 항목은 하나 이상의 소프트웨어 항목으로 구성되며, 각 소프트웨어 항목은 하나 이상의 소프트웨어 유닛이나 분해 가능한 소프트웨어 항목으로 구성된다. 소프트웨어 항목과 소프트웨어 유닛의 정의(definition)와 입도(granularity)를 제공하는 것은 제조업체의 책임이다.

3.26 소프트웨어 제품(SOFTWARE PRODUCT)

컴퓨터 프로그램, 절차 및 연관 문서와 데이터 [SIO/IEC 12207:1995 정의 3.26]

3.27 소프트웨어 시스템(SOFTWARE SYSTEM)

특정 기능이나 일련의 기능을 수행하도록 구성된 소프트웨어 항목(item)을 통합한 집합체

3.28 소프트웨어 유닛(SOFTWARE UNIT)

다른 항목(item)으로 다시 분류할 수 없는 소프트웨어 항목(item)

주 : 소프트웨어 유닛은 소프트웨어 형상관리 또는 시험 목적에 사용할 수 있다.

3.29 기원이 확인되지 않은 소프트웨어(SOUP; Software Of Unknown Provenance (acronym))

이미 개발되어 일반적으로 사용할 수 있으며, 의료기기에 채택할 목적으로 개발되지 않은 소프트웨어 항목(item)(“기성품 소프트웨어”라고도 함) 또는 개발 프로세스에 대한 적절한 기록이 없는, 이전에 개발된 소프트웨어

3.30 시스템(SYSTEM)

하나 이상의 프로세스, 하드웨어, 소프트웨어, 시설 및 명시한 요구나 목적을 충족시킬 수 있는 성능을 제공하는 사람으로 통합 구성된 합성체. [ISO/IEC 12207:1995, 정의 3.31]

3.31 업무(TASK)

완수해야 할 작업의 한 부분

3.32 추적성(TRACEABILITY)

개발 프로세스의 두 개 이상 제품 사이에 확립할 수 있는 관계의 정도 [IEEE 610.12:1990]

3.33 검증(VERIFICATION)

특정 요구사항이 충족되었다는 객관적 증거의 제공을 통해 확인(confirmation)하는 것

주 1 : 'Verified(검증된)'이라는 용어는 해당 상태의 지정에 사용된다. [ISO 9000:2000, 정의 3.8.4]

주 2 : 설계와 개발의 경우 검증은 일반적으로 특정 활동에 대해 명시한 요구사항과의 적합성을 판단하기 위해 그러한 활동의 결과를 조사하는 프로세스와 연관된다.

3.3.4 버전(VERSION)

형상 항목(item)의 식별된 인스턴스(instance).

주 1 : 새 버전이 작성되며 형상관리 조치가 필요한, 소프트웨어 제품 버전의 수정

주 2 : ISO/IEC 12207:1995, 정의 3.37에 기초.

4. 일반 요구사항

4.1 품질경영시스템

의료기기 소프트웨어 제조업체는 일관성 있게 고객 요구사항 및 해당 규제적 요구사항을 충족하는 의료기기 소프트웨어를 제공할 수 있는 능력을 입증해야 한다.

주 1 : 이 능력은 다음을 준수하는 품질경영시스템의 사용으로 입증할 수 있다.

- ISO 13485 [7] 또는
- 국가 품질경영시스템 규격, 또는
- 국가 법규에 따라 요구되는 품질경영시스템

주 2 : 소프트웨어에 대한 품질경영시스템 요구사항 적용에 관한 지침은 ISO/IEC 90003 [11]에 수록되어 있다.

4.2 위험관리

제조업체는 ISO 14971을 준수하는 위험관리 프로세스를 적용해야 한다.

4.3 소프트웨어 안전성 분류

a) 제조업체는 소프트웨어 시스템에서 발생하며, 환자, 작업자 또는 기타 사람에 대한 위해의 영향에 따라 각 소프트웨어 시스템에 소프트웨어 안전성 등급을 할당해야 한다(A, B 또는 C).

소프트웨어 안전성 등급은 초기에는 다음과 같은 심각도에 기초해 할당된다.

- 등급 A: 부상이나 건강 피해가 없다.
- 등급 B: 중상이 가능하지 않다.
- 등급 C: 사망이나 중상이 가능하다.

소프트웨어 시스템이 지정한대로 작동하지 않아 위해가 발생할 경우 그러한 고장의 발생 가능성은 100퍼센트로 간주되어야 한다.

b) 소프트웨어 고장에서 발생하는 사망이나 중상의 위험을 고장의 결과를 감소시키거나 고장에서 발생하는 사망이나 중상의 발생 가능성을 감소함으로써 하드웨어 위험관리 방법으로 허용할 수 있는 수준(ISO 14971에 정의)으로 축소할 수 있는 경우 소프트웨어 안전성 분류는 C에서 B로 축소할 수 있다. 또한 소프트웨어 고장에서 발생하는 비 중상의 위험이 하드웨어 위험관리 방법으로 유사하게 허용 가능한 수준으로 감소될 경우 소프트웨어 안전성 분류는 B에서 A로 축소할 수 있다. 제조업체는 위험관리 방법이 관리 중인 영향에 기초한 소프트웨어 안전성 등급을 위험관리 방법의 구현에 기여하는 각 소프트웨어 시스템에 할당해야 한다.

c) 제조업체는 각 소프트웨어 시스템에 할당된 소프트웨어 안전성 등급을 위험관리 파일에 문서화해야 한다.

d) 소프트웨어 시스템을 소프트웨어 항목(item)으로 분해하고 소프트웨어 항목(item)을 다시 다른 소프트웨어 항목(item)으로 분해하면 제조업체가 다른 소프트웨어 안전성 등급으로의 분류에 대한 이론적 근거를 문서화하지 않는 한 그러한 소프트웨어 항목(item)은 원래 소프트웨어 항목(item)(또는 소프트웨어 시스템)의 소프트웨어 안전성 분류를 계승한다. 그러한 이론적 근거에는 새로운 소프트웨어 항목(item)이 별도로 분류될 수 있도록 분리되는 방법의 설명이 포함되어야 한다.

e) 제조업체는 소프트웨어 안전성 등급이 분해에 의해 생성된 소프트웨어 항목(item) 등급과 다를 경우 각 소프트웨어 항목(item)의 소프트웨어 안전성 등급을 문서화해야 한다.

f) 이 표준의 준수성을 위해 특정 분류의 소프트웨어 항목(item)에 프로세스가 필요하거나 소프트웨어 항목(item) 그룹에 프로세스를 적용해야 할 때마다 제조업체가 낮은 수준의 분류 사용에 대한 이론적 근거를 위험관리 파일에 문서화하지 않는 한 제조업체는 그룹 중 가장 상위로 분류된 프로세스와 업무를 사용해야 한다.

g) 각 소프트웨어 시스템의 경우 소프트웨어 안전성 등급이 할당될 때까지 등급 C 요구사항이 적용된다.

주 : 뒤따르는 요구사항에서 요구사항을 수행해야 하는 소프트웨어 안전성 등급은 [등급 …]의 형식으로 요구사항 뒤에 식별된다.

5. 소프트웨어 개발 프로세스

5.1 소프트웨어 개발 기획

5.1.1 소프트웨어 개발 계획

제조업체는 개발 대상 소프트웨어 시스템의 범위, 규모 및 소프트웨어 안전성 분류에 적합한 소프트웨어 개발 프로세스의 활동 진행을 위한 소프트웨어 개발 계획을 확립해야 한다. 소프트웨어 개발 수명주기 모델을 계획에 완벽히 정의하거나 참조해야 한다. 계획은 다음의 내용을 취급해야 한다.

a) 소프트웨어 시스템의 개발에 사용되는 프로세스(주 4 참조)
b) 활동 및 업무의 산출물(문서화 포함)
c) 시스템 요구사항, 소프트웨어 요구사항, 소프트웨어 시스템 시험 및 소프트웨어에 구현되는 위험관리 방법 사이의 추적성
d) SOUP 형상 항목(item) 및 개발 지원에 사용되는 소프트웨어를 포함하여 소프트웨어 형상 및 변경관리
e) 수명주기의 각 단계에서 소프트웨어 제품, 산출물 및 활동 중 확인(detected)되는 문제 취급을 위한 소프트웨어 문제해결 [등급 A, B, C]

주 1 : 소프트웨어 개발 수명주기 모델은 소프트웨어 시스템의 각 소프트웨어 항목(item)에 대한 소프트웨어 안전성 분류에 따라 서로 다른 소프트웨어 항목(item)에 대해 서로 다른 요소(프로세스, 활동, 업무 및 산출물)를 확인(identify)할 수 있다.

주 2 : 이 활동과 업무는 중복되거나 상호 작용할 수 있으며, 반복적으로 또는 귀납적으로 수행할 수 있다. 단, 특정 수명주기 모델을 사용해야 한다고 암시하는 것이 아니다.

주 3 : 다른 프로세스는 개발 프로세스와는 별도로 이 표준에 설명된다. 단, 다른 프로세스가 별도 활동과 업무로서 구현되어야 한다고 암시하는 것은 아니다. 다른 프로세스의 활동과 업무는 개발 프로세스에 통합할 수 있다.

주 4 : 소프트웨어 개발 계획은 기존 프로세스를 참조하거나 새 프로세스를 정의할 수 있다.

주 5 : 소프트웨어 개발 계획은 전체 시스템 개발 계획에 통합할 수 있다.

5.1.2 갱신된 소프트웨어 개발 계획

제조업체는 개발이 진행되면 그에 상응하게 계획을 갱신해야 한다. [등급 A, B, C]

5.1.3 시스템 설계와 개발에 대한 소프트웨어 개발 계획 참조

a) 소프트웨어 요구사항은 소프트웨어 개발을 위한 입력으로서 제조업체가 소프트웨어 개발계획에 참조해야 한다.

b) 제조업체는 4.1의 충족을 위하여 소프트웨어 개발과 설계 및 개발 유효성 확인을 일원화하기 위해 소프트웨어 개발 계획 절차를 포함시키거나 참조해야 한다. [등급 A, B, C]

주 : 소프트웨어 시스템이 자립형 시스템(소프트웨어로만 구성된 장비)인 경우 소프트웨어 시스템 요구사항과 시스템 요구사항에는 차이가 있을 수 있다.

5.1.4 소프트웨어 개발 표준, 방법 및 도구 기획

제조업체는 소프트웨어 개발 계획에 다음 사항을 포함시키거나 참조해야 한다.

a) 표준

b) 방법

c) 도구

이들 사항은 등급 C의 소프트웨어 항목(item) 개발에 관련된다. [등급 C]

5.1.5 소프트웨어 통합 및 통합 시험 기획

제조업체는 소프트웨어 항목(item)(SOUP 포함)을 통합하고 통합 중 시험을 수행하기 위한 계획을 소프트웨어 개발 계획에 포함시키거나 참조해야 한다.[등급 B, C]

주 : 통합 시험 및 소프트웨어 시스템 시험을 단일 계획 및 활동으로 결합할 수 있다.

5.1.6 소프트웨어 검증 기획

제조업체는 소프트웨어 개발 계획에 다음과 같은 검증 정보를 포함시키거나 참조해야 한다. [등급 A, B, C]

a) 검증이 필요한 산출물

b) 각 수명주기 활동에 요구되는 검증업무

c) 산출물을 검증하는 일정

d) 산출물 검증 인수 기준

5.1.7 소프트웨어 위험관리 기획

제조업체는 SOUP에 관련된 위험의 경영을 포함해 소프트웨어 위험관리 프로세스의 활동과 업무 수행을 위한 계획을 소프트웨어 개발 계획에 포함시키거나 참조해야 한다. [등급 A, B, C]

주 : 제7조 참조

5.1.8 문서화 기획

제조업체는 소프트웨어 개발 수명주기 중 생산되는 문서에 관한 정보를 소프트웨어 개발 계획에 포함시키거나 참조해야 한다. 식별된 각 문서나 문서 유형에 있어 다음과 같은 정보를 포함시키거나 참조해야 한다. [등급 A, B, C]

a) 제목, 명칭 및 명칭 표기법

b) 목적

c) 문서의 사용 대상자

d) 개발, 검토, 승인 및 수정을 위한 절차와 책임

5.1.9 소프트웨어 형상관리 기획

제조업체는 소프트웨어 개발 계획에 소프트웨어 형상관리 정보를 포함시키거나 참조해야 한다. 소프트웨어 형상관리 정보에 다음 내용을 포함시키거나 참조해야 한다. [등급 A, B, C]

a) 관리 대상 항목(item)의 등급, 유형, 범주 또는 목록

b) 소프트웨어 형상관리 활동 및 업무

c) 소프트웨어 형상관리와 활동의 수행을 담당하는 조직

d) 소프트웨어 개발 또는 유지보수와 같은 다른 조직과의 관계

e) 항목(item)을 형상관리 대상으로 정하는 시기

f) 문제해결 프로세스를 사용할 시기

5.1.10 관리할 지원 항목(item)

관리할 항목(item)에는 의료기기 소프트웨어 개발에 사용되며 의료기기 소프트웨어에 영향을 줄 수 있는 도구, 항목(item) 및 설정이 포함된다. [등급 B, C]

주 : 그러한 항목의 예에는 컴파일러/어셈블러 버전, 제작 파일, 배치 파일 및 특정 환경 설정이 포함된다.

5.1.11 검증 전 소프트웨어 형상 항목(item) 관리

제조업체는 형상 항목(item)을 검증하기 전 문서화된 형상관리 제어 아래에 형상 항목(item)을 배치하도록 계획해야 한다. [등급 B, C]

5.2 소프트웨어 요구사항 분석

5.2.1 시스템 요구사항에서 소프트웨어 요구사항을 정의하고 문서화

의료기기에 사용되는 각 소프트웨어 시스템의 경우 제조업체는 시스템 차원 요구사항에서 소프트웨어 시스템 요구사항을 정의하고 문서화해야 한다.[등급 A, B, C]

주 : 소프트웨어 시스템이 자립형 시스템(소프트웨어로만 구성된 장비)인 경우 소프트웨어 시스템 요구사항과 시스템 요구사항에는 차이가 있을 수 있다.

5.2.2 소프트웨어 요구사항 내용

의료기기 소프트웨어에 적합하도록 제조업체는 소프트웨어 요구사항에 다음을 포함시켜야 한다.[등급 A, B, C]

a) 기능 및 성능 요구사항

주 1 : 예에는 다음이 포함된다.

- 성능(예를 들어 소프트웨어 목적, 시점 요구사항)
- 물리적 특성(예를 들어 코드 언어, 플랫폼, 운영체제)
- 소프트웨어가 수행되는 전산환경(예를 들어 하드웨어, 메모리 크기, 처리 장치, 시간대, 네트워크 기반구조)
- 업그레이드나 복수 SOUP 또는 기타 장비 버전과의 호환성 요구

b) 소프트웨어 시스템 입력과 출력

주 2 : 예에는 다음이 포함된다.

- 데이터 특성(예를 들어 숫자, 영숫자, 형식)
- 범위
- 한계
- 기본값

c) 소프트웨어 시스템과 다른 소프트웨어 사이의 연계성

d) 소프트웨어 구동 경보, 경고 및 운영자 메시지

e) 보안 요구사항

주 3 : 예에는 다음이 포함된다.

- 중요한 정보의 노출 위험에 관련되는 요구사항
- 정체확인(authentication)
- 인가
- 심사추적
- 통신 무결성

f) 사람에 의해 발생하는 오류와 훈련에 민감한 활용성 공학 요구사항

주 4 : 예에는 다음에 관련된 요구사항이 포함된다.

- 수동 운영 지원
- 인간-장치 상호작용
- 담당자에 대한 제약사항
- 사람의 주의 집중이 필요한 영역

주 5 : 활용성 공학 요구사항에 관한 정보는 IEC 60601-1-6에 수록되어 있다.

g) 데이터 정의 및 데이터베이스 요구사항

주 6 : 예에는 다음이 포함된다.

- 형태
- 적응성
- 기능

h) 운영 및 유지보수 현장에 인도된 의료기기 소프트웨어의 설치 및 인수 요구사항

i) 운영 및 유지보수 방법에 관련된 요구사항

j) 개발할 사용자 설명서

k) 사용자 유지보수 요구사항

l) 규제적 요구사항

주 7 : 이러한 모든 요구사항은 소프트웨어 개발을 시작할 때에는 사용하지 못할 수 있다.

주 8 : ISO/IEC 9126-1[8]에는 소프트웨어 요구사항 정의에 유용할 수 있는 품질 특성에 관한 정보가 수록되어 있다.

5.2.3 소프트웨어 요구사항에 위험관리 방법 포함

제조업체는 하드웨어 고장 및 잠재적 소프트웨어 결함에 대해 소프트웨어에 구현한 위험관리 방법을 의료기기 소프트웨어에 적합한 요구사항에 포함시켜야 한다. [등급 B, C]

주 : 이러한 요구사항은 소프트웨어 개발을 시작할 때 사용할 수 없을 수 있으며, 소프트웨어가 설계되고 위험관리 방법이 계속 정의되는 동안 변할 수 있다.

5.2.4 의료기기 위험분석의 재평가

제조업체는 소프트웨어 요구사항이 확립되고 갱신되면 의료기기 위험평가를 재평가해야 한다. [등급 A, B, C]

5.2.5 시스템 요구사항 갱신

제조업체는 시스템 요구사항을 포함해 기존의 요구사항이 소프트웨어 요구사항 분석 활동의 결과에 적합하게 재평가되고 갱신되는지 확인(ensure)해야 한다. [등급 A, B, C]

5.2.6 소프트웨어 요구사항 검증

제조업체는 소프트웨어 요구사항이 다음과 같은지 검증하고 문서화해야 한다. [등급 A, B, C]

a) 위험관리에 관련된 요구사항을 포함해 시스템 요구사항을 구현한다.

b) 서로 상충하지 않는다.

c) 모호한 표현을 방지하는 용어로 표현된다.

d) 시험 기준이 충족되는지 판단하기 위해 시험 기준과 시험 성능을 확립할 수 있는 용어로 언급된다.

e) 서로 다르게 식별할 수 있다.

f) 시스템 요구사항이나 다른 발생원까지 추적할 수 있다.

주 : 이 표준에 따르면 공식 시방서 언어를 사용할 필요는 없다.

5.3 소프트웨어 구축 설계

5.3.1 소프트웨어 요구사항을 아키텍처로 변형

제조업체는 의료기기 소프트웨어에 대한 요구사항을 소프트웨어 구조를 설명하고 소프트웨어 항목을 식별하는 문서화된 아키텍처로 변환해야 한다. [등급 B, C]

5.3.2 소프트웨어 항목의 연계성을 이한 아키텍처 개발

제조업체는 소프트웨어 항목과 소프트웨어 항목 외부의 구성요인(소프트웨어와 하드웨어) 사이의 연계성과 소프트웨어 항목 사이의 연계성을 위한 아키텍처를 개발하고 문서화해야 한다. [등급 B, C]

5.3.3 SOUP 항목의 기능 및 성능 요구사항 명시

소프트웨어 항목이 SOUP로 식별된 경우 제조업체는 의도한 사용에 필요한 SOUP 항목에 대한 기능 및 성능 요구사항을 명시해야 한다. [등급 B, C]

5.3.4 SOUP 항목에 필요한 시스템 하드웨어와 소프트웨어 명시

소프트웨어 항목이 SOUP로 식별된 경우 제조업체는 SOUP 항목의 적절한 운영 지원에 필요한 시스템 하드웨어와 소프트웨어를 명시해야 한다. [등급 B, C]

주 : 예에는 프로세서 유형과 속도, 메모리 유형과 크기, 시스템 소프트웨어 유형, 통신 및 디스플레이 소프트웨어 요구사항이 포함된다.

5.3.5 위험관리에 필요한 분리 식별

제조업체는 위험관리에 필수적인 소프트웨어 항목 사이의 분리를 식별하고 분리가 유효한지 확인(ensure)하는 방법을 명시해야 한다. [등급 C]

주 : 분리의 한 예는 소프트웨어 항목이 다른 프로세서에서 실행되도록 만드는 것이다. 분리의 유효성은 프로세서 사이에 자원을 공유하지 않으면 확인(ensure)할 수 있다.

5.3.6 소프트웨어 아키텍처 검증

제조업체는 다음을 검증하고 문서화해야 한다. [등급 B, C]

a) 소프트웨어 아키텍처가 위험관리에 관련된 요구사항을 포함해 시스템 및 소프트웨어 요구사항을 구현한다.

b) 소프트웨어 아키텍처는 소프트웨어 항목 사이의 연계성과 소프트웨어 항목 및 하드웨어 사이의 연계성을 지원할 수 있다.

c) 의료기기 아키텍처는 모든 SOUP 항목의 적절한 운영을 지원한다.

5.4 소프트웨어 상세 설계

5.4.1 소프트웨어 아키텍처를 소프트웨어 유닛으로 개량

제조업체는 소프트웨어 유닛으로 표현될 때까지 소프트웨어 아키텍처를 개량해야 한다. [등급 B, C]

5.4.2 각 소프트웨어 유닛에 대해 상세 설계 개발

제조업체는 소프트웨어 항목의 각 소프트웨어 유닛에 대한 상세 설계를 개발하고 문서화해야 한다. [등급 C]

5.4.3 연계성에 대한 상세 설계 개발

제조업체는 소프트웨어 유닛 및 외부 구성요인(하드웨어나 소프트웨어) 사이의 연계성 및 소프트웨어 유닛 사이의 연계성을 위한 상세 설계를 개발하고 문서화해야 한다. [등급 C]

5.4.4 상세 설계 검증

제조업체는 소프트웨어 상세 설계가 다음과 같은지 검증하고 문서화해야 한다. [등급 C]

a) 소프트웨어 아키텍처를 구현한다.

b) 소프트웨어 아키텍처와 상충하지 않는다.

5.5 소프트웨어 유닛 구현 및 검증

5.5.1 각 소프트웨어 유닛 구현

제조업체는 각 소프트웨어 유닛을 구현해야 한다. [등급 A, B, C]

5.5.2 소프트웨어 유닛 검증 프로세스 확립

제조업체는 각 소프트웨어 유닛 검증을 위한 전략, 방법 및 절차를 확립해야 한다. 시험을 검증할 경우 시험 절차의 정확성을 평가해야 한다. [등급 B, C]

주 : 통합 시험 및 소프트웨어 시스템 시험을 단일 계획 및 활동으로 결합할 수 있다.

5.5.3 소프트웨어 유닛 인수 기준

제조업체는 해당되는 경우 규모가 큰 소프트웨어 항목으로 통합하기 전 소프트웨어 유닛에 대한 인수 기준을 확립하고, 소프트웨어 유닛이 인수 기준을 충족하는지 확인해야 한다. [등급 B, C]

주 : 인수 기준의 예는 아래와 같다.

- 소프트웨어 코드가 위험관리 방법을 포함해 요구사항을 구현하는가?

- 소프트웨어 코드가 소프트웨어 유닛의 상세 설계에 문서화한 연계성과 상충하지 않는가?
- 소프트웨어 코드가 프로그래밍 절차나 코딩 표준과 일치하는가?

5.5.4 소프트웨어 유닛 추가 인수 기준

제조업체는 설계에 필요한 경우 다음과 같은 추가 인수 기준을 포함시켜야 한다. [등급 C]

a) 적절한 사상의 순서
b) 데이터 및 관리 흐름
c) 계획한 자원 할당
d) 결합 취급(오류 정의, 확인 및 복구)
e) 변수의 초기화
f) 자체 진단
g) 메모리 관리 및 메모리 용량초과
h) 경계 조건

5.5.5 소프트웨어 유닛 검증

제조업체는 소프트웨어 유닛 검증을 수행하고 결과를 문서화해야 한다. [등급 B, C]

5.6 소프트웨어 통합 및 통합 시험

5.6.1 소프트웨어 유닛 통합

제조업체는 통합계획(5.1.5 참조)에 따라 소프트웨어 유닛을 통합해야 한다.[등급 B, C]

5.6.2 소프트웨어 통합 검증

제조업체는 통합계획(5.1.5 참조)에 따라 소프트웨어 통합에 관한 다음과 같은 사항을 검증하고 기록해야 한다. [등급 B, C]

a) 소프트웨어 유닛이 소프트웨어 항목(item)과 소프트웨어 시스템으로 통합되었다.
b) 하드웨어 항목(item), 소프트웨어 항목(item) 및 시스템의 수동 운영(예; 사람 연계성, 온라인 도움말 메뉴, 음성 식별, 음성관리)을 위한 지원이 시스템에 통합되었다.

주 : 이 검증은 항목(item)이 의도된 대로 기능을 수행하는지 검증이 아니라, 계획에 따라 항목(item)이 통합되었는지 검증하는 것이다. 이 검증은 몇 가지 형태의 검사에 의해 구현된다.

5.6.3 통합 소프트웨어 시험

제조업체는 통합계획(5.1.5 참조)에 따라 통합된 소프트웨어 유닛을 시험하고 결과를 문서화해야 한다. [등급 B, C]

5.6.4 통합 시험 내용

소프트웨어 통합 시험의 경우 제조업체는 통합된 소프트웨어 항목이 의도한대로 기능을 발휘하는지 확인(address)해야 한다. [등급 B, C]

주 1 : 고려해야 할 예는 다음과 같다.

- 소프트웨어에 요구되는 기능성
- 위험관리 방법의 구현
- 지정한 시점 및 기타 작동상태
- 내부 및 외부 연계성의 지정 기능
- 예측할 수 있는 오용을 포함해 비정상 조건에서 시험

주 2 : 통합 시험 및 소프트웨어 시스템 시험을 단일 계획 및 활동으로 결합할 수 있다.

5.6.5 통합 시험 절차 검증

제조업체는 통합 시험 절차의 정확성을 평가해야 한다. [등급 B, C]

5.6.6 회귀시험 수행

소프트웨어 항목을 통합할 때 제조업체는 이전에 통합한 소프트웨어에 결함이 발생하지 않았음을 입증하기에 적합한 회귀시험을 수행해야 한다. [등급 B, C]

5.6.7 통합시험 기록 내용

제조업체는 다음을 수행해야 한다. [등급 B, C]

a) 시험 결과를 문서화한다(합격/불합격 및 비정상 상태 목록).

b) 시험을 반복할 수 있도록 충분한 기록을 유지한다.

c) 시험자를 식별한다.

주 : 요구사항 b)는 예를 들어 다음을 유지함으로써 구현할 수 있다.

- 요구되는 조치와 예상되는 결과를 나타내는 시험 사례 시방서
- 장치 기록
- 시험에 사용하는 시험 환경(소프트웨어 도구 포함)의 기록

5.6.8 소프트웨어 문제해결 프로세스의 사용

제조업체는 소프트웨어 통합 및 통합 시험 중 확인(found)한 비정상 상태를 소프트웨어 문제해결 프로세스에 입력해야 한다. [등급 B, C]

주 : 제9조 참조

5.7 소프트웨어 시스템 시험

5.7.1 소프트웨어 요구사항에 대한 시험 확립

제조업체는 모든 소프트웨어 요구사항이 포함되도록, 입력 촉진, 예상 출력, 합격/불합격 기준과 절차로 표현되는 시험을 확립하고 수행해야 한다. [등급 B, C]

주 1 : 통합 시험 및 소프트웨어 시스템 시험을 단일 계획 및 활동으로 결합할 수 있다. 또한 조기 단계에서 소프트웨어 요구사항을 시험할 수도 있다.

주 2 : 특히 요구사항 사이에 의존성이 있을 경우 각 요구사항에 대한 별도의 시험 뿐 아니라 요구사항의 결합에 대한 시험도 수행할 수 있다.

5.7.2 소프트웨어 문제해결 프로세스의 사용

제조업체는 소프트웨어 시스템 시험 중 확인(found)한 비정상 상태를 소프트웨어 문제해결 프로세스에 입력해야 한다. [등급 B, C]

5.7.3 변경 후 재시험

소프트웨어 시스템 시험 중 변경이 있을 경우 제조업체는 다음을 수행해야 한다. [등급 B, C]

a) 문제해결에 있어 변경의 유효성 검증을 위해 시험을 반복하건, 수정한 시험을 수행하거나 추가 시험을 수행한다.

b) 의도하지 않은 부작용이 발생하지 않음을 입증하기에 적합한 시험을 수행한다.

c) 7.4에 정의한 것처럼 관련 위험관리 활동을 수행한다.

5.7.4 소프트웨어 시스템 시험 검증

제조업체는 다음을 검증한다. [등급 B, C]

a) 사용한 검증 전략과 시험 절차가 적절하다.

b) 소프트웨어 시스템 시험 절차가 소프트웨어 요구사항까지 추적한다.

c) 모든 소프트웨어 요구사항을 시험하거나 검증했다.

d) 시험 결과가 요구되는 합격/불합격 기준을 충족한다.

5.7.5 소프트웨어 시스템 시험 기록 내용

제조업체는 다음을 수행해야 한다. [등급 B, C]

a) 시험 결과를 문서화한다(합격/불합격 및 비정상 상태 목록).

b) 시험을 반복할 수 있도록 충분한 기록을 유지한다.

c) 시험자를 식별한다.

주 : 요구사항 b)는 예를 들어 다음을 유지함으로써 구현할 수 있다.

- 요구되는 조치와 예상되는 결과를 나타내는 시험 사례 시방서
- 장치 기록
- 시험에 사용하는 시험 환경(소프트웨어 도구 포함)의 기록

5.8 소프트웨어 릴리즈

5.8.1 소프트웨어 확인의 완벽성 확인(ensure)

제조업체는 소프트웨어 검증이 완료되었으며 소프트웨어를 릴리즈 하기 전 결과가 평가되었는지 확인(ensure)해야 한다. [등급 B, C]

5.8.2 확인(known)된 잔류 비정상 상태 문서화

제조업체는 모든 확인(known)된 잔류 비정상 상태를 문서화해야 한다. [등급 B, C]

5.8.3 확인(known)된 잔류 비정상 상태 평가

제조업체는 확인(known)한 모든 잔류 비정상 상태가 허용되지 않는 위험을 발생시키지 않는지 확인(ensure)하기 위해 그러한 비정상 상태가 평가되었는지 확인(ensure)해야 한다. [등급 B, C]

5.8.4 릴리즈 버전의 문서화

제조업체는 릴리즈 중인 소프트웨어 제품의 버전을 문서화해야 한다.[등급 A, B, C]

5.8.5 릴리즈한 소프트웨어를 작성한 방법 문서화

제조업체는 릴리즈한 소프트웨어 작성에 사용한 절차와 환경을 문서화해야 한다.[등급 B, C]

5.8.6 활동과 업무의 완결 확인(ensure)

제조업체는 모든 관련 문서화와 함께 모든 활동과 업무가 완결되었는지 확인(ensure)해야 한다. [등급 B, C]

5.8.7 소프트웨어 저장

제조업체는 최소 제조업체가 정의한 장비의 수명시간과 관련 규제 요구사항에 의해 명시된 시간 중 긴 시간으로 결정된 기간 동안 다음을 저장해야 한다. [등급 B, C]

a) 소프트웨어 제품 및 형상 항목

b) 문서

5.8.8 소프트웨어 릴리즈의 반복 정밀도 보장

제조업체는 릴리즈한 소프트웨어 제품이 개악이나 무단 변경 없이 신뢰성 있게 사용 지점까지 인도되는지 확인(ensure)할 수 있는 절차를 확립해야 한다. 이 절차는 다음을 포함해 소프트웨어 제품이 수록된 매체의 생산과 취급을 처리해야 한다. [등급 B, C]

- 복제
- 매체 라벨링
- 포장
- 보호
- 보관
- 인도

6. 소프트웨어 유지보수 프로세스

6.1 소프트웨어 유지보수 계획 확립

제조업체는 유지보수 프로세스의 활동과 업무 진행을 위한 소프트웨어 유지보수 계획을 확립해야 한다. 계획은 다음의 내용을 취급해야 한다. [등급 A, B, C]

a) 다음에 대한 절차
   - 수신
   - 문서화
   - 평가
   - 문제해결
   - 추적; 의료기기 소프트웨어의 릴리즈 후 발생하는 피드백

b) 피드백이 문제로 간주되는지 여부를 결정하기 위한 기준

c) 소프트웨어 위험관리 프로세스의 사용

d) 의료기기 소프트웨어 릴리즈 후 발생하는 문제 분석과 해결을 위한 소프트웨어 문제해결 프로세스의 사용

e) 기존 소프트웨어 수정 관리를 위해 형상관리 프로세스(제8조) 사용

f) 평가 및 구현을 위한 절차

- 갱신
- 버그 해결
- 패치
- SOUP의 폐기

6.2 문제 및 수정 분석

6.2.1 피드백 문서화 및 평가

6.2.1.1 피드백 모니터링

제조업체는 자체 조직 내부와 사용자로부터 릴리즈된 소프트웨어 제품에 대한 피드백을 모니터링 해야 한다. [등급 A, B, C]

6.2.1.2 피드백 문서화 및 평가

릴리즈한 소프트웨어 제품에 문제가 존재하는지 여부를 판단하기 위해 피드백을 문서화하고 평가해야 한다. 그러한 문제는 모두 문제 보고서로서 기록해야 한다(제9조 참조). 문제 보고서에는 실제 또는 잠재적 부정적 사상 및 시방서와의 편차가 포함되어야 한다. [등급 A, B, C]

6.2.1.3 안전에 관한 문제 보고서 영향의 평가

문제가 릴리즈된 소프트웨어 제품에 어떤 영향을 주는지, 릴리즈한 소프트웨어 제품에 대한 변경 때문에 문제의 처리가 필요한지 여부의 결정을 위해 각 문제 보고서를 평가해야 한다. [등급 A, B, C]

6.2.2 소프트웨어 문제해결 프로세스의 사용

제조업체는 문제 보고서 처리를 위해 소프트웨어 문제해결 프로세스(제9조 참조)를 사용해야 한다. [등급 A, B, C]

주 : 이 활동이 완료되면 소프트웨어 시스템이나 소프트웨어 항목의 안전 등급의 변경을 파악해야 한다.

6.2.3 변경 요청 분석

제9조에서 요구하는 분석 이외에도 제조업체는 조직, 릴리즈한 소프트웨어 제품 및 제품이 연계성을 갖는 시스템에 대한 영향의 파악을 위해 각 변경 요청을 분석해야 한다. [등급 B, C]

6.2.4 변경 요청 승인

제조업체는 릴리스한 소프트웨어 제품을 수정하는 변경 요청을 평가하고 승인해야 한다. [등급 A, B, C]

6.2.5 사용자 및 규제자와의 의사소통

제조업체는 승인된 변경 요청 중 릴리스한 소프트웨어 제품에 영향을 주는 변경 요청을 식별해야 한다.

제조업체는 지역 규정에 의해 요구되는 경우 다음의 내용을 사용자와 규제자에게 알려야 한다. [등급 A, B, C]

a) 릴리스한 소프트웨어 제품의 문제 및 변경하지 않고 계속 사용함으로써 발생하는 결과
b) 릴리즈한 소프트웨어 제품에 사용할 수 있는 변경의 성격 및 변경 내용을 확보하고 설치하는 방법

6.3 수정 구현

6.3.1 확립된 프로세스를 사용해 수정 구현

제조업체는 소프트웨어 개발 프로세스(제5조 참조)나 수정의 구현을 위해 확립한 유지보수 프로세스를 사용해야 한다. [등급 A, B, C]

주 : 소프트웨어 변경의 위험관리에 관련된 요구사항은 7.4를 참조하라.

6.3.2 수정한 소프트웨어 시스템의 재 릴리즈

제조업체는 5.8에 따라 수정한 소프트웨어 시스템을 릴리즈해야 한다. 수정은 소프트웨어 시스템의 전체 재 릴리즈 일환으로서, 또는 변경된 소프트웨어 키트와 기존 소프트웨어 시스템에 변경을 수정으로서 설치하기에 필요한 도구로 구성된 수정 키트로서 릴리즈할 수 있다. [등급 A, B, C]

7. 소프트웨어 위험관리 프로세스

7.1 위험한 상황에 대한 소프트웨어 영향의 분석

7.1.1 위험한 상황에 영향을 줄 수 있는 소프트웨어 항목의 식별

제조업체는 ISO 14971의 의료기기 분석활동에서 식별된 위험한 상황에 영향을 줄 수 있는 소프트웨어 항목을 식별해야 한다(4.2 참조). [등급 B, C]

주 : 위험한 상황은 소프트웨어 고장의 직접적인 결과, 또는 소프트웨어에 구현되는 위험관리 방법의 결함 결과에 의한 것일 수 있다.

7.1.2 위험한 상황에 영향을 줄 수 있는 잠재적 원인의 식별

제조업체는 위험한 상황에 영향을 주는 것으로 위에서 식별한 소프트웨어 항목의 잠재적 원인을 식별해야 한다. [등급 B, C]

제조업체가 고려해야 할 잠재적 원인은 다음과 같다.

a) 기능성의 부정확하거나 불완전한 시방서
b) 식별한 소프트웨어 항목 기능성에 있어 소프트웨어의 결함
c) SOUP의 고장 또는 예상하지 못한 결과
d) 예측할 수 없는 소프트웨어 운영을 야기할 수 있는 하드웨어 고장이나 기타 소프트웨어 결함
e) 타당한 수준으로 예측할 수 있는 오용

7.1.3 발표된 SOUP 비정상 상태 목록의 평가

SOUP의 고장이나 예상하지 못한 결과가 소프트웨어 항목이 위험한 상황에 영향을 주는 잠재적 원인일 경우 제조업체는 최소한 의료기기에 사용되는 SOUP 항목의 버전과 관련해 SOUP 항목의 공급업체가 발표한 비정상 상태 목록을 평가해 확인(known)된 비정상 상태에 의해 위험한 상황을 야기할 수 있는 일련의 사상이 발생했는지 판단해야 한다. [등급 B, C]

7.1.4 잠재적 원인의 문서화

제조업체는 위험한 상황에 영향을 주는 소프트웨어 항목의 잠재적 원인을 위험관리 파일에 문서화해야 한다(ISO 14971 참조). [등급 B, C]

7.1.5 사상 발생순서 문서화

제조업체는 7.1.2에서 식별한 위험한 상황을 야기할 수 있는 사상의 발생순서를 위험관리 파일에 문서화해야 한다. [등급 B, C]

7.2 위험관리 방법

7.2.1 위험관리 방법 정의

위험관리 파일에 문서화한, 위험한 상황에 영향을 주는 소프트웨어 항목의 각 잠재적 원인에 있어 제조업체는 위험관리 방법을 정의하고 문서화해야 한다. [등급 B, C]

주 : 위험관리 방법은 하드웨어, 소프트웨어, 작업환경 또는 사용자 설명서에 구현할 수 있다.

7.2.2 소프트웨어에 구현된 위험관리 방법

위험관리 방법이 소프트웨어 항목의 기능 일부로서 구현된 경우 제조업체는 다음과 같이 수행해야 한다. [등급 B, C]

a) 소프트웨어 요구사항에 위험관리 방법을 포함시킨다.

b) 위험관리 방법이 관리 중인 위해의 영향에 기초해 소프트웨어 항목에 소프트웨어 안전성 등급을 할당한다.

c) 제5조에 따라 소프트웨어 항목을 개발한다.

주 : 이 요구사항은 ISO 14971의 위험관리 요구사항에 대한 상세한 설명을 추가로 제공한다.

7.3 위험관리 방법의 검증

7.3.1 위험관리 방법 검증

7.2에 문서화한 각 위험관리 방법의 구현은 검증되어야 하며, 검증 결과를 문서화해야 한다. [등급 B, C]

7.3.2 사상의 새로운 발생순서 문서화

위험관리 방법을 소프트웨어 항목으로서 구현한 경우 제조업체는 위험관리 방법을 평가해 위험한 상황을 야기할 수 있는 사상의 새로운 발생순서를 식별하고 위험관리 파일에 문서화해야 한다. [등급 B, C]

7.3.3 추적성 문서화

제조업체는 소프트웨어 위험요인의 추적성을 다음과 같이 문서화해야 한다. [등급 B, C]

a) 위험한 상황부터 소프트웨어 항목까지

b) 소프트웨어 항목부터 특정 소프트웨어 원인까지

c) 소프트웨어 원인이나 위험관리 방법까지

d) 위험관리 방법부터 위험관리 방법의 검증까지

주 : ISO 14971 – 위험관리 보고서 참조

7.4 소프트웨어 변경의 위험관리

7.4.1 안전성과 관련해 의료기기 소프트웨어에 대한 변경 분석

제조업체는 다음의 내용을 파악하기 위해 의료기기 소프트웨어(SOUP 포함)에 대한 변경을 분석해야 한다. [등급 A, B, C]

a) 위험한 상황에 영향을 줄 수 있는 추가 잠재적 원인

b) 추가로 필요한 소프트웨어 위험관리 방법

7.4.2 기존 위험관리 방법에 대한 소프트웨어 변경의 영향 분석

제조업체는 SOUP에 대한 변경을 포함해 소프트웨어에 대한 변경을 분석해 소프트웨어 수정이 기존 위험관리 방법을 방해할 수 있는지의 여부를 판단해야 한다. [등급 B, C]

7.4.3 분석에 기초해 위험관리 활동 수행

제조업체는 이러한 분석에 기초해 7.1, 7.2 및 7.3에 정의한 관련 위험관리 활동을 수행해야 한다. [등급 B, C]

8. 소프트웨어 형상관리 프로세스

8.1 형상 식별

8.1.1 형상항목(item)을 식별하기 위한 방법 확립

제조업체는 프로젝트를 위해 관리해야 하는 형상 항목 및 그 버전의 고유 식별을 위한 계획을 수립해야 한다. 이 계획에는 SOUP와 문서와 같은 다른 소프트웨어 제품이나 실체가 포함되어야 한다. [등급 A, B, C]

8.1.2 SOUP 식별

표준 라이브러리를 포함해 사용 중인 각 SOUP 형상항목의 경우 제조업체는 각 형상항목에 대해 다음을 문서화해야 한다. [등급 A, B, C]

a) 제목
b) 제조업체
c) 고유 SOUP 지정자

주 : 고유 SOUP 지정자는 예를 들어 버전, 릴리즈 날짜, 패치 번호 또는 업그레이드 명칭일 수 있다.

8.1.3 시스템 형상 문서 식별

제조업체는 소프트웨어 시스템 형상을 구성하는 형상항목과 그 버전을 문서화해야 한다. [등급 A, B, C]

8.2 변경관리

8.2.1 변경 요청 승인

제조업체는 승인된 변경요청에 대해서만 형상항목을 변경해야 한다. [등급 A, B, C]

주 1 : 변경요청의 승인 결정은 변경관리 프로세스의 일부이거나 다른 프로세스 일부일 수 있다. 이 부절에 따르면 변경을 승인할 경우 구현을 대신한다.

주 2 : 계획에 언급된 것처럼 수명주기의 다양한 단계에서 다양한 인수 프로세스를 변경 요청에 사용할 수 있다. 5.1.1 e) 및 6.1 e) 참조.

8.2.2 변경 구현

제조업체는 변경요청에 명시한 것처럼 변경을 구현해야 한다. 제조업체는 소프트웨어 시스템과 소프트웨어 항목의 소프트웨어 안전성 분류에 대한 변경을 포함해, 변경의 결과로서 반복해야 할 활동을 식별하고 수행해야 한다. [등급 A, B, C]

주 : 이 부절에는 적절한 변경관리 달성을 위해 변경을 구현하는 방법이 설명된다. 단, 구현이 변경관리 프로세스의 필수적인 부분이라고 암시하는 것은 아니다. 구현은 계획한 프로세스를 사용해야 한다. 5.1.1 e) 및 6.1 e) 참조.

8.2.3 변경 검증

제조업체는 변경에 의해 무효가 된 버전의 반복과 5.7.3 및 9.7의 고려를 포함해 변경을 검증해야 한다. [등급 A, B, C]

주 : 이 부절에 따르면 변경만 검증하면 된다. 단, 검증이 변경관리 프로세스의 필수적인 부분이라고 암시하는 것은 아니다. 검증은 계획한 프로세스를 사용해야 한다. 5.1.1 e) 및 6.1 e) 참조.

8.2.4 변경 추적성에 대한 방법 제공

제조업체는 다음을 추적할 수 있는 경우 그에 대해 심사 추적을 작성해야 한다. [등급 A, B, C]

a) 변경 요청
b) 관련 문제 보고서
c) 변경요청의 승인

8.3 형상 상태 설명

제조업체는 시스템 형상이 포함된, 관리된 형상 항목에 대해 검색할 수 있는 이력 기록을 유지해야 한다. [등급 A, B, C]

9. 소프트웨어 문제해결 프로세스

9.1 문제 보고서 작성

제조업체는 소프트웨어 제품에서 검출된 각 문제에 대해 문제 보고서를 작성해야 한다. 문제 보고서는 다음과 같이 분류해야 한다. [등급 A, B, C]

a) 유형: 예제 1 교정, 예방 또는 새 환경에 대한 적응성 유형
b) 범위: 예제 2 변경 규모, 해당 장비 모델의 수, 지원되는 해당 부속품, 관련 자원 및 변경할 시간
c) 심각도: 예제 3 성능, 안전성 또는 보안에 대한 영향

주 : 문제는 제조업체 조직 내부나 외부에서 릴리즈 전 또는 이후 발견할 수 있다.

9.2 문제 조사

제조업체는 다음을 수행해야 한다. [등급 A, B, C]

a) 문제를 조사하고, 가능한 경우 원인을 식별한다.
b) 소프트웨어 위험관리 프로세스를 사용해 안전성에 관련된 문제를 평가한다(제7조)
c) 조사와 평가의 결과를 문서화한다.
d) 문제 해결에 필요한 조치에 대해 변경 요청을 작성하거나, 조치를 취하지 않을 경우 그 이론적 근거를 문서화한다.

주 : 문제가 안전과 관련되지 않은 경우 제조업체는 소프트웨어 문제해결 프로세스의 준수를 위해 문제를 해결할 필요는 없다.

9.3 관련 당사자에게 통보

제조업체는 문제가 있는 경우 그를 관련 당사자에게 통보해야 한다. [등급 A, B, C]

주 : 문제는 제조업체 조직 내부나 외부에서 릴리즈 전 또는 이후 발견할 수 있다. 제조업체는 상황에 따라 관련 당사자를 결정한다.

9.4 변경관리 프로세스의 사용

제조업체는 변경관리 프로세스의 요구사항을 준수하면서 모든 변경요청을 승인하고 구현해야 한다(8.2 참조). [등급 A, B, C]

9.5 기록 유지

제조업체는 문제 보고서와, 검증을 포함해 문제해결의 기록을 유지해야 한다.

제조업체는 필요한 경우 위험관리 파일을 갱신해야 한다(7.4 참조). [등급 A, B, C]

9.6 문제 경향 분석

제조업체는 문제 보고서에서 경향을 확인(detect)하기 위한 분석을 수행해야 한다. [등급 A, B, C]

9.7 소프트웨어 문제해결 검증

제조업체는 다음을 판단하기 위해 해결을 검증해야 한다. [등급 A, B, C]

a) 문제가 해결되었고 문제 보고서가 마감되었다.

b) 부정적인 경향이 반전되었다.

c) 변경요청이 해당 소프트웨어 제품 및 활동에 구현되었다.

d) 문제가 추가로 발생했다.

9.8 시험 문서의 내용

변경 후 소프트웨어 항목과 시스템을 시험, 재시험 또는 회귀 시험할 때 제조업체는 시험 문서에 다음 내용을 포함시켜야 한다. [등급 A, B, C]

a) 시험 결과

b) 확인(found)한 비정상 상태

c) 시험한 소프트웨어의 버전

d) 관련 하드웨어와 소프트웨어 시험 형상

e) 관련 시험 도구

f) 시험 날짜

g) 시험자의 신원

부록 A (참조용) 이 표준의 요구사항에 대한 이론적 근거

이 시험의 조항에 대한 이론적 근거가 이 부록에 제공된다.

A.1 이론적 근거

이 표준의 일차적인 요구사항은 프로세스가 의료기기 소프트웨어의 개발과 유지보수에 따라 진행되며, 프로세스의 선택이 환지 및 다른 사람에 대한 위험에 적절해야 한다는 것이다. 이러한 요구사항은 소프트웨어의 시험으로는 안전하게 운영되는지 판단하기 충분하지 않다는 판단에 따른 것이다.

이 표준에 요구되는 프로세스는 두 가지 범주로 분류할 수 있다.

- 소프트웨어의 각 소프트웨어 항목 운영에서 발생하는 위험의 판단에 필요한 프로세스
- 결정한 위험을 기준으로 선택한, 각 소프트웨어 항목에 대한 소프트웨어 고장의 적절하게 낮은 발생가능성의 달성에 필요한 프로세스

이 표준에 따르면 첫 번째 범주는 모든 의료기기 소프트웨어에, 두 번째 범주는 선택한 소프트웨어 항목에 대해 수행해야 한다.

따라서 이 표준 준수성 주장에는 소프트웨어가 포함되며 위험한 상황을 야기할 수 있는 예측 가능한 발생순서가 식별되는 문서화된 위험분석이 포함되어야 한다. 소프트웨어에 의해 간접적으로 발생할 수 있는 위험요인(예를 들어 잘못된 정보를 제공해 치료가 부적절하게 수행되는 경우)은 이 위험분석에 포함되어야 한다.

프로세스의 첫 번째 범주의 일부로서 필요한 모든 활동은 규정 텍스트에 “[등급 A, B, C]”로 표시되어, 프로세스가 적용되는 소프트웨어의 분류와 관련이 없음을 나타낸다.

다음과 같은 이유 때문에 등급 A, B와 C 전체에 활동이 필요하다.

- 활동은 위험관리 또는 소프트웨어 안전성 분류에 관련된 계획을 생성한다.
- 활동은 위험관리 또는 소프트웨어 안전성 분류에 대한 입력인 출력을 생성한다.
- 활동은 위험관리 또는 소프트웨어 안전성 분류의 일부이다.
- 활동은 위험관리 또는 소프트웨어 안전성 분류를 지원하는 행정 시스템, 문서화 및 기록유지 시스템을 확립한다.

- 일반적으로 활동은 관련 소프트웨어의 분류를 확인(unknown)할 수 없을 경우 발생한다.
- 활동에 따라서 연관 소프트웨어의 현재 소프트웨어 안전성 분류를 무효화할 수 있는(invalidate) 변경이 발생할 수 있다. 여기에는 릴리즈 후 안전성 관련 문제의 확인(discovery)과 분석이 포함된다.

다른 프로세스는 소프트웨어 안전성 등급 B나 C로 분류된 소프트웨어 시스템이나 소프트웨어 항목에만 필요하다. 이러한 프로세스의 일부로서 필요한 활동은 규정 텍스트에 "[등급 B, C]" 또는 "[등급 C]"로 표시되어, 프로세스가 적용되는 소프트웨어의 분류에 선택적으로 의존해야 함을 나타낸다.

다음과 같은 이유 때문에 등급 B와 C의 소프트웨어에 선택적 활동이 필요하다.
- 활동은 설계, 시험 및 기타 검증에 있어 보다 상세한 내용이나 엄격성 강화가 필요하기 때문에 소프트웨어의 신뢰성을 강화한다.
- 활동은 등급 B나 C에 필요한 다른 활동을 지원하는 행정적 활동이다.
- 활동은 안전관련 문제 해결을 지원한다.
- 활동은 안전관련 소프트웨어의 설계, 구현, 검증 및 릴리즈의 기록을 생성한다.

다음과 같은 이유 때문에 C의 소프트웨어에 선택적 활동이 필요하다.
- 활동은 설계, 시험 및 기타 검증에 있어 보다 상세한 내용이나 강화된 엄격성 또는 주의가 필요하기 때문에 소프트웨어의 신뢰성을 한층 강화한다.

이 표준에 정의한 모든 프로세스와 활동은 고품질 소프트웨어 개발과 유지보수 확보에 귀중한 것으로 간주된다. 위험요인을 발생시킬 수 없는 등급 A 소프트웨어에 대한 요구사항으로서 이러한 프로세스 및 활동을 다수 생략하는 경우에도 그러한 프로세스와 활동에 가치가 없거나 권장되지 않음을 의미하는 것은 아니다. 생략의 목적은 의료기기 설계 중 일차적으로 종합적인 밸리데이션 활동(이는 이 표준의 범위에 포함되지 않음) 및 일부 단순한 소프트웨어 수명주기 관리를 통하여 안전성과 효과성에 있어 위험요인을 유발할 수 없는 소프트웨어를 쉽게 확인(assure)할 수 있는 소프트웨어를 확인(recognize)하는 것이다.

A.2 등급별 요구사항의 요약

표 A.1에는 각 요구사항에 할당된 소프트웨어 안전성 등급이 요약 설명된다. 이 표는 참고용이며, 사용상 편의 목적만을 위해 제공된다. 규정 부분에 각 요구사항에 대한 소프트웨어 안전성 등급이 설명된다.

조항 및 부절		등급 A	등급 B	등급 C
제4조	전체 요구사항	x	x	x
제5.1조	5.1.1, 5.1.2, 5.1.3, 5.1.6, 5.1.7, 5.1.8, 5.1.9	x	x	x
	5.1.5, 5.1.10, 5.1.11		x	x
	5.1.4			x
제5.2조	5.2.1, 5.2.2, 5.2.4, 5.2.5, 5.2.6	x	x	x
	5.2.3		x	x
제5.3조	5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.3.6		x	x
	5.3.5			x
제5.4조	5.4.1		x	x
	5.4.2, 5.4.3, 5.4.4			x
제5.5조	5.5.1	x	x	x
	5.5.2, 5.5.3, 5.5.5		x	x
	5.5.4			x
제5.6조	전체 요구사항		x	x
제5.7조	전체 요구사항		x	x
제5.8조	5.8.4,	x	x	x
	5.8.1, 5.8.2, 5.8.3, 5.8.5, 5.8.6, 5.8.7, 5.8.8		x	x
제6.1조	6.1	x	x	x
제6.2조	6.2.1, 6.2.2, 6.2.4, 6.2.5	x	x	x
	6.2.3		x	x
제6.3조	전체 요구사항	x	x	x
제7.1조	전체 요구사항		x	x
제7.2조	전체 요구사항		x	x
제7.3조	전체 요구사항		x	x
제7.4조	7.4.1	x	x	x
	7.4.2, 7.4.3		x	x
제8조	전체 요구사항	x	x	x
제9조	전체 요구사항	x	x	x

[표 A.1] 소프트웨어 안전성 등급별 요구사항의 요약

부록 B (참조용) 이 표준의 규정에 관한 지침

B.1 범위

B.1.1 목적

이 표준의 목적은 고품질의 안전한 의료기기 소프트웨어를 일관성 있게 생산하는 개발 프로세스를 제공하는 것이다. 이 목적의 달성을 위해 이 표준은 소프트웨어가 신뢰성 높고 안전한 소프트웨어 제품을 생산할 수 있는 방식으로 개발되었다는 확신을 제공하기 위해 달성해야 하는 최소한의 활동과 업무를 식별해야 한다.

이 부록에는 이 표준의 요구사항 적용에 대한 지침이 제공된다. 이 부록은 이 표준의 요구사항에 새로운 요구사항을 추가하거나 기존 요구사항을 변경하지 않는다. 이 부록은 이 표준의 요구사항을 보다 잘 이해하기 위해 사용할 수 있다.

이 표준에서 활동은 프로세스 안에서 필요한 부절이며, 업무는 활동 안에 정의된다. 예를 들어, 소프트웨어 개발 프로세스를 위해 정의한 활동에는 소프트웨어 개발 기획, 소프트웨어 요구사항 분석, 소프트웨어 구축설계, 소프트웨어 상세설계, 소프트웨어 유닛 구현과 검증, 소프트웨어 통합과 통합 시험, 소프트웨어 시스템 시험 및 소프트웨어 릴리즈가 포함된다. 이러한 활동 안의 업무는 개별 요구사항이다.

이 표준에는 특정 소프트웨어 개발 수명주기 모델이 필요 없다. 단, 프로세스는 다른 프로세스에 의해 생성되기 때문에 이 표준에 따라 프로세스 사이의 의존성이 있다고 암시하는 것은 아니다. 예를 들어 위험분석 프로세스에 따라 소프트웨어 시스템의 고장에서 발생할 수 있는 위해를 확인(establish)한 후 소프트웨어 시스템의 소프트웨어 안전성 분류를 완료해야 한다.

프로세스 사이의 이러한 논리적 의존성 때문에 이 표준의 프로세스를 발생순서대로, 즉 “일괄수행”이나 “일회성” 수명주기 모델로 설명하는 것이 가장 쉽다. 단, 다른 수명주기도 사용할 수 있다. ISO/IEC 12207[9]에 정의한 일부 개발(모델) 전략에는 다음이 포함된다(표 B.1 참조).

- 일괄수행 Waterfall ; “일괄수행”이라고도 하는 “일회성” 전략은 개발 프로세스를 한번 수행하는 것으로 구성된다. 단순화: 고객 요구를 결정하고, 요구사항을 정의하며, 시스템을 설계하고, 시스템을 구현하고 시험, 수리 및 인도한다.
- 증분 Incremental ; “증분” 전략은 고객이 요구를 결정하고 시스템 요구사항을 정의한 다음 빌드 순서로 나머지 개발을 수행한다. 첫째 빌드는 계획한 능력의 일부

를 사용하며, 둘째 빌드는 능력을 더 추가하는 식으로 시스템이 완료될 때까지 지속된다.

- 진화 Evolutionary ; “진화” 전략 또한 시스템을 빌드로 개발하지만, 사용자 요구가 완전히 이해되지 않으며 모든 요구사항을 처음부터 정의할 수 없다는 사실을 인정한다는 점에서 증분 전략과는 차이를 보인다. 이 전략에서 고객 요구와 시스템 요구사항 일부가 처음에 정의되며, 후속 빌드에서 개량된다.

개발 전략	모든 요구사항을 처음부터 정의하는가?	개발 주기가 복수인가?	임시 소프트웨어를 배포하는가?
일괄수행 (일회성)	예	아니오	아니오
증분 (계획한 제품 개선)	예	예	가능
진화	아니오	예	예

[표 B.1] ISO/IEC 12207에 정의한 개발(모델) 전략

어떤 수명주기를 선택하건 시방서, 설계 문서 및 소프트웨어와 같은 프로세스 출력 사이의 논리적 의존성을 유지해야 한다. 일괄수행 수명주기 모델은 프로세스의 입력이 완료되어 승인될 때까지 프로세스의 시작을 지연함으로써 이러한 의존성을 달성한다.

다른 수명주기, 특히 진화 수명주기를 사용하면 프로세스에 대한 입력을 모두 사용할 수 있기 전 프로세스 출력을 생성할 수 있다. 예를 들어 전체 소프트웨어 형상이 완성되기 전 새 소프트웨어 항목(item)을 지정, 분류, 구현 및 검증할 수 있다. 그러한 수명주기는 한 프로세스의 변경이나 개발 때문에 다른 프로세스 출력을 무효화하는(invalidate) 위험을 발생한다. 따라서 모든 수명주기는 종합적인 형상관리 시스템을 사용해 모든 프로세스 출력을 일관성 있는 상태로 유지하며, 의존성을 유지한다.

아래에 설명하는 원칙은 소프트웨어 개발 수명주기 종류에 관계 없이 중요하다.

- 모든 프로세스 출력은 일관성 있는 상태로 유지되어야 한다. 프로세스 출력이 작성되거나 변경될 때마다 관련된 모든 프로세스 출력은 서로 일관성을 유지하고, 이 표준이 명시적으로 또는 묵시적으로 요구하는 모든 의존성 유지를 위해 즉시 갱신해야 한다.
- 모든 프로세스 출력은 소프트웨어에 대한 후속 작업에 입력으로 필요한 경우 사용할 수 있어야 한다.

- 의료기기 소프트웨어를 릴리즈하기 전 모든 프로세스 출력은 서로 일관성을 갖추어야 하며, 이 표준이 명시적으로 또는 묵시적으로 요구하는 프로세스 사이의 모든 의존성이 준수되어야 한다.

B.1.2 적용분야

이 표준은 의료기기 소프트웨어의 개발과 유지보수는 물론 SOUP가 포함된 의료기기 개발과 유지보수에 적용된다.

이 표준을 사용하려면 제조업체는 ISO 14971에 따라 의료기기 위험관리을 수행해야 한다. 따라서 의료기기 소프트웨어 아키텍처에 SOUP가 포함된 프린터/플로터와 같은 구성요소가 포함될 경우(구입한 구성요소이거나 기원을 알 수 없는 구성요소일 수 있다) 그러한 구성요소는 제조업체가 책임을 담당해야 하며, 의료기기 위험관리에 포함되어야 한다. 의료기기 위험관리을 적절히 수행할 경우에만 제조업체는 그 구성요소를 이해하고, 그 구성요소에 SOUP가 포함된 사실을 확인(recognize)할 수 있다. 이 표준을 사용하는 제조업체는 위험관리 프로세스를 의료기기 위험관리 프로세스의 일환으로 사용할 수 있을 것이다.

릴리즈한 의료기기 소프트웨어의 유지보수는 의료기기 소프트웨어와의 생산 후 경험에 적용된다. 소프트웨어 유지보수에는 요구되는 성능과 릴리즈한 소프트웨어 제품에 관련된 수정 요청을 수행할 수 있는 상태로 항목(item)을 유지하거나 그러한 상태로 복귀시키기 위해 문제 보고서에 따라 조치를 취하기 위한 감독 조치가 포함된 모든 기술적 및 행정적 방법의 조합이 포함된다. 예를 들어, 여기에는 문제 확인(rectification), 규제 보고, 타당성 재확인 및 예방 조치가 포함된다. ISO/IEC 14764 [10] 참조.

B.2 규격상 참고문헌

ISO/IEC 90003[11]은 품질경영시스템을 소프트웨어 개발에 적용하기 위한 지침을 제공한다. 이 지침은 이 표준에서는 요구되지 않지만 강력히 권장되는 지침이다.

B.3 용어 정의

가능한 한 용어는 국제 표준의 정의를 사용해 정의했다.

이 표준은 소프트웨어 시스템(최상위)의 분해 설명에 세 가지 용어를 사용하기로 선택했다. 소프트웨어 시스템은 의료기기의 서브시스템(IEC 60601-1-4[2] 참조)이거나 의료기기 그 자체일 수 있다. 시험이나 소프트웨어 형상관리의 목적을 위해 더 이상 분

해할 수 없는 최저 수준이 소프트웨어 유닛이다. 최상위와 최하위가 포함된 모든 수준의 형상을 소프트웨어 항목이라고 부를 수 있다. 즉, 소프트웨어 항목은 하나 이상의 소프트웨어 항목으로 구성되며, 각 소프트웨어 항목은 하나 이상의 소프트웨어 유닛 또는 분해가능 소프트웨어 항목으로 구성된다. 소프트웨어 항목과 소프트웨어 유닛의 정의와 입도를 제공하는 것은 제조업체의 책임이다. 이러한 용어를 애매한 상태로 유지하면 다양한 의료기기에 사용되는 소프트웨어의 개발방법과 소프트웨어 유형에 이러한 용어를 적용할 수 있다.

B.4 일반 요구사항

어떤 종류의 소프트웨어에도 안전성을 100% 보장할 수 있는 방법은 없다.

의료기기 소프트웨어의 안전성을 향상시키는 주요 원칙에는 세 가지가 있다.

- 위험관리
- 품질경영
- 소프트웨어 엔지니어링

안전한 의료기기 소프트웨어를 개발하고 유지 보수하려면 적절한 소프트웨어 엔지니어링 방법 및 기술에 대한 전체적인 체제로서의 품질경영시스템의 일부로서 위험관리을 확립해야 한다. 이러한 세 가지 개념을 조합하면 의료기기 제조업체가 잘 구성되고 일관성 있게 반복 정밀도가 높은 의사결정 프로세스를 사용해 의료기기 소프트웨어의 안전성을 향상시킬 수 있다.

B.4.1 품질경영시스템

잘 구성되고 효과적인 소프트웨어 프로세스에는 경영, 기반구조, 개선 및 훈련과 같은 조직의 프로세스가 포함된다. 중복을 방지하고 이 표준을 소프트웨어 엔지니어링에 집중하기 위해 이러한 프로세스를 이 표준에서 생략했다. 이러한 프로세스는 품질경영시스템이 담당한다. ISO 13485[7]은 의료기기에 품질경영 개념을 적용하기 위한 국제표준이다. ISO 13485 품질경영시스템 요구사항을 준수한다 하더라도 국가 또는 지역 규제 요구사항과의 준수성이 자동으로 이루어지는 것은 아니다. 관련 규제 요구사항과의 준수성을 식별하고 확립하는 책임은 제조업체의 몫이다.

B.4.2 위험관리

의료기기 소프트웨어와 관련해 타당한 수준으로 예상할 수 있는 모든 위험을 고려할 수 있도록 소프트웨어 개발은 위험관리 활동에 충분히 참여한다.

이 소프트웨어 엔지니어링 표준에서 해당 위험관리 프로세스를 정의하기보다는 제조업체가 의료기기에 대한 위험관리을 명백히 취급하며, ISO 14971을 준수하는 위험관리 프로세스를 적용하는 것이 바람직하다. 소프트웨어가 원인 중 하나가 되는 위험요인에서 발생하는 특정 소프트웨어 위험관리 활동은 제7조의 지원 프로세스에 설명되어 있다.

B.4.3 소프트웨어 안전성 분류

의료기기 일부로서, 의료기기 부속품으로서, 또는 의료기기 자체로서의 소프트웨어에 관련된 위험은 소프트웨어 안전성 분류 계획에 대한 입력을 사용되며, 이 경우 소프트웨어 개발과 유지보수 중 사용되는 프로세스가 결정된다.

위험은 부상 심각도 및 발생 가능성의 조합으로 간주된다. 단, 재래식 통계적 방법을 사용해 소프트웨어 고장 발생 가능성을 결정하는 방법에는 어떤 합의도 없다. 따라서 이 표준에서 소프트웨어 시스템 분류는 고장이 발생한다고 가정해, 소프트웨어의 고정에서 발생하는 위험요인의 심각도에 기초해 결정된다. 위험관리 방법의 구현에 기여하는 소프트웨어 시스템은 시스템이 관리하고 있는 위험요인의 심각도에 기초해 분류된다.

소프트웨어 시스템을 소프트웨어 항목을 분해할 경우 각 소프트웨어 항목에는 자체 소프트웨어 안전성 분류가 할당된다.

소프트웨어 항목의 고장에 관련된 위험은 다음과 같은 경우에만 결정할 수 있다.

- 시스템 아키텍처와 소프트웨어 아키텍처가 그 목적 및 다른 소프트웨어와 하드웨어 항목과의 연계성 관점에서 소프트웨어 항목의 역할을 정의할 경우
- 시스템에 대한 변경이 관리 중일 경우
- 지정된 아키텍처와 위험관리 방법에 위험분석을 수행한 후

이 표준에는 모든 등급의 소프트웨어에 대해 위에 규정한 조건을 달성하는 최소한의 활동이 필요하다.

소프트웨어 아키텍처 활동의 종료 시점은 전체 소프트웨어 항목이 정의되고 위험관리 활동에 따라 소프트웨어 항목이 안전성에 어떤 관련이 있는지 식별되는 경우 개발의 최초 시점이 된다. 따라서 이 시점은 소프트웨어 항목을 안전성 역할에 따라 명확하게 분류할 수 있는 최초의 시점이 된다.

이 시점은 위험관리가 ISO 14971에서 시작하는 시점에 해당된다.

이 시점 이전 위험관리 프로세스는 예를 들어 보호 서브시스템 추가 또는 소프트웨어 고장이 위해를 발생시킬 수 있는 가능성 축소와 같은 구조적 위험관리 방법을 식별한다. 이 시점 이후 위험관리 프로세스는 소프트웨어 항목 고장 발생가능성 축소에 집중하는 프로세스를 사용한다. 즉, 소프트웨어 항목의 분류에 따라 그 항목에 적용되는 프로세스 기반 위험관리 방법이 지정된다.

제조업체는 예를 들어 조사 대항 영역에 주의를 집중하면 이 시점 이전 소프트웨어를 쉽게 분류할 수 있지만, 그러한 분류는 예비적인 것으로 간주되어 프로세스 생략의 정당화에는 사용할 수 없다.

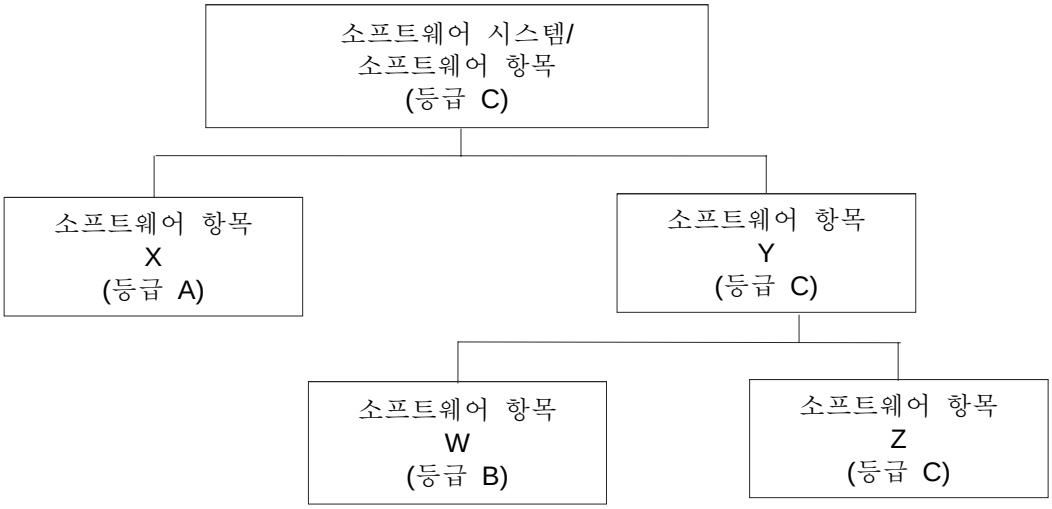
소프트웨어 안전성 분류 계획의 목적은 ISO 14971의 위험 분류와 일치시키는 것이 아니다. ISO 14971 계획은 위험을 심각도와 가능성에 따라 분류하지만, 소프트웨어 안전성 분류 계획은 개발 및 유지보수에 적용되는 프로세스에 따라 소프트웨어 시스템과 소프트웨어 항목을 분류한다.

설계가 진행되면 새로운 위험이 발생할 수 있다. 따라서 위험관리은 개발 프로세스의 일부로 적용해야 한다. 이 경우 안전한 운영과 결함이 위해를 야기하는 것을 방지하는 운영의 확보를 위해 기능이 정확하게 발휘되어야 하는 소프트웨어 항목이 포함된 전체 소프트웨어 항목을 식별할 수 있는 구축설계를 개발할 수 있다.

소프트웨어 아키텍처는 안전한 운영에 필요한 소프트웨어 항목의 분리를 촉진해야 하며, 그러한 소프트웨어 항목의 효과적인 분리 확보에 사용되는 방법을 설명해야 한다.

B.3에서 언급한 것처럼, 이 표준은 소프트웨어 시스템(최상위)의 분해 설명에 세 가지 용어를 사용하기로 선택한다.

그림 B.1은 소프트웨어 시스템 내부 소프트웨어 항목을 분할하는 방법과 분해 중 소프트웨어 항목 그룹에 소프트웨어 안전성 등급을 적용하는 방법이 설명된다.



[그림 B.1] 소프트웨어 항목 분할의 예

이 예의 경우 개발 중인 의료기기 소프트웨어의 유형 때문에 제조업체는 소프트웨어 시스템에 대한 예비 소프트웨어 안전성 등급이 소프트웨어 안전성 등급 C라는 사실을 안다. 소프트웨어 구축설계 중 제조업체는 그림처럼 세 가지 소프트웨어 항목, X, W 및 Z로 시스템을 분할할 것을 결정했다. 제조업체는 사망이나 중상을 발생할 수 있는 위험요인에 대한 모든 소프트웨어 시스템 영향을 소프트웨어 항목 Z로 분리하고, 비중상을 발생시킬 수 있는 위험요인에 대한 나머지 모든 소프트웨어 시스템을 소프트웨어 항목 W로 분리할 수 있다. 소프트웨어 항목 W는 소프트웨어 안전성 등급 B로 분류되며, 소프트웨어 항목 Z는 소프트웨어 안전성 등급 C로 분류된다. 따라서 소프트웨어 항목 Y는 4.3d)에 따라 등급 C로 분류되어야 한다. 또한 이 요구사항에 따라 소프트웨어 시스템은 소프트웨어 안전성 등급 C로 분류된다. 소프트웨어 항목 X는 소프트웨어 안전성 등급 A로 분류되었다. 제조업체는 분리의 무결성 확보를 위해 소프트웨어 항목 X와 Y 및 소프트웨어 항목 W와 Z 사이의 분리에 대한 이론적 근거를 문서화할 수 있다. 분할이 불가능할 경우 소프트웨어 항목 X와 Y는 소프트웨어 안전성 등급 C로 분류되어야 한다.

B.5 소프트웨어 개발 프로세스

B.5.1 소프트웨어 개발 기획

이 활동의 목표는 소프트웨어에 의해 발생하는 위험을 줄이고 절차와 목적을 개발팀 구성원에게 알리며 의료기기 소프트웨어에 대한 시스템 품질 요구사항이 충족되는지 확인(ensure)하기 위해 소프트웨어 개발 업무를 계획하는 것이다.

소프트웨어 개발 기획 활동에 다라 업무를 단일 계획이나 복수 계획에 문서화할 수 있다. 모든 의료기기 소프트웨어의 개발에 적용되는 정책과 절차를 이미 확립한 제조업체가 있을 수 있다. 이 경우 계획은 기존 정책과 절차를 참조할뿐 이다. 특정 활동을 자세히 규정하고 일반적인 절차를 참조하는 각 의료기기 소프트웨어 제품의 개발 고유의 계획을 작성하는 제조업체도 있을 수 있다. 또한 각 의료기기 소프트웨어 제품의 개발에 적합하게 조정한 계획도 있을 수 있다. 기획은 개발 프로세스 수행에 필요한 상세 수준으로 지정되어야 하며, 위험에 비례해야 한다. 예를 들어 위험 수준이 높은 시스템이나 항목은 보다 엄격한 개발 프로세스의 대상이 되며, 매우 상세하게 작성되어야 한다.

기획은 개발이 진행되는 동안 계속 관찰하고 갱신해야 하는 회귀적 활동이다. 계획은 시스템을 더 잘 이해하고 시스템 개발에 어느 정도 노력이 요구됨에 따라 더 많은 양호한 정보를 채택하도록 진화할 수 있다. 예를 들어 시스템의 초기 소프트웨어 안전성 분류는 위험관리 프로세스를 활용한 결과와 소프트웨어 아키텍처의 개발에 따라 변화할 수 있다. 또한 시스템에 SOUP를 채택하도록 결정할 수도 있다. 현재 파악된 시스템의 정보와 시스템이나 시스템 내부 항목에 필요한 엄격성 수준을 반영할 수 있도록 계획을 갱신해 개발 프로세스를 적절히 관리하는 것이 중요하다.

B.5.2 소프트웨어 요구사항 분석

이 활동을 수행하려면 제조업체는 의료기기 소프트웨어에 대한 소프트웨어 요구사항을 확립하고 검증해야 한다. 구축할 내용을 결정하고 의료기기 소프트웨어가 허용되는 작동상태를 나타내는지 판단하며 완성한 의료기기 소프트웨어를 즉시 사용할 수 있다고 입증하려면 검증할 수 있는 요구사항의 확립이 필수적이다. 요구사항이 원하는 대로 구현되었는지 입증하려면 요구사항이 정확히 구현되었는지 판단하기 위한 객관적 기준을 수립할 수 있는 방식으로 표현해야 한다. 기기 위험관리 프로세스가 식별한 위험제어를 위한 요구사항을 소프트웨어에 부과한 경우 이러한 요구사항은 위험관리 방법을 소프트웨어 요구사항까지 추적할 수 있도록 소프트웨어 요구사항에서 식별할 수 있어야 한다. 모든 소프트웨어 요구사항은 요구사항과 소프트웨어 시스템 시험 사이의 추적성을 입증할 수 있는 방식으로 식별해야 한다. 일부 국가에서 규정에 의거한 승인을 받으려면 특정 규정이나 국제 표준을 준수해야 할 경우 이 준수성 요구사항은 소프트웨어 요구사항에 문서화해야 한다. 소프트웨어 요구사항에 의해 소프트웨어에 구현되는 내용이 결정되기 때문에 요구사항 분석 활동을 완료하기 전 요구사항을 평가해야 한다.

혼동이 자주 발생하는 분야는 고객 요구, 설계 입력, 소프트웨어 요구사항, 소프트웨어 기능 시방서 및 소프트웨어 설계 시방서를 구별하는 것이다. 설계 입력은 고객 요구를 공식적으로 문서화한 의료기기 요구사항으로 해석한 것이다. 소프트웨어 요구사항은 소프트웨어가 고객 요구와 설계 입력을 충족해야 하는, 공식적으로 문서화된 시방서이다. 소프트웨어 기능 시방서는 소프트웨어 요구사항에 포함되는 경우가 많으며, 많은 다른 대안도 요구사항을 충족할 수 있지만, 소프트웨어가 그 요구사항을 충족하기 위해 수행해야 할 작업을 상세하게 정의한다. 소프트웨어 설계 시방서는 요구사항과 기능 시방서 규현을 위해 소프트웨어를 설계하고 분해하는 방법을 정의한다.

일반적으로 소프트웨어 요구사항, 기능 시방서 및 설계 시방서는 하나 이상의 문서로 작성되었다. 이제는 이 정보를 공통 데이터베이스 안의 데이터 항목으로 간주할 수 있다. 각 항목에는 항목의 목적 및 데이터베이스 내 다른 항목과의 연결성을 정의하는 속성이 하나 이상 있다. 이 접근방식을 사용하면 대상 사용자(예를 들어 마케팅 담당자, 제조업체, 시험자 및 심사자)에 가장 적합한 정보를 다양하게 표현하고 인쇄할 수 있으며, 모든 요구사항의 적절한 구현을 입증하고 적용범위를 시험할 수 있다. 이 접근방식을 지원하기 위한 도구는 HTML 하이퍼링크를 사용하는 단순한 하이퍼텍스트 문서일 수도 있고 컴퓨터 지원 소프트웨어 엔지니어링(CASE) 도구와 요구사항 분석 도구처럼 복잡하지만 성능이 뛰어난 도구일 수도 있다.

시스템 요구사항 프로세스는 이 표준에서는 다루지 않는다. 단, 의료기기 기능성을 소프트웨어에 구현하고자 하는 결정은 일반적으로 시스템 설계 중 이루어진다. 일부 또는 전체 시스템 요구사항은 소프트웨어에서 구현되도록 할당된다. 소프트웨어 요구사항 분석 활동은 시스템 요구사항 프로세스에서 소프트웨어에 할당한 요구사항의 분석과 할당된 요구사항을 반영하는 종합적인 소프트웨어 요구사항을 도출하는 것으로 구성된다.

시스템 무결성 확보를 위해 제조업체는 상위 시스템 요구사항이나 소프트웨어 요구사항의 현실성 결여, 일관성 결여 또는 모호성 해결을 위해 시스템 요구사항에 대한 변경과 해명의 교섭을 위한 메커니즘을 제공해야 한다.

시스템 및 소프트웨어 요구사항의 포착과 분석 프로세스는 회귀적일 수 있다. 이 표준에 따르면 프로세스는 엄격하게 두 층으로 분리할 필요는 없다. 사실상, 시스템 아키텍처와 소프트웨어 아키텍처는 동시에 설명되는 경우가 많으며, 시스템과 소프트웨어 요구사항은 층으로 구성된 양식으로 문서화된다.

B.5.3 소프트웨어 구축 설계

이 활동을 수행하려면 제조업체가 소프트웨어의 주요 구조적 구성요인, 외부에서 확인할 수 있는 특성 및 특성 사이의 관계를 정의해야 한다. 어느 한 구성요인의 작동상태가 다른 구성요인에 영향을 줄 경우 그 작동상태를 소프트웨어 아키텍처에 기술해야 한다. 이 설명은 소프트웨어 이외의 의료기기 구성요인에 영향을 줄 수 있는 작동상태에 특히 중요하다. 위험관리 방법 구현에 있어 구축 결정은 매우 중요하다. 다른 구성요인에 영향을 줄 수 있는 구성요인의 작동상태를 이해(문서화 포함)하지 못하면 시스템이 안전하다고 표시하기는 거의 불가능에 가깝다. 소프트웨어 아키텍처는 소프트웨어 요구사항의 정확한 구현에 필요하다. 소프트웨어 아키텍처는 모든 소프트웨어 요구사항이 식별된 소프트웨어 항목으로 구현할 수 없는 한 완성되지 않는다. 소프트웨어의 설계와 구현은 아키텍처에 따라 결정되기 때문에 아키텍처를 확인해 이 활동을 완성해야 한다. 아키텍처의 확인은 일반적으로 기술 평가에 의해 이루어진다.

소프트웨어 아키텍처 활동 중 소프트웨어 항목을 분류하면 소프트웨어 프로세스의 후속 선택을 위한 기초가 마련된다. 분류의 기록은 위험관리 파일의 일부로 변경관리의 대상이 된다.

다양한 후속 사상 때문에 분류가 무효로 되는(invalidate) 경우가 있다. 예를 들면 다음과 같다.

- 시스템 시방서, 소프트웨어 시방서 또는 아키텍처의 변경
- 특히 예상할 수 없는 위험요인과 같은 오류를 위험분석에서 발견
- 특히 위험제어 방법과 같은 요구사항의 실현 불가능성 확인(discovery)

따라서 소프트웨어 아키텍처 설계 후 모든 활동 중 소프트웨어 시스템과 소프트웨어 항목(item)의 분류는 다시 평가해야 하며, 이 경우 개정이 필요할 수도 있다. 이 경우 높은 수준의 등급으로 업그레이드되기 때문에 소프트웨어 항목(item)에 프로세스를 추가로 적용하기 위한 재작업이 발생할 수 있다. 소프트웨어 형상관리 프로세스(제8조)는 필요한 모든 재작업이 식별되고 완료되는지 확인(ensure)할 경우 사용된다.

B.5.4 소프트웨어 상세 설계

이 활동을 수행하려면 제조업체가 아키텍처에 정의된 소프트웨어 항목 및 연계성을 다시 정의해 소프트웨어 유닛과 연계성을 작성해야 한다. 소프트웨어 유닛은 단일 기능이나 모듈로 간주되는 경우가 많지만 이 견해가 항상 정확한 것은 아니다. 여기에서는 소프트웨어 유닛을 더 작은 항목으로는 분할할 수 없는 소프트웨어 항목으로 정의한

다. 소프트웨어 유닛은 별도로 시험할 수 있다. 제조업체는 소프트웨어 유닛의 상세 수준을 정의해야 한다. 상세 설계는 알고리즘, 데이터 표현, 다양한 소프트웨어 유닛 사이의 연계성 및 소프트웨어와 데이터 구조 사이의 연계성을 명시한다. 또한 상세 설계는 소프트웨어 제품의 패키지 구성에도 관련된다. 소프트웨어 유닛을 정확히 구현할 수 있도록 각 소프트웨어 유닛의 설계와 연계성을 문서화해야 한다. 상세 설계에는 소프트웨어 구성에 필요한 상세 내용이 수록된다. 상세 설계는 프로그래머가 임시로 설계 결정을 하지 않아도 될 정도로 완벽해야 한다.

소프트웨어 항목은 새 소프트웨어 항목 중 극히 일부만 원래 소프트웨어 항목의 안전성 관련 요구사항을 구현하도록 분해할 수 있다. 나머지 소프트웨어 항목은 안전성 관련 기능을 구현하지 않으며, 낮은 수준의 소프트웨어 안전성 등급으로 다시 분류할 수 있다. 단, 이러한 작업에 대한 결정은 위험관리 프로세스의 일부이며, 위험관리 파일에 문서화된다.

구현은 상세 설계에 따라 결정되기 때문에 활동을 완료하기 전 상세 설계를 검증해야 한다. 상세 설계의 검증은 일반적으로 기술 검토에 의해 이루어진다. 부절 5.4.4에 따라 제조업체는 상세설계 활동의 산출을 검증해야 한다. 설계는 요구사항이 구현되는 방식을 지정한다. 설계에 결함이 있을 경우 코드는 요구사항을 정확히 구현할 수 없다.

제조업체는 안전성에 중요하다고 제조업체가 판단하는 설계 특성을 검증해야 한다. 그러한 특성의 예 몇 가지는 아래와 같다.

- 의도된 사상, 입력, 출력, 논리적 흐름, CPU 할당, 메모리 자원의 할당 및 오류와 예외 정의, 오류와 예외 확인(isolation)과 오류 복구의 구현
- 위험한 상황을 유발할 수 있는 모든 결함이 사상과 전환으로 취급되는 기본 상태의 정의
- 변수, 메모리 관리의 초기화
- 위험관리 방법에 영향을 줄 수 있는 콜드 리셋과 웜 리셋, 대기 및 기타 상태 변경

B.5.5 소프트웨어 유닛 구현 및 검증

이 활동을 수행하려면 제조업체는 소프트웨어 유닛에 대한 코드를 작성하고 검증해야 한다. 상세 설계는 원시 코드로 변환해야 한다. 코딩은 시방서 분해가 종료되고 실행 가능한 소프트웨어의 구성이 시작되는 지점을 의미한다. 원하는 코드 특성을 일관성

있게 달성하려면 코딩 표준을 사용해 선호하는 코딩 스타일을 지정해야 한다. 코딩 표준에는 이해성, 언어 사용 규칙이나 제약조건 및 복잡성 관리에 대한 요구사항이 포함된다. 각 유닛에 대한 코드를 검증해 상세 설계에서 지정한 대로 기능을 발휘하며 지정한 코딩 표준을 준수하는지 확인(ensure)해야 한다.

부절 5.5.5에 따라 제조업체는 코드를 검증해야 한다. 코드가 설계를 정확히 구현하지 못할 경우 의료기기 소프트웨어는 의도된 대로 기능을 수행하지 않는다.

B.5.6 소프트웨어 통합 및 통합 시험

이 활동을 수행하려면 개발자는 소프트웨어 유닛을 집합 소프트웨어 항목으로 통합하고 소프트웨어 항목을 높은 수준의 집합 소프트웨어 항목으로 통합하기 위한 계획을 입안하고 그러한 통합을 실행해야 하며, 소프트웨어가 의도된 대로 작동하는지 확인해야 한다.

통합에 대한 접근방식은 비점증 통합에서 점증 통합에 이르기까지 다양할 수 있다. 조립되는 소프트웨어의 특성에 따라 통합을 위해 선택한 방법이 표시된다.

소프트웨어 통합 시험은 소프트웨어 항목의 내부와 외부 연계성에서의 데이터 전달과 관리에 집중한다. 외부 연계성은 운영체제 소프트웨어가 포함된 다른 소프트웨어와 의료기기 하드웨어와의 연계성을 의미한다.

통합 시험의 엄격성과 통합 시험과 연관된 문서화의 상세 수준은 기기와 연관된 위험, 잠재적으로 위험한 기능에 대한 기기의 의존성 및 위험도가 높은 기기 기능에 있어 특정 소프트웨어의 역할에 적합해야 한다. 예를 들어 모든 소프트웨어는 시험해야 하지만 안전성에 영향을 미치는 항목은 보다 직접적이고 철저하며 상세한 시험의 대상이 된다.

해당되는 경우 통합 시험 결과는 프로그램 작동상태가 입력과 출력 도메인의 경계에 있다는 사실을 나타내며, 프로그램이 유효하지 않고(invalid) 예상할 수 없는 특수 입력에 응답한다고 확인(confirm)한다. 프로그램의 조치는 입력 또는 예상하지 못한 입력 순서의 조합이나 정의한 타이밍 요구사항이 위반되는 경우 밝혀진다. 계획의 시험 요구사항에는 통합 시험의 일환으로 수행되는 화이트박스 시험의 유형이 포함되어야 한다.

glass box, 구조적(structural) clear box 및 open box 시험이라고도 하는 White box 시험은 시험 중인 소프트웨어 항목의 내부 작용에 대한 명백한 지식을 시험데이터 선택에 사용하는 경우의 시험 기술이다. 화이트박스 시험은 출력의 관찰을 위해 소프트웨어의 특정 지식을 사용한다. 시험은 소프트웨어 항목(item)이 어떤 기능을 발휘하는지 시험자가 아는 경우에만 정밀하게 수행할 수 있다. 이 경우 시험자는 소프트웨어 항목이 의도된 목표에서 벗어나는지 여부를 확인(see)할 수 있다. 화이트박스 시험은 소프트웨어 항목(item) 구현의 시험에 집중하기 때문에 전체 시방서의 구현을 보장하지 못한다. 작동상태(behavioural), 기능적(functional), 불투명 박스(opaque-box) 및 closed-box 시험이라고도 하는 Black box 시험은 기능적 시방서에 집중하며, 구현의 모든 부분이 시험되었다고 보증하지 못한다. 따라서 블랙박스 시험은 시방서와 비교한 시험을 수행하며, 생략의 결함을 확인(discover)함으로써 시방서의 그 부분이 충족되지 않았음을 나타낸다. 화이트박스 시험은 구현과 비교한 시험을 수행하며, 임무의 결함을 확인(discover)함으로써 구현의 그 부분이 결함임을 나타낸다. 소프트웨어 제품을 완전히 시험하려면 블랙박스와 화이트박스 시험이 모두 필요할 수 있다.

5.6과 5.7에 설명한 계획 및 시험 문서는 개발이나 진화 프로토타입의 특정 단계에 관련된 개발 문서일 수 있다. 또한 단일 문서나 일련의 문서가 복수 소구분의 요구사항을 취급할 수 있도록 조합할 수도 있다. 문서의 전부 또는 일부는 하드웨어와 소프트웨어의 모든 측면을 취급하는 소프트웨어나 프로젝트 품질보증 계획 또는 종합적인 시험 계획과 같은 상위 프로젝트 문서의 일부로 구성할 수 있다. 그러한 경우 다양한 프로젝트 문서가 각 소프트웨어 통합 업무와 어떻게 관련되는지 식별하기 위한 교차 참조를 작성해야 한다.

소프트웨어 통합 시험은 시뮬레이션 환경, 실제 대상 하드웨어 또는 전체 의료기기에서 수행할 수 있다.

부절 5.6.2에 따라 제조업체는 소프트웨어 통합 활동의 산출을 검증해야 한다. 소프트웨어 통합 활동의 산출이 통합된 소프트웨어 항목이다. 이러한 통합 소프트웨어 항목은 정확하고 안전하게 기능을 수행하려면 전체 의료기기 소프트웨어에서 제대로 기능을 발휘해야 한다.

B.5.7 소프트웨어 시스템 시험

이 활동을 수행하려면 제조업체는 소프트웨어에 대한 요구사항이 성공적으로 구현되었는지 검증함으로써 소프트웨어의 기능성을 검증해야 한다.

소프트웨어 시스템 시험에 따라 지정 기능성 존재 여부가 밝혀진다. 이 시험은 소프트웨어에 대한 요구사항과 관련해 구축된 프로그램의 기능성과 성능을 검증한다.

소프트웨어 시스템 시험은 보다 효율적으로 특정 시험을 수행하고 스트레스 조건이나 결함을 유발하며 자격검증 시험의 코드 적용범위의 확대를 위해 화이트박스 시험(이전 부 참조)을 사용하는 것이 바람직할 경우에도 기능(블랙박스) 시험에 집중한다. 유형 및 시험 단계별 시험 조직에는 융통성이 부여되지만 요구사항, 위험관리, 활용성 및 시험 유형(예를 들어 결함, 설치, 스트레스)에 대한 적용범위를 입증하고 문서화해야 한다.

소프트웨어 시스템시험은 통합 소프트웨어를 시험하며 시뮬레이션 환경, 실제 대상 하드웨어 또는 전체 의료기기에서 수행할 수 있다.

소프트웨어 시스템이 변경될 경우(사소한 변경이라도) 회귀 시험의 정도(개별 변경의 시험에 국한되지 않음)를 판단하여 의도되지 않은 부작용이 발생하지 않는지 확인(ensure)해야 한다. 이 회귀 시험(소프트웨어 시스템 시험을 완전히 반복하지 않는 이론적 근거 포함)을 계획하고 문서화해야 한다.

소프트웨어 시스템 시험은 다양한 지점에서 발생하고 다양한 조직에서 수행하기 때문에 시험에 대한 책임이 분산될 수 있다. 그러나 업무의 분산, 계약에 따른 관계, 구성요인 발생원 또는 개발 환경에도 불구하고 기기 제조자는 소프트웨어가 의도한 사용에 적합하게 기능을 발휘하는지 확인(ensuring)하는 궁극적인 책임을 져야 한다.

시험 중 발견된 비정상 상태가 반복되지만 그러한 비정상 상태의 해결을 위한 결정을 하지 않은 경우 그러한 비정상 상태는 위험요인 분석과 관련해 평가함으로써 기기의 안전성에 영향을 주지 않는지 검증해야 한다. 그러한 비정상 상태의 근본 원인과 증상을 이해해야 하며, 해결하지 않기로 결정한 이론적 근거를 문서화해야 한다.

부절 5.7.4에 따라 소프트웨어 시스템 시험의 결과를 평가해 예상한 결과가 도출되는지 확인(ensure)해야 한다.

B.5.8 소프트웨어 릴리즈

이 활동을 수행하려면 제조업체는 릴리즈되는 의료기기 소프트웨어의 버전을 문서화하고 소프트웨어를 작성한 방법을 명시하며 소프트웨어 릴리즈에 대해 적합한 절차를 따라야 한다.

제조업체는 개발 프로세스를 사용해 개발한 소프트웨어가 현재 릴리즈되는 소프트웨어인지 나타낼 수 있어야 한다. 또한 제조업체는 향후 필요할 경우 소프트웨어 및 소프트웨어 생성에 사용한 도구를 검색할 수 있어야 하며, 소프트웨어의 손상과 오용을 최소화하도록 소프트웨어를 보관, 포장 및 공급해야 한다. 정의된 절차를 확립해 이러한 업무가 적절히 수행되며 일관성 있는 결과를 도출하는지 확인(ensure)해야 한다.

B.6 소프트웨어 유지보수 프로세스

B.6.1 소프트웨어 유지보수 계획 확립

소프트웨어 유지보수 프로세스는 소프트웨어 개발 프로세스와는 두 가지가 다르다.

- 제조업체는 전체 소프트웨어 개발 프로세스보다 적은 규모의 프로세스를 사용해 긴급한 문제의 대응에 있어 신속한 변경을 구현할 수 있다.
- 릴리즈된 제품과 관련해 소프트웨어 문제 보고서에 대응함에 있어 제조업체는 문제 해결은 물론 지역 법규를 준수해야 한다(주로 제출된 문제 데이터를 취합하고 문제에 관해 사용자와 규제자와 의사소통하기 위한 전향적 조사 계획을 실행한다).

부절 6.1에 따라 이들 프로세스는 유지보수 계획에 확립되어야 한다.

이 활동을 수행하려면 제조업체는 유지보수 활동과 업무를 구현하는 절차를 작성하거나 식별해야 한다. 시정조치를 구현하고 유지보수 중 변경을 관리하며 개정 소프트웨어의 릴리즈를 관리하려면 제조업체는 문제 보고서와 사용자의 요청을 기록하고 문제를 해결해야 하며, 의료기기 소프트웨어에 대한 수정을 관리해야 한다. 문제가 발생하거나 개선 또는 적용의 필요성 때문에 의료기기 소프트웨어의 코드 및 관련 문서가 수정되는 경우 이 프로세스가 시작된다. 목적은 릴리즈된 의료기기 소프트웨어 무결성을 보존하면서 소프트웨어를 수정하는 것이다. 이 프로세스에는 원래는 릴리즈되지 않은 환경이나 플랫폼에 대한 의료기기 소프트웨어의 마이그레션이 포함된다. 이 조항에 규정한 활동은 유지보수 프로세스에 관한 것이다. 단, 유지보수 프로세스에는 이 표준의 다른 프로세스도 사용될 수 있다.

제조업체는 유지보수 프로세스의 활동과 업무를 수행하는 방법을 계획해야 한다.

B.6.2 문제 및 수정 분석

이 활동을 수행하려면 유지보수자는 그 영향에 대한 피드백을 분석하고, 보고 받은 문제를 검증하며 수정 옵션의 구현을 고려하고 선택하며 승인을 확보해야 한다. 문제 및 기타 변경 요청은 의료기기의 성능 안전성이나 규제적 허가에 영향을 미칠 수 있다. 문제 보고서 때문에 영향이 발생하는지의 여부 또는 문제를 해결하고 요청을 구현하기 위한 수정에 의해 영향이 발생하는지 여부의 판단에 분석이 필요하다. 소프트웨어 유지보수 활동의 일환으로 구현되는 소프트웨어 변경에 의해 기기에 내장된 위험관리 수단이 변경되거나 수정되는지 여부를 추적이나 회귀 분석을 통해 검증하는 것이 특히 중요하다. 또한 이전에는 위험요인을 발생하거나 위험을 최소화하지 않았던 소프트웨어에 수정된 소프트웨어가 위험요인을 발생하거나 위험을 최소화하지 않는다는 사실의 검증도 중요하다. 소프트웨어 항목(item)의 소프트웨어 안전성 분류는 소프트웨어 수정으로 위험요인이 발생하거나 위험이 최소화할 경우 변경될 수 있다.

소프트웨어 유지보수(제6조)와 소프트웨어 문제해결(제9조)을 구별하는 것이 중요하다.

소프트웨어 유지보수 프로세스의 초점은 소프트웨어 제품의 릴리즈 후 발생하는 피드백에 적절히 대응하는 것이다. 의료기기의 일부인 소프트웨어 유지보수 프로세스는 다음을 확인(ensure)해야 한다.

- 안전관련 문제보고서가 처리되었으며, 해당 규제당국과 해당 사용자에게 보고되었다.
- 문제의 해결과 후속 문제 방지를 보장하는 공식 관리장치로 수정 후 소프트웨어 제품이 타당성 재확인되거나 다시 릴리즈되었다.
- 제조업체가 영향을 받을 수 있는 다른 소프트웨어 제품을 고려하며, 적절한 조치를 취한다.

소프트웨어 문제해결의 초점은 다음과 같은 종합적인 관리 시스템을 운영하는 것이다.

- 문제 보고서를 분석하고 문제의 모든 측면을 식별한다.
- 변경의 숫자를 결정하고 부작용을 모두 식별한다.
- 위험관리 파일을 포함해 소프트웨어 형상 항목의 일관성을 유지하는 동시에 변경을 구현한다.
- 변경의 구현을 검증한다.

소프트웨어 유지보수 프로세스는 소프트웨어 문제해결 프로세스를 사용한다. 소프트웨어 유지보수 프로세스는 문제 보고서에 관한 상위 차원의 결정을 취급하며(문제가 존재하는지의 여부, 문제가 안전성에 상당한 영향을 미치는지의 여부, 필요한 변경 및 변경의 구현 시점), 소프트웨어 문제해결 프로세스를 사용해 모든 암시를 파악하고, 변경할 필요가 있는 모든 형상 항목(item)과 필요한 모든 검증 단계를 식별하는 변경 요청을 생성한다.

B.6.3 수정 구현

이 활동을 수행하려면 제조업체는 수정을 위해 확립한 프로세스를 사용해야 한다. 유지보수 프로세스가 정의되어 있지 않은 경우 해당 개발 프로세스 업무를 사용해 수정을 수행할 수도 있다. 또한 제조업체는 수정에 의해 의료기기 소프트웨어의 다른 부분에 부작용이 발생하지 않는다고 확인(ensure)해야 한다. 의료기기 소프트웨어를 새로 개발한 소프트웨어로 취급하지 않는 한 전체 의료기기 소프트웨어에 대한 수정의 영향을 분석해야 한다. 수정되지 않은 의료기기 소프트웨어 부분이 수정 전과 같은 성능을 발휘할 수 있도록 수행되는 회귀 시험의 정도를 정당화하는 이론적 근거를 마련해야 한다.

B.7 소프트웨어 위험관리 프로세스

소프트웨어 위험관리은 전체적인 의료기기 위험관리의 일부이며, 따로 취급할 수는 없다. 이 표준을 사용하려면 ISO 14971을 준수하는 의료기기 위험관리 프로세스를 사용해야 한다. ISO 14971에 정의된 위험관리은 특히 의료기기 사용에 연관된 위험의 효과적인 경영을 위한 체제를 취급한다. 위험분석 중 식별한 각 위험요인에 관련해 식별한 위험의 관리에 ISO 14971 일부가 관련된다. 이 표준에 규정한 소프트웨어 위험관리 프로세스의 목적은 위험 분석 중 위험한 상황을 잠재적으로 발생할 것으로 식별된 소프트웨어 또는 의료기기 위험 관리에 사용되는 소프트웨어가 포함된 소프트웨어를 위한 위험관리의 추가 요구사항을 제공하는 것이다. 소프트웨어 위험관리 프로세스는 다음과 같은 두 가지 이유 때문에 이 표준에 포함되었다.

a) 이 표준의 대상 사용자는 담당 분야인 소프트웨어의 위험관리 수단을 위한 최소한의 요구사항을 이해해야 한다.
b) 이 표준에 규정 참조로 제공되는 일반 위험관리 표준 ISO 14971은 소프트웨어에 관한 위험관리 및 소프트웨어 개발 수명주기에서의 위험관리를 제대로 취급하지 못한다.

소프트웨어 위험관리은 전체적인 의료기기 위험관리의 일부이다. 소프트웨어 위험

관리 활동에 필요한 계획, 절차 및 문서는 일련의 독립된 문서이거나 단일 문서일 수 있으며, 이 표준의 모든 요구사항이 충족될 경우 의료기기 위험관리 활동과 문서에 통합할 수도 있다.

B.7.1 위험한 상황에 대한 소프트웨어 영향의 분석

의료기기 위험요인 분석에 따라 위험한 상황과 그에 대응하는 위험관리 방법이 파악되어 그러한 위험한 상황의 발생가능성과 심각도를 허용되는 수준으로 줄일 수 있을 것으로 기대된다. 또한 위험관리 방법은 그러한 위험관리 방법을 구현할 것으로 예상되는 소프트웨어 기능에 할당될 것으로 예상된다.

단, 소프트웨어 아키텍처를 생성할 때까지 모든 의료기기의 위험한 상황이 식별될 것으로 예상되지는 않는다. 그러한 경우 소프트웨어 기능이 소프트웨어 요소에서 구현되는 방법과, 소프트웨어 기능에 할당된 실질적인 위험관리 방법을 평가할 수 있는 방법을 파악할 수 있다. 이 경우 의료기기 위험요인 분석은 다음을 포함하도록 개정되어야 한다.

- 개정된 위험한 상황
- 개정된 위험관리 방법 및 소프트웨어 요구사항
- 예를 들어 인간에 의해 발생되는 요인에 관련된 위험한 상황과 같이, 소프트웨어에서 발생하는 새로운 위험한 상황

소프트웨어 아키텍처에는 소프트웨어 요소가 불안전한 방식으로는 상호 작용할 수 없도록 소프트웨어 요소를 분리하기 위한 신뢰성 있는 전략이 포함되어야 한다.

B.8 소프트웨어 형상관리 프로세스

소프트웨어 구성관리 프로세스는 문서를 포함해 시스템의 소프트웨어 항목을 식별하고 정의하며 기준을 정하고, 수정을 관리하고 수정을 관리하고 항목을 릴리즈하며, 항목의 상태와 변경 요청을 기록하고 보고하기 위해 소프트웨어 수명주기 전체에 행정적 및 기술적 절차를 적용하는 프로세스이다. 소프트웨어 구성관리는 소프트웨어 항목을 재창출하고 구성부품을 식별하며 항목에 이루어진 변경의 이력 제공에 필요하다.

B.8.1 형상 식별

이 활동을 수행하려면 제조업체는 소프트웨어 항목과 버전을 별도로 식별해야 한다. 의료기기 소프트웨어에 포함되는 소프트웨어 형상 항목과 버전의 식별에 이 식별이 필요하다.

B.8.2 변경관리

이 활동을 수행하려면 제조업체는 소프트웨어 형상 항목의 변경을 관리하고, 변경 요청을 식별하며 성격에 관한 문서를 제공하는 정보를 기록해야 한다. 소프트웨어 형상 항목에 무단 또는 의도되지 않은 변경이 발생하지 않으며 승인된 변경요청이 구현되고 완전히 검증되도록 보장하려면 이 활동이 필요하다.

변경 요청은 소프트웨어 형상관리 계획에 따라 변경관리위원회나 관리자 또는 선임 기술자가 승인할 수 있다. 승인된 변경요청은 소프트웨어의 실제 수정과 검증까지 추적할 수 있어야 한다. 요구사항은 실질적인 각 변경이 변경 요청에 연계되며, 변경 요청 승인을 나타내는 문서가 존재해야 한다는 것이다. 문서는 변경관리위원회의 의사록, 승인 서명 또는 데이터베이스의 기록일 수 있다.

B.8.3 형상 상태 설명

이 활동을 수행하려면 제조업체는 소프트웨어 형상 항목(item) 이력의 기록을 유지해야 한다. 변경이 이루어진 시점과 이유 판단에 이 활동이 필요하다. 소프트웨어 형상 항목에 승인된 수정 내용만 수록되는지 확인(ensure)하려면 이 정보에 액세스해야 한다.

B.9 소프트웨어 문제해결 프로세스

소프트웨어 문제해결 프로세스는 개발, 유지보수 또는 기타 프로세스 실행 중 확인(discovered)된 것을 포함해 성격이나 발생원에 관계 없이 문제(비준수성 포함)를 분석하고 해결하는 프로세스이다. 이 프로세스의 목적은 확인(discovered)된 문제를 분석하고 해결하며 경향을 확인(recognized)할 수 있도록 적시에 적절하고 문서화된 방법을 제공하는 것이다. 소프트웨어 엔지니어링 문서에서는 이 프로세스를 '결함 추적'이라고 하는 경우가 많다. 이를 ISO/IEC 12207[9]와 IEC 60601-1-4[2] 부록 1에서는 "문제해결"이라고 한다. 이 표준에서는 "소프트웨어 문제해결"이라고 칭하기로 한다.

이 활동을 수행하려면 제조업체는 소프트웨어 문제나 비준수성이 식별된 경우 문제해결 프로세스를 사용해야 한다. 발견된 문제가 안전성(ISO 14971에 명시된)에 관련되었는지 분석하고 평가하려면 이 활동을 수행해야 한다.

5.1에서 요구되는 것과 같은 소프트웨어 개발 계획이나 절차에는 문제나 비준수성을 취급할 방법이 설명되어야 한다. 여기에는 수명주기의 각 단계에서 공식적으로 문서화되는 소프트웨어 문제해결 프로세스의 측면은 물론 문제 및 비준수성을 소프트웨어 문제해결 프로세스에 입력해야 하는 시점의 지정이 포함된다.

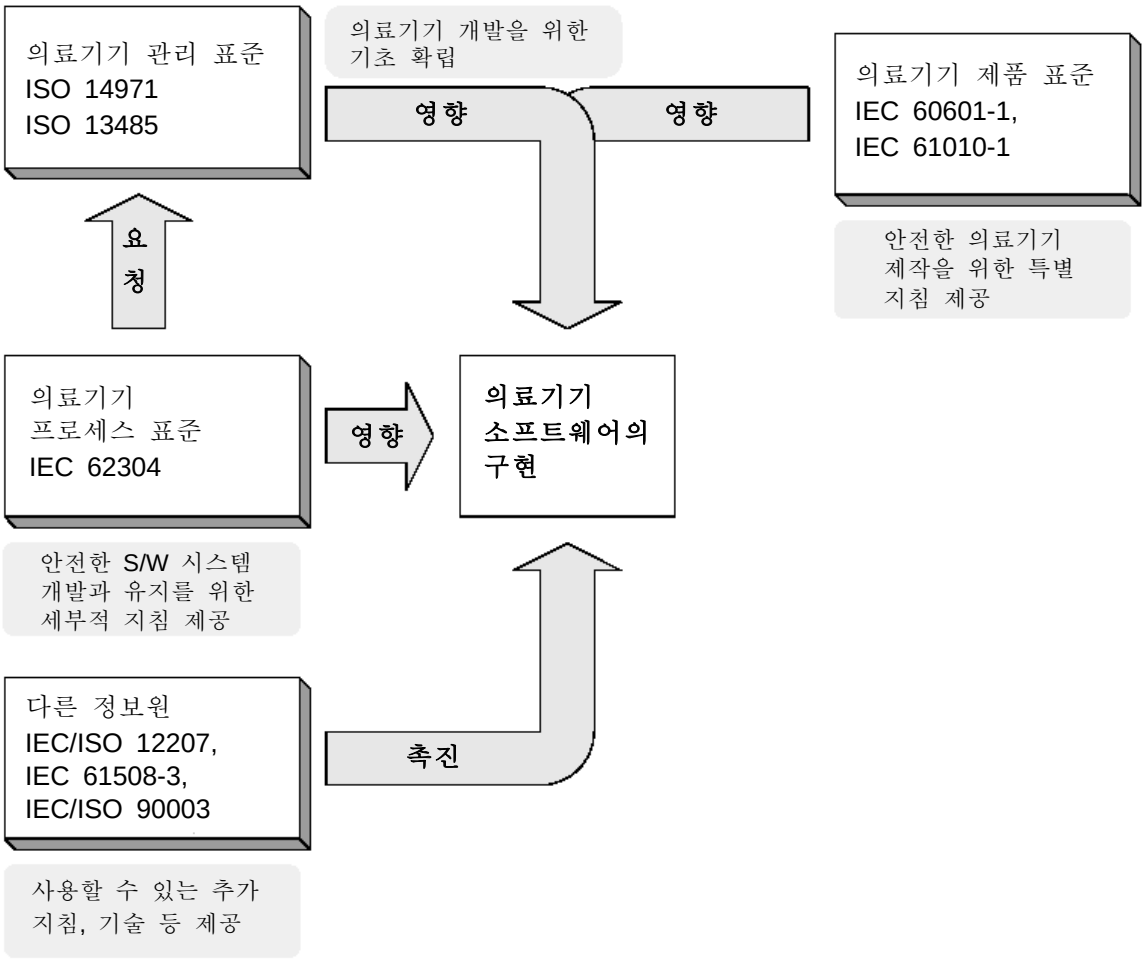
부록 C (참조용) 다른 표준과의 관계

C.1 일반

이 표준은 의료기기 소프트웨어 개발과 유지보수에 적용된다. 소프트웨어는 의료기기의 서브 시스템 또는 그 자체가 의료기기로 간주된다. 이 표준은 의료기기를 개발할 때 다른 해당 표준과 함께 사용해야 한다.

ISO 13485[7](C.2 및 부록 D 참조)과 ISO 14971(부록 C.3 참조)과 같은 의료기기 관리 표준은 제품을 개발하는 조직을 위한 기초를 형성하는 관리 환경을 제공한다. IEC 60601-1[1](부록 C.4참조)과 IEC 61010-1[4](부록 C.5 참조)와 같은 안전성 표준은 안전한 의료기기 생성에 특별히 적용되는 지침을 제공한다. 소프트웨어가 이러한 의료기기의 일부일 경우 IEC 62304가 안전한 의료기기 소프트웨어 개발과 유지에 필요한 보다 상세한 지시를 제공한다. ISO/IEC 12207[9](부록 C.6 참조), IEC 61508-3[3](부록 C.7 참조) 및 ISO/IEC 90003[11]과 같은 많은 다른 표준은 IEC 62304의 요구사항의 구현에 사용할 수 있는 방법, 툴 및 기술로서 간주할 수 있다. 그림 C.1에는 이러한 표준 사이의 관계가 표시된다.

다른 표준의 조항이나 요구사항이 인용되는 경우 인용된 항목에 정의된 용어는 이 표준에 정의된 용어가 아니라 다른 표준에 정의된 용어이다.



[그림 C.1] IEC 62304에 대한 핵심 의료기기 표준의 관계

C.2 ISO 13485와의 관계

이 표준에 따라 제조업체는 품질경영시스템을 채택해야 한다. 제조업체가 ISO 13485[7]을 사용할 경우 ISO 62304의 요구사항은 표 C.1에 표시된 ISO 13485의 요구사항 일부와 직접 관련된다.

IEC 62304 조항	ISO 13485:2003의 관련 조항
5.1 소프트웨어 개발 기획	7.3.1 설계 및 개발 기획
5.2 소프트웨어 요구사항 분석	7.3.2 설계 및 개발 입력
5.3 소프트웨어 아키텍처 설계	
5.4 소프트웨어 상세 설계	
5.5 소프트웨어 유닛 구현 및 검증	
5.6 소프트웨어 통합 및 통합 시험	
5.7 소프트웨어 시스템 시험	7.3.3 설계및 개발 출력 7.3.4 설계 및 개발 검토
5.8 소프트웨어 릴리즈	7.3.5 설계 및 개발 검증 7.3.6 설계 및 개발 밸리데이션
6.1 소프트웨어 유지보수 계획 확립	7.3.7 설계 및 개발 변경 관리
6.2 문제 및 수정 분석	
6.3 수정 구현	7.3.5 설계 및 개발 검증 7.3.6 설계 및 개발 밸리데이션
7.1 위험한 상황에 대한 소프트웨어 영향의 분석	
7.2 위험관리 방법	
7.3 위험관리 방법의 검증	
7.4 소프트웨어 변경의 위험관리	
8.1 형상 식별	7.5.3 식별및 추적성
8.2 변경관리	7.5.3 식별 및 추적성
8.3 형상 상태 설명	
9 소프트웨어 문제해결 프로세스	

[표 C.1] ISO 13485:2003과의 관계

C.3 ISO 14971와의 관계

표 C.2에는 ISO 14971에 의해 요구되는 위험관리 프로세스에 대한 요구사항을 IEC 62304가 강조하는 영역이 표시된다.

ISO 14971:2000 조항	ISO 62304의 관련 조항
4.1 위험 평가 절차	
4.2 의료기기 안전 관련 의도한 용도/목적 및 특성의 식별	
4.3 식별 또는 예측가능한 위험요인의 식별	7.1 위험한 상황에 대한 소프트웨어 영향 분석
4.4 각 위험요인에 대한 위험 추정	4.3 소프트웨어 안전성 분류
5 위험 평가	
6.1 위험 축소	
6.2 옵션 분석	7.2.1 위험관리 방법 정의
6.3 위험관리 방법의 구현	7.2.2 소프트웨어에 구현된 위험관리 방법 7.3.1 위험관리 방법 검증
6.4 잔류 위험 평가	
6.5 위험/장점 분석	
6.6 기타 발생 위험요인	7.3.2 사상의 새로운 발생순서 문서화
6.7 위험 평가 완료	
7 전체 잔류 위험 평가	
8 위험관리 보고서	7.3.3 추적성 문서화
9 생산 후 정보	7.4 소프트웨어 변경의 위험관리

[표 C.2] ISO 14971:2000과의 관계

C.4 IEC 60601-1:2004의 PEMS 요구사항에 대한 관계

C.4.1 일반

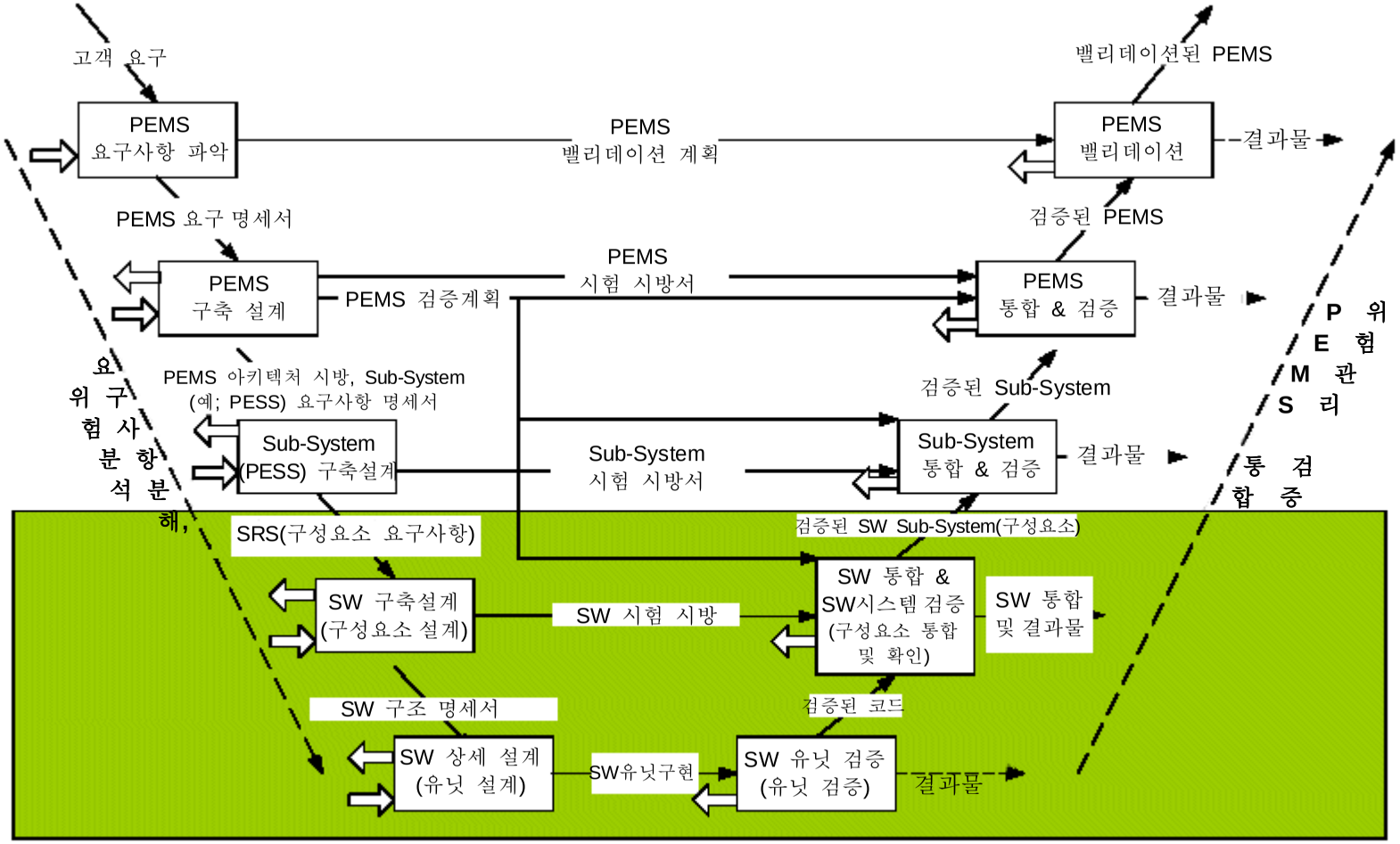
소프트웨어에 대한 요구사항은 프로그램이 가능한 전기 의료기기(PEMS)에 대한 요구사항의 하위 요구사항이다. 이 표준은 PEMS을 위한 IEC 60601-1[1] 요구사항에 추가되지만 그 요구사항들과 호환되지 않는 소프트웨어에 대한 요구사항에 관한 것이다. PEMS에는 소프트웨어가 아닌 요소도 포함되므로 PEMS에 대한 IEC 60601-1의 요구사항이 모두 이 표준에서 취급되는 것은 아니다.

C.4.2 PEMS 개발과의 소프트웨어 관계

PEMS 개발 중 발생하는 현상의 설명을 위해 그림 C.2에 표시된 V 모델을 사용하면 소프트웨어 요구사항의 지정에서부터 소프트웨어 항목을 소프트웨어 시스템에 통합할 때까지 PEMS 요소 수준에서 이 소프트웨어 수준의 요구사항이 적용되는 것을 확인할 수 있다. 이 소프트웨어 시스템은 PEMS의 일부인 프로그램이 가능한 전기 서브 시스템(PESS)의 일부이다.

C.4.3 개발 프로세스

이 표준의 소프트웨어 개발 프로세스(제5조)와의 준수성을 유지하려면 소프트웨어 개발 계획을 지정하고 그를 따라야 한다. 여기에는 특정 수명주기 모델의 사용은 필요하지 않지만 계획에는 특정 아키텍처와 속성이 포함되어야 한다.



Key:
상자 안의 내용은 대표적인 개발 수명주기 활동을 나타낸다.
실선 화살표는 활동의 입출력으로 전달되는 대표적인 산출물을 나타낸다.
점선 화살표는 위험관리 파일에 대한 산출물을 나타낸다.

⇨ 문제해결 프로세스의 출력
⇦ 문제해결 프로세스에 대한 입력

[그림 C.2] V 모델 일부로서의 소프트웨어

이러한 요구사항은 개발 수명주기, 요구사항 시방서, 아키텍처, 설계와 구현 및 검증에 대한 IEC 60601-1의 PEMS 요구사항에 관련된다. 이 표준의 요구사항은 IEC 60601-1보다 소프트웨어 개발에 관해 더 많은 정보를 제공한다.

C.4.4 유지보수 프로세스

이 표준의 소프트웨어 유지보수 프로세스(제6조)와의 준수성을 유지하려면 소프트웨어를 변경할 때 절차를 확립하고 그를 따라야 한다. 이러한 요구사항은 PEMS의 수정에 대한 IEC 60601-1의 요구사항에 해당된다. 소프트웨어 유지보수를 위한 이 표준의 요구사항은 소프트웨어 유지보수를 위해 수행하는 것과 관련해 IEC 60601-1의 PEMS 수정에 대한 요구사항보다 더 많은 정보를 제공한다.

C.4.5 기타 프로세스

이 표준의 기타 프로세스는 IEC 60601-1의 PEMS에 대한 유사한 요구사항 이외의 소프트웨어 요구사항을 추가로 지정한다. 대부분의 경우 이 표준의 프로세스가 확장되는 IEC 60601-1의 PEMS에 대한 일반적인 요구사항이 있다.

이 표준의 소프트웨어 위험관리 프로세스는 IEC 60601-1의 PEMS에 추가로 식별된 위험관리 요구사항에 해당되며, 이 표준의 소프트웨어 문제해결 프로세스는 IEC 60601-1의 PEMS에 대한 문제해결 위험관리 요구사항에 해당된다.

이 표준의 소프트웨어 형상관리 프로세스는 문서화를 제외하고 IEC 60601-1의 PEMS에 대해 제시되지 않는 요구사항을 추가로 지정한다.

C.4.6 IEC 60601-1의 PEMS 요구사항의 취급 범위

표 C.3에는IEC 60601-1의 PEMS 요구사항 및 이 표준의 해당 요구사항이 수록된다.

IEC 60601-1:005의 PEMS 요구사항	PEM의 소프트웨어 서브 시스템에 관련된 IEC 62304의 요구사항
14.1 일반 이 요구사항은 다음 경우를 제외하고 PEMS에 적용된다. - PESS가 기본 안전성이나 필수 성능을 제공하지 않는 경우 - ISO 14971 적용 결과 PESS 결함으로도 허용할 수 없는 위험이 발생하지 않음을 확인(demonstrate)될 경우	4.3 소프트웨어 안전성 분류 IEC 60601-1의PEMS 요구사항은 소프트웨어 안전성 등급 B와 C에만 적용된다. 이 표준에는 소프트웨어 안전성 등급 A에 대한 요구사항이 일부 포함된다.
14.2 문서화 ISO 14971에서 요구되는 기록과 문서 이외 제14조	4.2 위험관리

적용에 따라 발생하는 문서를 유지해야 하며, 그러한 문서는 위험관리 파일의 일부를 형성해야 한다.	
제14조에 따라 요구되는 문서는 공식 문서관리 절차에 따라 검토, 승인, 발급 및 변경되어야 한다.	5.1 소프트웨어 개발 기획 소프트웨어 개발기획 활동의 특정 요구사항 이외에 ISO 14971에 따라 위험관리 파일의 일부인 문서가 유지되어야 한다. 또한 품질 시스템이 필요로 하는 문서의 경우 ISO 13485[7]에 따라 문서를 관리해야 한다.
14.3 위험관리 계획 ISO 14971의3.5에 따라 요구되는 위험관리 계획에는 PEMS 밸리데이션 계획에 대한 기준도 포함되어야 한다(14.11 참조).	특별히 필요하지 않다. 특정 소프트웨어 밸리데이션 계획이 없다. PEMS 밸리데이션 계획 시스템 차원에서 이루어지므로 이 소프트웨어 표준의 범위에 포함되지 않는다. 이 표준에는 위험관리 방법에 대한 특정 소프트웨어 원인에 대한 위험요인부터 위험관리 방법의 검증까지의 추적성이 필요하지 않다(7.3 참조).
14.4 PEMS 개발 수명주기 PEMS 개발 수명주기를 문서화해야 한다.	5.1 소프트웨어 개발 기획 5.1.1 소프트웨어 개발 계획 소프트웨어 개발 계획이 처리하는 항목(item)은 소프트웨어 개발 수명주기를 구성한다.
PEMS 개발 수명주기에는 정의된 일정이 포함되어야 한다.	
각 일정에서 완료해야 할 활동과 그러한 활동에 적용되는 검증 방법을 정의해야 한다.	5.1.6 소프트웨어 검증 기획 검증 업무, 일정 및 인수 기준을 계획해야 한다.
각 활동은 입력과 출력을 포함해 정의해야 한다.	5.1.1 소프트웨어 개발 계획 활동은 이 표준에서 정의된다. 제작할 문서가 각 활동에서 정의된다.
각 일정은 일정 전 완료해야 하는 위험관리 활동을 식별해야 한다.	
PEMS 개발 수명주기는 활동, 일정 및 스케줄이 자세히 규정된 계획을 작성함으로써 특정 개발에 적합하게 조정해야 한다.	5.1.1 소프트웨어 개발 계획 이 표준에 따라 개발 수명주기를 개발 계획에 문서화할 수 있다. 즉, 개발 계획에는 조정된 개발 수명주기가 포함된다.
PEMS 개발 수명주기에는 문서화 요구사항이 포함되어야 한다.	5.1.1 소프트웨어 개발 계획 5.1.8 문서화 기획
14.5 문제해결 필요한 경우 PEMS 개발 수명주기의 모든 단계와 활동 사이의 문제해결에 대해 문서화한 시스템을 개발하고 유지해야 한다.	9 소프트웨어 문제해결 프로세스
제품 유형에 따라 문제해결 시스템은 다음과 같을 수 있다. - PEMS 개발 수명주기 일환으로 문서화된다. - 기본 안전성이나 필수적 성능에 영향을 주는 잠재적 또는 기존 문제를 보고한다. - 연관 위험에 대한 각 문제의 평가가 포함된다. - 마감해야 할 현안에 대하여 충족해야 할 기준을 식별한다. - 각 문제해결을 위해 취해야 할 조치를 식별한다.	5.1.1 소프트웨어 개발 계획 9.1 문제 보고서 작성
14.6 위험관리 프로세스	7 소프트웨어 위험관리 프로세스
14.6.1 확인(known)되거나 예측할 수 있는 위험요인	7.1 위험한 상황에 대한 소프트웨어 영향의 분석

의 식별 확인(known)되거나 예측 가능한 위험요인의 목록을 작성할 때 제조업체는 네트워크/데이터 결합, 제3자 원본 구성요소와 레거시 서브 시스템에 관련된 것을 포함해 PEMS의 소프트웨어와 하드웨어 측면에 관련된 위험요인을 고려해야 한다.	이 표준에는 네트워크/데이터 결합은 언급되지 않는다.
14.6.2 위험관리 각 위험관리 방법을 구현하려면 적절하게 밸리데이션된 도구와 절차를 선택하고 식별해야 한다. 이러한 도구와 절차는 각 위험관리 방법이 식별된 위험을 만족스러운 수준으로 축소하는지 확인(assure)할 수 있도록 적절해야 한다.	5.1.4 소프트웨어 개발 표준, 방법 및 도구 기획 이 표준은 각 위험관리 방법이 아닌, 개발에 일반적으로 사용되는 특정 도구와 방법을 식별해야 한다.
14.7 소프트웨어 시방서 PEMS와 그 서브 시스템(예를 들어 PESS에 대한)의 경우 문서화한 요구사항 시방서가 있어야 한다.	5.2 소프트웨어 요구사항 분석 이 표준은 PEMS의 소프트웨어 서브 시스템만 취급한다.
시스템이나 서브 시스템에 대한 요구사항 시방서에는 그 시스템이나 서브 시스템이 구현한 핵심 성능 및 위험관리 방법이 포함되어야 한다.	5.2.1 시스템 요구사항에서 소프트웨어 요구사항을 정의하고 문서화 5.2.2 소프트웨어 요구사항 내용 5.2.3 소프트웨어 요구사항에 위험관리 방법 포함 이 표준에 따르면 필수 성능과 위험관리 방법을 다른 요구사항과 구별할 필요는 없지만 모든 요구사항은 각각 식별해야 한다.
14.8 아키텍처 PEMS과 각 서브 시스템의 경우 아키텍처는 요구사항 시방서를 충족하도록 지정되어야 한다.	5.3 소프트웨어 구축 설계
필요한 경우 위험을 허용 수준으로 축소하려면 아키텍처 시방서는 아래의 내용을 사용해야 한다. a) 무결성이 높은 특성의 구성요소 b) 결함이 발생하지 않는 기능 c) 중복성 d) 다양성 e) 기능성의 분할 f) 방어 설계. 일례로 가용 출력 파워를 제한하거나 작동기 왕복운동을 제한하는 방법을 채택함으로써 잠재적으로 위험한 영향을 제한.	5.3.5 위험관리에 필요한 분리 식별 분할은 식별된 유일한 기술이며, 분할의 무결성을 보장하는 방법을 나타내야 한다는 요구사항이 있기 때문에 유일하게 식별된다.
아키텍처 시방서는 다음을 고려해야 한다. g) 위험관리 방법을 PEMS의 서브시스템과 구성요소에 할당 h) 구성요소의 결함 모드 및 그 영향 i) 결함의 공통 원인 j) 시스템 결함 k) 시험 간격과 진단 적용범위 l) 유지보수 가능성 m) 타당한 수준으로 예측할 수 있는 오용의 방지 n) 해당되는 경우 네트워크/데이터 결합 시방서	이 표준에는 포함되지 않는다.
14.9 설계 및 구현 필요한 경우 설계는 각각 설계와 시험 시방서가 있는 서브 시스템으로 분해해야 한다.	5.4 소프트웨어 상세 설계 5.4.2 각 소프트웨어 유닛에 대해 상세 설계 개발 이 표준에 따르면 상세 설계에는 시험 시방서가 필요하지 않다.
설계환경에 관련된 설명 데이터를 위험관리 파일이 포함해야 한다.	5.4.2 각 소프트웨어 유닛에 대해 상세 설계 개발
14.10 검증	5.1.6 소프트웨어 검증 기획

기본 안전성, 필수 성능 또는 위험관리 방법을 구현하는 모든 기능을 검증해야 한다.	각 활동에 검증이 필요하다.
이러한 기능을 검증해야 하는 방법을 나타내기 위한 검증 계획을 작성해야 한다. 계획에는 다음과 같은 내용이 포함되어야 한다. - 각 기능에 대해 검증을 수행해야 하는 시점 - 검증 전략, 활동, 기술 및 검증을 수행하는 담당자의 적절한 독립성 수준의 선택과 문서화 - 검증 도구의 선택과 활용 - 검증을 위한 적용 범위 기준	5.1.6 소프트웨어 검증 기획 이 표준에는 담당자 독립성은 포함되지 않는다. 이 독립성은 ISO 13485에서 취급한다.
검증 계획에 따라 검증을 수행해야 한다. 검증 활동의 결과는 문서화해야 한다.	대부분의 활동에는 검증 요구사항이 있다.
14.11 PEMS 밸리데이션 PEMS 밸리데이션 계획에는 기본 안전성과 필수 성능의 밸리데이션이 포함되어야 하며, PEMS의 의도되지 않은 기능 여부를 점검해야 한다.	이 표준은 소프트웨어 밸리데이션을 처리하지 않는다. PEMS 밸리데이션은 시스템 차원 활동이므로 이 표준의 범위에 포함되지 않는다.
PEMS 밸리데이션은 PEMS 밸리데이션 계획에 따라 수행해야 한다. PEMS 밸리데이션 활동의 결과는 문서화해야 한다.	이 표준은 소프트웨어 밸리데이션을 처리하지 않는다. PEMS 밸리데이션은 시스템 차원 활동이므로 이 표준의 범위에 포함되지 않는다.
PEMS 밸리데이션에 관해 전반적인 책임을 지는 담당자는 설계 팀과 독립되어야 한다. 제조업체는 독립성 수중에 대한 이론적 근거를 문서화해야 한다.	이 표준은 소프트웨어 밸리데이션을 처리하지 않는다. PEMS 밸리데이션은 시스템 차원 활동이므로 이 표준의 범위에 포함되지 않는다.
설계 팀의 구성원은 자신의 설계에 대한 PEMS 밸리데이션을 담당할 수 없다.	이 표준은 소프트웨어 밸리데이션을 처리하지 않는다. PEMS 밸리데이션은 시스템 차원 활동이므로 이 표준의 범위에 포함되지 않는다.
PEMS 밸리데이션 팀 구성원과 설계 팀 구성원의 모든 전문적 관계를 위험관리 파일에 문서화해야 한다.	이 표준은 소프트웨어 밸리데이션을 처리하지 않는다. PEMS 밸리데이션은 시스템 차원 활동이므로 이 표준의 범위에 포함되지 않는다.
PEMS 밸리데이션의 방법과 결과에 대한 참고문서가 위험관리 파일에 포함되어야 한다.	이 표준은 소프트웨어 밸리데이션을 처리하지 않는다. PEMS 밸리데이션은 시스템 차원 활동이므로 이 표준의 범위에 포함되지 않는다.
14.12 수정 설계 일부나 전부가 이전 설계의 수정에 의한 것일 경우 수정에 의한 설계가 새로운 설계인 것으로 간주해 이 조항을 전부 적용하거나, 이전 설계 문서의 연속 유효성을 문서화된 수정/변경 절차에 따라 평가해야 한다.	6 소프트웨어 유지보수 프로세스 이 표준은 소프트웨어 유지보수를 기획해야 하고 수정의 구현이 소프트웨어 개발 프로세스나 확립된 소프트웨어 유지보수 프로세스를 사용해야 한다는 접근방식을 선택한다.
14.13 다른 장치에 대한 Network/Data 연계에 의한 PEMS 연결 PEMS 제조업체의 관리 영역을 벗어나는 다른 장치와의 네트워크/데이터 연계를 사용해 PEMS를 연결할 경우 기술 설명은 다음과 같아야 한다. a) 의도된 사용의 달성을 위해 PEMS에 필요한 네트워크/데이터 연계의 특성을 명시한다. b) 명시한 특성 제공을 위해 네트워크/데이터 연계의 실패에서 발생하는 위험한 상황을 나열한다. c) 해당 조직에 다음을 알린다. - 다른 장치가 포함된 네트워크/Data 연계에 대한 PEMS 연결에 따라 이전에 확인되지 않은(unidentified) 위험이 환자, 운영자 또는 제3자에게 발생할 수 있다.	이 표준에는 네트워크/데이터 연계 요구사항은 포함되지 않는다.

- 해당 조직은 이러한 위험을 식별, 분석, 평가 및 관리해야 한다. - 네트워크/Data 연계에 대한 후속 변경 때문에 새로운 위험이 발생할 수 있으므로 분석을 추가로 수행해야 한다. - 네트워크/데이터 연계의 변경에는 다음이 포함된다. - 네트워크/Data 형상의 변경 - 네트워크/Data 연계에 추가 항목 연결 - 네트워크/Data 연계에서 항목 분리 - 네트워크/Data 연계에 연결된 장치의 갱신 - 네트워크/Data 연계에 연결된 장치의 업그레이드	

[표 C.3] IEC 60601-1과의 관계

C.4.7 IEC 60601-1-4의 요구사항에 대한 관계

IEC 60601-1-4는 IEC 60601-1:2005에 대한 전환 기간이 완료될 때까지 계속 사용된다.

표 C.4에는IEC 60601-4[2]의 요구사항 및 이 표준의 해당 요구사항이 수록된다. 이 표는 이 표준의 관련 요구사항이 IEC 60601-1-4의 요구사항을 모두 처리한다고 명시하지 않는다. 60601-1-4 요구사항의 대부분은 ISO 14971과의 준수성에 따라 처리된다. IEC 60601-1-4의 일부 요구사항은 IEC 62304가 처리하지 않는다.

IEC 60601-1-4:1996 및 개정 1:1999의 PEMS 요구사항	IEC 62304의 관련 요구사항
6.8 첨부 문서	
6.8.210	4.2와 4.3 c)
52.201 문서화	
52.201.1	4.1
52.201.2	4.1와 4.2
52.201.3	4.2
52.202 위험관리 계획	
52.202.1	4.2
52.202.2	5.1.1, 5.1.5
52.202.3	4.1, 5.1.1 g
52.203 개발 수명주기	
52.203.1	5.1.1
52.203.2	5.1.1
52.203.3	
52.203.4	5.1.7
52.203.5	7
52.204 위험관리 프로세스	
52.204.1	4.2

52.204.2	4.2, 7
52.204.3	
52.204.3.1	
52.204.3.1.1	4.2, 7.1
52.204.3.1.2	4.2, 7.1.2
52.204.3.1.3	4.2
52.204.3.1.4	4.2, 7.1.2 e)
52.204.3.1.5	4.2, 7.1.2
52.204.3.1.6	4.2, 7.1
52.204.3.1.7	4.2
52.204.3.1.8	4.2
52.204.3.1.9	4.2
52.204.3.1.10	4.2
52.204.3.2	
52.204.3.2.1	4.2
52.204.3.2.2	4.2, 4.3
52.204.3.2.3	
52.204.3.2.4	
52.204.3.2.5	4.2
52.204.4	
52.204.4.1	4.2
52.204.4.2	4.2
52.204.4.3	4.2
52.204.4.4	4.2
52.204.4.5	
52.204.4.6	4.2
52.205 담당자의 자격검증	4.1
52.206 요구사항 시방서	
52.206.1	5.2
52.206.2	7.2.2
52.206.3	
52.207 아키텍처	
52.207.1	5.3.1
52.207.2	5.3
52.207.3	
52.207.4	
52.207.5	
52.208 설계 및 구현	
52.208.1	5
52.208.2	
52.209 검증	
52.209.1	5.7.1
52.209.2	5.1.5, 5.1.6
52.209.3	5.2.6, 5.3.6, 5.4.4, 5.5.5, 5.6, 5.7
52.209.4	

52.210 밸리데이션	
52.210.1	4.1
52.210.2	4.1
52.210.3	4.1
52.210.4	
52.210.5	
52.210.6	
52.210.7	
52.211 수정	
52.211.1	6
52.211.2	4.1, 6
52.212 평가	
52.212.1	4.1

[표 C.4] IEC 60601-4와의 관계

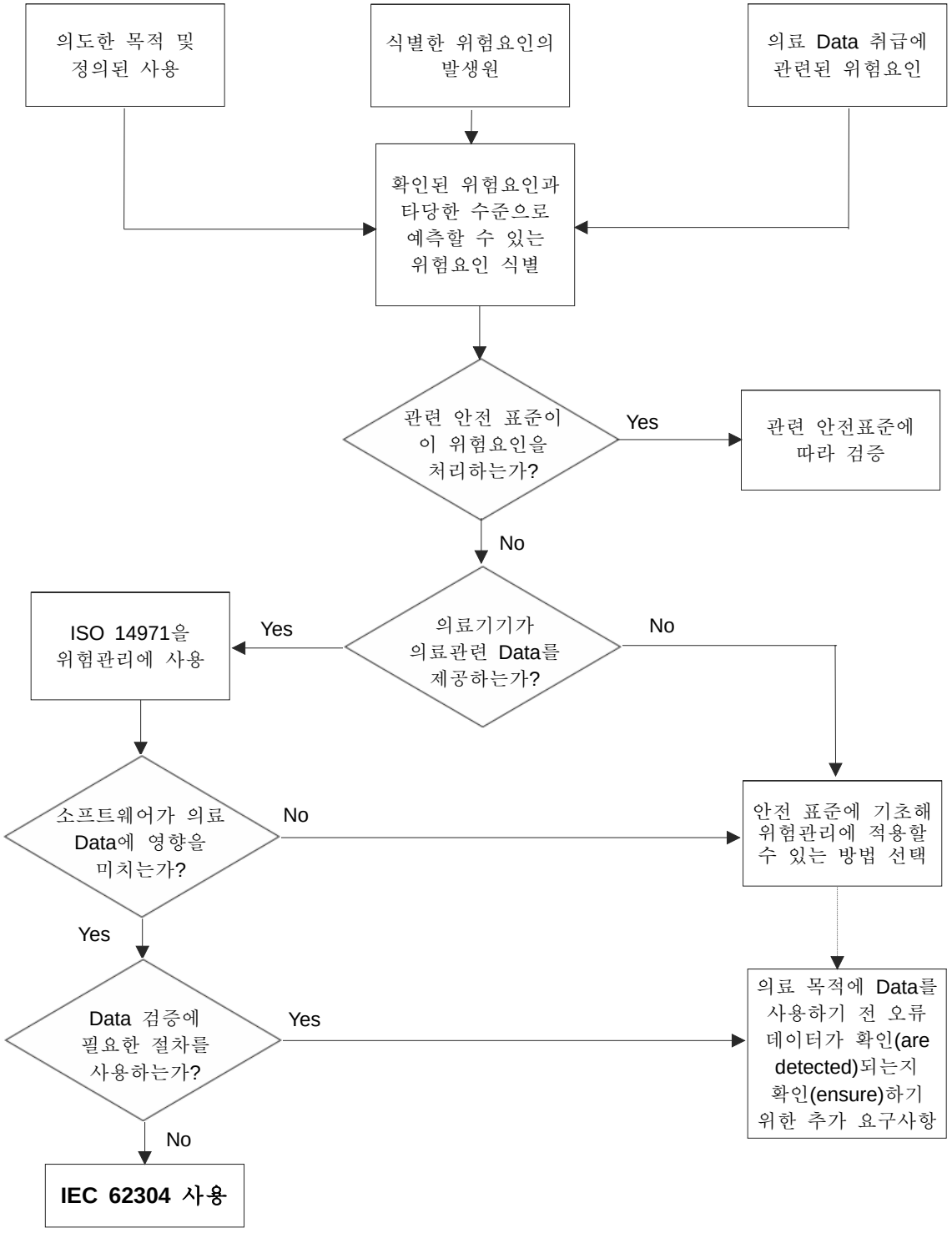
C0.5 IEC 61010-1에 대한 관계

IEC 61010-1[4]의 범위에는 전기 시험과 측정장치, 전기관리 장치와 전기 실험실 장치가 포함된다. 의료 환경이나 체외 진단장치(IVD)에는 실험실 장치만 사용된다.

법규 또는 규제적 참고문헌 때문에 IVD 장치는 의료기기로 할당되지만, IEC 60601-1[1]의 범위 안에 든다. 그 이유는 엄격히 말해 IVD 계측기는 환자와 직접 접촉하는 의료기기가 아니라는 사실과 그러한 제품은 다양한 실험실에서 다양한 용도를 위해 제조된다는 사실에서 비롯된다. 따라서 IVD 계측기나 IVD 계측기의 부속품으로서 사용하는 경우는 드물다.

실험실 장치를 IVD 장치로 사용할 경우 특정 결과는 의료 기준에 따라 평가해야 한다. 위험관리에는 ISO 14971의 적용이 필요하다. 또한 그러한 제품에 예를 들어 소프트웨어 결함에 의한 의료 데이터(측정결과)의 불필요한 변경과 같은 위험요인을 발생시킬 수 있는 소프트웨어가 포함된 경우 IEC 62304를 고려해야 한다.

그림 C.3의 흐름도를 보면 위험관리 프로세스와 IEC 62304의 적용에 관한 일차적인 방법을 쉽게 이해할 수 있다.



[그림 C.3] IEC 62304를 IEC 61010-1에 적용

C0.6 ISO/IEC 12207와의 관계

이 표준은 일반적인, 즉 의료기기에 제한되지 않는 소프트웨어 수명주기 프로세스에 대한 요구사항을 정의하는 ISO/IEC 12207[9]의 접근방식과 개념에서 도출된 것이다.

이 표준은 주로 다음과 관련해 ISO/IEC 12207과는 다르다. 이 표준의 성격은 다음과 같다.

- 시스템 요구사항, 시스템 아키텍처 및 밸리데이션과 같은 시스템 관련 측면을 배제한다.
- 의료기기에 대해 다른 표준에 문서화되고 중복되는 활동이라고 간주되는 프로세스 일부를 생략한다.
- (안전성) 위험관리 프로세스와 소프트웨어 릴리즈 프로세스를 추가한다.
- 프로세스를 지원하는 문서와 검증을 개발과 유지보수 프로세스에 포함시킨다.
- 각 프로세스의 프로세스 구현과 기획을 개발과 유지보수 프로세스의 단일 활동으로 통합한다.
- 안전성 필요와 관련한 요구사항을 분류한다.
- ISO/IEC 12207과는 달리 프로세스를 일차 또는 지원 프로세스로 명백히 분류하지 않으며, 프로세스를 그룹으로 구성하지 않는다.

이러한 변경의 대부분은 표준을 의료기기 분야의 요구에 따라 다음과 같은 방식으로 조정하고자 하는 필요성에 의해 발생한다.

- 안전성 측면 및 의료기기 위험관리 표준 ISO 14971에 집중한다.
- 규제된 환경에 유용한 적절한 프로세스를 선택한다.
- 소프트웨어 개발이 품질시스템에 포함되어 있다는 사실을 고려한다(ISO/IEC 12207의 프로세스와 요구사항 일부 취급).
- 사용 용이성의 촉진을 위해 추상적인 수준을 내린다.

이 표준은 ISO/IEC 12207과 상반되는 것이 아니다. ISO/IEC 12207은 이 표준의 요구사항이 포함되는, 잘 구성된 소프트웨어 개발 수명주기 모델의 설정에 유용하다.

표 C.5에 IEC 62304와 ISO/IEC 12207과의 관계를 표시한다.

ISO/IEC 62304 프로세스		ISO/IEC 12207 프로세스	
활동	업무	활동	업무
5 소프트웨어 개발 프로세스		5.3 개발 프로세스 6.1 문서화 프로세스 6.2 형상관리 프로세스 6.4 검증 프로세스 6.5 밸리데이션 프로세스 6.8 문제해결 프로세스 7.1 경영 프로세스	

5.1 소프트웨어 개발 기획		5.3.1 프로세스 구현 5.3.3 소프트웨어 구축 설계 5.3.7 소프트웨어 코딩 및 시험 5.3.8 소프트웨어 통합 5.3.9 소프트웨어 자격검증 시험 5.3.10 시스템 통합 6.1.1 프로세스 구현 6.2.1 프로세스 구현 6.2.2 형상 식별 6.4.1 프로세스 구현 6.5.1 프로세스 구현 6.8.1 프로세스 구현 7.1.2 기획 7.1.3 실행 및 관리 7.2.2 기반구조의 확립 7.2.3 기반구조의 유지보수	
	5.1.1 소프트웨어 개발 계획	5.3.1 프로세스 구현 7.1.2 기획	5.3.1.1 5.3.1.3 5.3.1.4 7.1.2.1
	5.1.2 소프트웨어 개발 계획 갱신 유지	7.1.3 실행 및 관리	7.1.3.3
	5.1.3 시스템 설계와 개발에 대한 소프트웨어 개발 계획 참조	5.3.3 시스템 구축 설계 5.3.10 시스템 통합 6.5.1 프로세스 구현	5.3.3.1 5.3.10.1 6.5.1.4
	5.1.4 소프트웨어 개발 표준, 방법 및 도구 기획	5.3.1 프로세스 구현	5.3.1.3 5.3.1.4
	5.1.5 소프트웨어 통합 및 통합 시험 기획	5.3.8 소프트웨어 통합	5.3.8.1
	5.1.6 소프트웨어 검증 기획	6.4.1 프로세스 구현 5.3.7 소프트웨어 코딩 및 시험 5.3.8 소프트웨어 통합 5.3.9 소프트웨어 자격검증 시험	6.4.1.4 6.4.1.5 5.3.7.5 5.3.8.5 5.3.9.3
	5.1.7 소프트웨어 위험관리 기획	개정 1:2002 – F 3.15 위험관리 프로세스	
	5.1.8 문서화 기획	6.1.1 프로세스 구현	6.1.1.1
	5.1.9 소프트웨어 형상관리 기획	6.2.1 프로세스 구현 6.8.1 프로세스 구현	6.2.1.1 6.8.1.1
	5.1.10 관리할 지원 항목	7.2.2 기반구조의 확립 7.2.3 기반구조의 유지보수	7.2.2.1 7.2.3.1
	5.1.11 검증 전 소프트웨어	6.2.2 형상 식별	6.2.2.1

	형상 항목(item) 식별		
5.2 소프트웨어 요구사항 분석		5.3.3 시스템 구축 설계 5.3.4 소프트웨어 요구사항 분석 6.4.2 검증	
	5.2.1 시스템 요구사항에서 소프트웨어 요구사항을 정의하고 문서화	5.3.3 시스템 구축 설계	5.3.3.1
	5.2.2 소프트웨어 요구사항 내용	5.3.4 소프트웨어 요구사항 분석	5.3.4.1
	5.2.3 소프트웨어 요구사항에 위험관리 방법 포함		
	5.2.4 의료기기 위험분석 재평가		없음
	5.2. 시스템 요구사항 갱신	5.3.4 소프트웨어 요구사항 분석	a) b)
	5.2.6 소프트웨어 요구사항 검증	5.3.4 소프트웨어 요구사항 분석 6.4.2 검증	5.3.4.2 6.4.2.3
5.3 소프트웨어 구축 설계		5.3.5 소프트웨어 구축 설계	
	5.3.1 소프트웨어 요구사항을 아키텍처로 변형	5.3.5 소프트웨어 구축 설계	5.3.5.1
	5.3.2 소프트웨어 항목의 연계성을 위한 아키텍처 개발		5.3.5.2
	5.3.3 SOUP 항목의 기능 및 성능 요구사항 명시		없음
	5.3.4 SOUP 항목에 필요한 시스템 하드웨어와 소프트웨어 명시		없음
	5.3.5 위험관리에 필요한 분리 식별		없음
	5.3.6 소프트웨어 아키텍처 검증	5.3.5 소프트웨어 구축 설계	5.3.5.6
5.4 소프트웨어 상세 설계		5.3.6 소프트웨어 상세 설계 6.4.2 검증	
	5.4.1 소프트웨어 아키텍처를 소프트웨어 유닛으로 개량	5.3.6 소프트웨어 상세 설계	5.3.6.1
	5.4.2 각 소프트웨어 유닛에		

	대해 상세 설계 개발		
	5.4.3 연계성에 대한 상세 설계 개발		5.3.6.2
	5.4.4 상세 설계 검증	6.4.2 검증	5.3.6.7
5.5 소프트웨어 유닛 구현 및 확인		5.3.6 소프트웨어 상세 설계 5.3.7 소프트웨어 코딩 및 시험 6.4.2 검증	
	5.5.1 각 소프트웨어 유닛 구현	5.3.7 소프트웨어 코딩 및 시험	5.3.7.1
	5.5.2 소프트웨어 유닛 검증 프로세스 확립	5.3.6 소프트웨어 상세 설계 5.3.7 소프트웨어 코딩 및 시험	5.3.6.5 5.3.7.5
	5.5.3 소프트웨어 유닛 인수 기준	5.3.7 소프트웨어 코딩 및 시험	5.3.7.5
	5.5.4 소프트웨어 유닛 추가 인수 기준	5.3.7 소프트웨어 코딩 및 시험 6.4.2 검증	5.3.7.5 6.4.2.5
	5.5.5 소프트웨어 유닛 검증	5.3.7 소프트웨어 코딩 및 시험	5.3.7.2
5.6 소프트웨어 통합 및 통합 시험		5.3.8 소프트웨어 통합 5.3.9 소프트웨어 자격검증 시험 5.3.10 시스템 통합 6.4.1 프로세스 구현 6.4.2 검증	
	5.6.1 소프트웨어 유닛 통합	5.3.8 소프트웨어 통합	5.3.8.2
	5.6.2 소프트웨어 통합 검증	5.3.8 소프트웨어 통합 5.3.10 시스템 통합	5.3.8.2 5.3.10.1
	5.6.3 통합 소프트웨어 시험	5.3.9 소프트웨어 자격검증 시험	5.3.9.1
	5.6.4 통합 시험 내용		5.3.9.3.
	5.6.5 통합 시험 절차 검증	6.4.2 검증	6.4.2.2
	5.6.6 회귀시험 수행	5.3.8 소프트웨어 통합	5.3.8.2
	5.6.7 통합시험 기록 내용	5.3.8 소프트웨어 통합	5.3.8.2
	5.6.8 소프트웨어 문제해결 프로세스의 사용	6.4.1 프로세스 구현	6.4.1.6
5.7 소프트웨어 시스템 시험		5.3.8 소프트웨어 통합 5.3.9 소프트웨어 자격검증 시험 6.4.1 프로세스 구현 6.4.2 검증	

		6.8.1 프로세스 구현	
	5.7.1 각 소프트웨어 요구사항에 대한 시험 확립	5.3.8 소프트웨어 통합 5.3.9 소프트웨어 자격검증 시험	5.3.8.4 5.3.9.1
	5.7.2 소프트웨어 문제해결 프로세스의 사용	6.4.1 프로세스 구현	6.4.1.6
	5.7.3 변경 후 재시험	6.8.1 프로세스 구현	6.8.1.1
	5.7.4 소프트웨어 시스템 시험 검증	6.4.2 검증 5.3.9 소프트웨어 자격검증 시험	6.4.2.2 5.3.9.3
	5.7.5 각 시험 소프트웨어 시스템 시험 기록 내용에 대한 데이터 문서화	5.3.9 소프트웨어 자격검증 시험	5.3.9.1
5.8 소프트웨어 릴리즈		5.3.9 소프트웨어 자격검증 시험 5.4.2 작동 시험 6.2.5 형상 평가 6.2.6 릴리즈 관리 및 인도	
	5.8.1 소프트웨어 검증의 완벽성 확인(ensure)	5.4.2 작동 시험 6.2.6 릴리즈 관리 및 인도	5.4.2.1 5.4.2.2 6.2.6.1
	5.8.2 확인(known)된 잔류 비정상 상태 문서화	6.2.5 형상 평가 5.3.9 소프트웨어 자격검증 시험	6.2.5.1 5.3.9.3
	5.8.3 확인(known)된 잔류 비정상 상태 평가		
	5.8.4 릴리즈 버전의 문서화	6.2.6 릴리즈 관리 및 인도	6.2.6.1
	5.8.5 릴리즈한 소프트웨어를 작성한 방법 문서화		
	5.8.6 활동과 업무의 완결 확인(ensure)		
	5.8.7 소프트웨어 저장		
	5.8.8 소프트웨어 릴리즈의 반복 정밀도 보장		
6 소프트웨어 유지보수 프로세스		5.5 유지보수 프로세스 6.2 형상관리 프로세스	
6.1 소프트웨어 유지보수 계획 확립		5.5.1 프로세스 구현	5.5.1.1
6.2 문제 및 수정		5.5.1 프로세스 구현	

분석		5.5.2 문제 및 코드화 분석 5.5.3 수정 구현 5.5.5 감소(마이그레이션)	
	6.2.1 피드백 기록 및 평가		
	6.2.1.1 피드백 모니터링	5.5.1 프로세스 구현	5.5.1.1 5.5.1.2
	6.2.1.2 피드백 문서화 및 평가		
	6.2.1.3 안전에 관한 문제 보고서 영향의 평가	5.5.2 문제 및 수정 분석	5.5.2.1 5.5.2.2 5.5.2.3 5.5.2.4
	6.2.2 소프트웨어 문제해결 프로세스의 사용	5.5.1 프로세스 구현	5.5.1.2
	6.2.3 변경 요청 분석	5.5.2 문제 및 수정 분석	5.5.2.1
	6.2.4 변경 요청 승인	5.5.2 문제 및 수정 분석	5.5.2.5
	6.2.5 사용자 및 규제자와의 의사소통	5.5.3 수정 구현 5.5.5 마이그레이션	5.5.3.1 5.5.5.3
6.3 수정 구현		5.5.3 수정 구현 6.2.6 릴리즈 관리 및 인도	
	6.3.1 확립된 프로세스를 사용해 수정 구현	5.5.3 수정 구현	5.5.3.2
	6.3.2 수정한 소프트웨어 시스템의 재 릴리즈	6.2.6 릴리즈 관리 및 인도	6.2.6.1
7 소프트웨어 위험관리 프로세스		개정 1:2002 – F 3. 15 위험관리 프로세스 62304의 프로세스는 개정 1에서 취급되지 않은 위험/위험요인 현안을 취급한다. 일부 공통점이 있지만(위험 측정 등) 분석의 초점은 전혀 다르다.	
8 소프트웨어 형상관리 프로세스		5.5 유지보수 프로세스 6.2 형상관리 프로세스	
8.1 형상 식별		6.2.2 형상 식별	
	8.1.1 형상 항목 식별을 위한 방법 확립	6.2.2 형상 식별	6.2.2.1
	8.1.2 SOUP 식별		없음
	8.1.3 시스템 형상 문서 식별	6.2.2 형상 식별	6.2.2.1
8.2 변경관리		5.5.3 수정 구현 6.2.3 형상관리	

	8.2.1 변경 요청 승인	6.2.3 형상관리	6.2.3.1
	8.2.2 변경 구현	5.5.3 수정 구현 6.2.3 형상관리	5.5.3.2 6.2.3.1
	8.2.3 변경 검증	6.2.3 형상관리	6.2.3.1
	8.2.4 변경 추적성에 대한 방법 제공		
8.3 형상 상태 설명		6.2.4 형상 상태 설명	6.2.4.1
9 소프트웨어 문제해결 프로세스		5.5 유지보수 프로세스 6.2 형상관리 6.8 문제해결 프로세스	
9.1 문제 보고서 작성		6.8.1 프로세스 구현 6.8.2 문제해결	6.8.1.1 b) 6.8.2.1
9.2 문제 조사		6.8.2 문제해결 6.8.1 프로세스 구현	6.8.2.1 6.8.1.1 b)
9.3 관련 당사자에게 통보		6.8.1 프로세스 구현	6.8.1.1 a)
9.4 변경관리 프로세스 사용		6.2.3 형상관리 5.5.3 수정 구현	
9.5 기록 유지		6.8.1 프로세스 구현	6.8.1.1 a)
9.6 문제 경향 분석		6.8.1 프로세스 구현 6.8.2 문제해결	6.8.1.1 b) 6.8.2.1
9.7 소프트웨어 문제해결 검증		6.8.1 프로세스 구현	6.8.1.1 d)
9.8 시험 문서의 내용			12207의 모든 시험 업무 문서화

C.7 IEC 61508과의 관계

안전성이 중요한 소프트웨어의 설계와 관련된 이 표준이 IEC 61508의 원칙을 준수해야 하는지에 관한 의문이 제기되었다. 아래에는 이 표준의 범위가 설명된다.

IEC 61508은 세 가지 주요 현안을 취급한다.

1) 위험관리 수명주기와 수명주기 프로세스

2) 안전 무결성 수준의 정의
3) 소프트웨어 개발을 위한 기술, 도구 및 방법의 권장사항 및 다양한 업무 수행 담당자의 독립성 수준

현안 1)은 이 표준에서 ISO 14971(위험관리에 대한 의료기기 부문 표준)에 대한 규정 참조에 의해 취급된다. 이 참조의 영향은 위험관리에 대한 ISO 14971의 접근방식을 의료기기 소프트웨어에 대한 소프트웨어 프로세스 일부로 채택한다는 것이다.

현안 2)의 경우 이 표준은 IEC 61508보다 단순한 접근방식을 선택한다. IEC 61508은 신뢰성 목표의 관점에서 정의한 네 개 “소프트웨어 무결성 수준”으로 소프트웨어를 분류한다. 신뢰성 목표는 소프트웨어 결함에 의해 발생하는 위해의 심각도와 가능성을 모두 정량화하는 위험 분석 후 식별된다.

이 표준은 분류 전 소프트웨어 결함의 발생가능성의 고려를 허용하지 않음으로써 현안 2)을 단순화한다. 세 개 소프트웨어 안전성 등급으로 분류하는 것은 결함에 의해 발생되는 위해의 심각도에만 기초해 결정된다. 분류 후 다양한 소프트웨어 안전성 등급에 다양한 프로세스가 필요하다. 그 의도는 소프트웨어 결함 발생가능성을 더욱 축소하는 것이다.

현안 3)은 이 표준에서는 처리되지 않는다. 이 표준의 사용자는 IEC 61508을 양호한 소프트웨어 방법, 기술 및 도구를 제공하는 표준으로 사용하는 것이 바람직하다. 단, 현재 사용되고 미래에 개발될 다른 접근방식도 동일하게 양호한 결과를 제공할 수 있다. 이 표준은 소프트웨어 활동(예를 들어 확인) 하나를 담당하는 담당자와 다른 활동(예를 들어 검증) 담당자 사이의 독립성에 관련해 어떤 권고도 하지 않는다. 특히, 이 표준은 독립적인 안전성 평가자에 대해 어떤 요구사항도 제시하지 않는다. 이는 ISO 14971의 범위이기 때문이다.

부록 D (참조용) 구현

D.1 서문

이 부록에는 이 표준을 제조업체 프로세스에 구현할 수 있는 방법이 개괄적으로 설명된다. 또한 이 부록에서는 ISO 13485[7]과 같은 다른 표준이 적절하고 유사한 프로세스를 필요로 한다고 간주한다.

D.2 품질경영시스템

표준 관점에서 의료기기 소프트웨어가 포함되는 의료기기 제조업체의 경우 품질경영시스템(QMS)이 4.1에서 요구된다. 이 표준에 의하면 QMS를 꼭 인증할 필요는 없다.

D.3 품질경영 프로세스 평가

확립되고 문서화된 QMS 프로세스가 얼마나 잘 소프트웨어 수명주기의 프로세스를 담당하는지 제조업체의 책임으로 심사, 검사 또는 분석을 통해 평가할 것이 권장된다. 식별한 간극은 QM 프로세스를 확대해 처리할 수도 있고 별도로 설명할 수도 있다. 소프트웨어의 개발, 검증 및 밸리데이션을 규제하는 프로세스 설명을 제조업체가 이미 파악한 경우 이러한 설명도 평가해 이 표준에 얼마나 잘 부합하는지 판단해야 한다.

D.4 이 표준의 요구사항을 제조업체의 품질경영 프로세스에 통합

이 표준은 이미 QMS 시스템에 확립되어 있거나 새 프로세스에 통합된 프로세스를 채택하거나 확장하면 구현할 수 있다. 이 표준은 그러한 구현 방법은 명시하지 않는다. 제조업체가 적절한 방식으로 구현할 수 있다.

자체에 문서화된 QMS가 없는 자기제품 생산자(OEM)나 하청업체가 의료기기 소프트웨어를 개발할 때 제조업체는 이 표준에 설명한 프로세스가 적절히 적용되는지 확인(ensuring)할 책임이 있다.

D.5 인증된 QMS가 없는 소규모 제조업체를 위한 점검목록

제조업체는 소프트웨어의 가장 높은 안전성 분류(A, B 또는 C)를 결정해야 한다. 표 D.1에는 이 표준에 설명한 활동이 모두 수록되어 있다. ISO 13485를 참조하면 QMS에 해당되는 수준을 쉽게 정의할 수 있을 것이다. 제조업체는 요구되는 소프트웨어 안전성 등급에 기초해 요구되는 각 프로세스를 기존 프로세스와 비교해 평가해야 한다. 요구사항이 이미 취급된 경우 관련 프로세스 설명을 참고해야 한다.차이가 있을 경우 프로

세스를 개선하기 위한 조치가 필요하다.

이 목록은 조치를 수행한 후 프로세스의 평가에도 사용할 수 있다.

활 동	ISO 13485:2003 관련 조항	기존 절차에서 취급여부?	'예'인 경우: 참조	수행할 조치
5.1 소프트웨어 개발 기획	7.3.1 설계 및 개발 기획	Yes/No		
5.2 소프트웨어 요구사항 분석	7.3.2 설계 및 개발 입력	Yes/No		
5.3 소프트웨어 구축 설계		Yes/No		
5.4 소프트웨어 상세 설계		Yes/No		
5.5 소프트웨어 유닛 구현 및 검증		Yes/No		
5.6 소프트웨어 통합 및 통합 시험		Yes/No		
5.7 소프트웨어 시스템 시험	7.3.3 설계 및 개발 출력 7.3.4 설계 및 개발 검토	Yes/No		
5.8 소프트웨어 릴리즈	7.3.5 설계 및 개발 검증 7.3.6 설계 및 개발 밸리데이션	Yes/No		
6.1 소프트웨어 유지보수 계획 확립	7.3.7 설계 및 개발 변경 관리	Yes/No		
6.2 문제 및 수정 분석		Yes/No		
6.3 수정 구현	7.3.5 설계 및 개발 검증 7.3.6 설계 및 개발 밸리데이션	Yes/No		
7.1 위험한 상황에 대한 소프트웨어 영향의 분석		Yes/No		
7.2 위험관리 방법		Yes/No		
7.3 위험관리 방법의 검증		Yes/No		
7.4 소프트웨어 변경의 위험관리		Yes/No		
8.1 형상 식별	7.5.3 식별 및 추적성	Yes/No		
8.2 변경관리	7.5.3 식별 및 추적성	Yes/No		
8.3 형상 상태 설명		Yes/No		
9 소프트웨어 문제해결 프로세스		Yes/No		

[표 D.1] 인증된 QMS가 없는 소규모 제조업체를 위한 점검목록

참고문헌

[1] IEC 60601-1, Ed. 3: Medical electrical equipment - Part 1: General requirements for basic safety and essential performance

[2] IEC 60601-1-4:1996, Medical electrical equipment - Part 1-4: General requirements for safety - Collateral standard: Programmable electrical medical systems Amendment 1 (1999)

[3] IEC 61508:1998, Functional safety of electrical/electronic/programmable electronic safety related systems

[4] IEC 61010-1:2001, Safety requirements for electrical equipment for measurement, control, and laboratory use - Part 1: General requirements

[5] ISO 9000:2000, Quality management systems - Fundamentals and vocabulary

[6] ISO 9001:2000, Quality management systems - Requirements

[7] ISO 13485:2003, Medical devices - Quality management systems - Requirements for regulatory purposes

[8] ISO/IEC 9126-1:2001, Software engineering - Product quality - Part 1: Quality model

[9] ISO/IEC 12207:1995, Information technology - Software life cycle processes Amendment 1 (2002), Amendment 2 (2004)

[10] ISO/IEC 14764:1999, Information technology - Software maintenance

[11] ISO/IEC 90003:2004, Software and system engineering - Guidelines for the application of ISO 9001:2000 to computer software

[12] ISO/IEC Guide 51:1999, Safety aspects - Guidelines for their inclusion in standards

[13] IEEE 610.12:1990, IEEE standard glossary of software engineering terminology

[별첨 5]

소프트웨어 밸리데이션의 일반 원칙
(Guidance for Industry, GENERAL PRINCIPLES OF SOFTWARE VALIDATION)

발행일: 2002. 1. 11

Section 1. 목적

이 지침은 소프트웨어 의료기기 또는 의료기기의 설계·개발 및 제조에 사용되는 소프트웨어의 밸리데이션에 적용되는 FDA의 일반적 원칙을 제시한다. 이 최종 지침(Version 2.0)는 1997.7.9판 소프트웨어 밸리데이션의 일반원칙(Version 1.1)을 대체하는 것이다.

Section 2. 적용 범위

이 지침은 의료기기 품질관리규정(Quality System regulation)이 소프트웨어에 어떻게 적용이 되며 FDA(agency)에서 밸리데이션 시스템을 평가하는 접근법을 보여주고 있다. 예를 들면 이 문서는 소프트웨어 밸리데이션에 있어 FDA에서의 수용가능한 요소들 목록을 제시하였다. 그러나 법령을 준수함에 있어 적용되는 모든 활동과 작업을 제시한 것은 아니다.

이 지침의 적용 범위는 엄밀한 의미의 밸리데이션 적용 범위보다 넓다. 계획(planning), 검증, 시험, 추적성, 형상 관리(configuration management) 및 기타 이 지침 내에서 언급되는 우수 소프트웨어 공학(good software engineering)의 많은 측면은 소프트웨어가 유효하다는 최종 결론을 뒷받침할 수 있는 중요한 활동이다.

이 지침은 소프트웨어 life cycle 관리와 위험관리 활동의 통합을 권장한다. 개발되는 소프트웨어와 관련된 사용목적 및 안전 위험(safety risk)에 기초하여 소프트웨어 개발자는 구체적인 접근방법, 사용되는 기술의 조합, 적용되는 노력(effort)의 수준을 결정하여야 한다. 이 지침은 특정 life cycle 모델, 기술 또는 방법을 권장하지는 않지만, 소프트웨어 밸리데이션 및 검증 활동이 소프트웨어 전체 life cycle동안 행해져야 한다는 것은 권장하고 있다.

소프트웨어가 기기 제조업자 이외의 다른 개발자에 의하여 개발(예: OTS 소프트웨어)되는 경우, 소프트웨어 개발자는 FDA 규정에 적합해야 하는 직접적인 책임이 있는 것은 아니다.

이런 이유 때문에 법적 책임이 있는 자(예: 기기 제조업자)는 OTS 소프트웨어 개발자의 활동이 적절한지 평가할 필요가 있고 기기 제조업자의 사용목적을 위하여 소프트웨어가 유효한지를 보장하기 위하여 어떤 추가적인 노력이 필요한지 결정하여야 한다.

2.1. 적용

이 지침의 적용 범위는 다음과 같다.

- 의료기기의 구성 요소, 부분, 또는 부속품으로서 사용되는 소프트웨어
- 소프트웨어 자체가 의료기기인 것(예: 혈액기기(blood establishment) 소프트웨어)
- 의료기기 생산에 사용되는 소프트웨어(예: 제조 장비내의 programmable logic controller)
- 의료기기 제조업자의 품질시스템 운영에 사용되는 소프트웨어(예 : 기기 이력 기록 및 관리하는 소프트웨어)

이 문서는 일반적인 소프트웨어 밸리데이션 원칙에 기초하므로 어떠한 소프트웨어에도 적용될 수 있다. FDA의 목적상, 식품의약품화장품법의 201(h)조항 및 FDA 소프트웨어와 규제 정책에 정의된 바와 같이 이 지침은 규제대상이 되는 의료기기와 관련된 어떠한 소프트웨어에도 적용된다. 이 문서는 어떠한 소프트웨어가 규제대상인지 특별히 규정하고 있지는 않다.

2.2. 대상(AUDIENCE)

이 지침은 다음과 같은 대상에게 유용한 정보와 권고(recommendation)를 제공한다.

- 의료기기 품질시스템 규정에 적용을 받는 사람
- 의료기기 소프트웨어의 설계, 개발, 제작의 책임이 있는 사람
- 의료기기의 설계, 개발 또는 제조를 위하여 사용되는 자동화 장비의 설계, 개발, 제작 또는 획득(procurement)이나 품질시스템 이행에 사용되는 소프트웨어 툴(Tool)에 대한 책임이 있는 사람
- FDA 감시원(Investigators)
- FDA 적합성 심사원(Compliance office)
- FDA 기술문서 검토자(Scientific reviewer)

2.3. 가장 부담이 적은 접근방법

의료기기 규정의 모든 범위 내에서 가장 부담이 적은 접근법을 적용하여야 하는 것이 바람직하다. 이 지침은 관련된 과학적, 법적 요구사항에 대한 철저한(careful) 검토를 반영하고, 그 요구사항에 부합하기 위하여 제조업자가 준수하여야 할 가장 부담이 적은 방법을 제시한 것이다. 그러나 보다 효율적인 접근법이 있다면, 이를 고려해볼 수 있다.

이 지침의 서문에 적혀있는 사람이나 CDRH 옴부즈맨에게 서면으로 당신의 견해를 보낼 수 있다. CDRH의 옴부즈맨과의 연락방법 등은 다음의 인터넷 사이트에서 알 수 있다.
http://www.fda.gov/cdrh/resolvingdisputes/ombudsman.html

2.4. 소프트웨어 밸리데이션을 위한 법적 요구사항

1992~1998년 동안 실시된 3,140건의 의료기기 리콜(recall) 중 242건(7.7%)은 소프트웨어 불량이 원인이라고 FDA 분석결과 나타났다. 소프트웨어와 관련된 리콜 중 192건(79%)은 소프트웨어가 최초 생산 및 시판 후 변경되었을 때 발생된 소프트웨어 결함에 의한 것이었다. 이 지침에서 다루는 소프트웨어 밸리데이션 및 기타 관련된 우수 소프트웨어공학 기준은 이러한 결함 및 리콜을 피할 수 있는 원칙적인 방법을 제시한 것이다.

소프트웨어 밸리데이션은 품질시스템 규정(1996.9.7 공포, 1997.6.1 시행)의 요구사항이다. 밸리데이션은 의료기기 구성 요소로 사용되는 소프트웨어, 그 자체가 의료기기인 소프트웨어, 의료기기 생산에 사용되거나 제조업자의 품질시스템 운영에 사용되는 소프트웨어에 적용되는 요구사항이다.

특별히 등급분류규정에서 제외된 것이 아니라면, 1997.6.1 이후 개발된 의료기기 소프트웨어 제품은 기기등급에 관계없이 설계관리(21 CFR 820.30 참조)를 적용할 수 있다. 이 요구사항은 현재 개발 중인 프로젝트, 신규 개발 프로젝트 및 기존의 의료기기 소프트웨어의 모든 변경사항에 적용된다. 의료기기 소프트웨어의 밸리데이션을 위한 특정 요구사항은 21 CFR 820.30(g) 조항을 참조바란다. 또한 의료기기 소프트웨어에 대해서 계획, 입력, 검증, 검토와 같은 설계 관리가 요구되어진다.(21 CFR 820.30 참조) 이러한 활동에 따른 문서화 결과는 의료기기 소프트웨어가 유효한지 결정하는데 도움이 될 것이다.

품질시스템의 구성요소이거나 의료기기 생산 공정의 자동화에 사용되는 소프트웨어는 21 CFR 820.70(i)에서 요구되는 사용목적에 대하여 밸리데이션을 하여야 한다. 이는 기기 설계, 시험, 구성요소 수락(acceptance), 제조, 라벨링, 포장, 배포, 불만 처리를 자동화하거나 품질시스템의 다른 측면을 자동화하는 모든 소프트웨어에 적용되는 요구사항이다.

또한 전자기록의 작성, 변경, 유지 및 전자서명 관리에 사용되는 컴퓨터 시스템도 밸리데이션이 요구된다(21 CFR 11.10(a) 참조). 이런 컴퓨터 시스템은 정확성, 신뢰성, 의도된 성능과의 일치성 및 효력이 상실되고 변경된 기록의 식별능력을 확실히 하기 위하여 유효성이 확인되어야 한다.

상기와 같은 적용을 위한 소프트웨어는 조직 내 또는 계약에 의해 개발된다. 그러나 소프트웨어는 간혹 특별한 사용목적을 위하여 기성품(off-the-shelf)으로 구입하기도 한다. OTS로 구매된 소프트웨어일지라도 본래의 사용 목적에 맞게 유효하다는 것을 입증하기 위하여 모든 생산 및/또는 품질시스템 소프트웨어는 본래 의도된 사용 목적, 비교가능한 시험결과 및 증거에 대한 정보를 규정한 문서화된 요구사항을 수립하여야 한다.

자동화 의료기기, 자동화 생산 및 품질시스템에서 OTS 소프트웨어의 사용이 증가하고 있다. OTS 소프트웨어가 다양한 기능을 가지고 있더라도 의료기기 제조업자는 그 중 일부 기능만 필요로 할 수 있다. 의료기기 제조업자는 해당 의료기기 또는 그 의료기기의 생산에 사용되는 소프트웨어가 적절한지에 대한 책임이 있다. 의료기기 제조업자는 OTS 소프트웨어의 구매시 해당 소프트웨어가 의도된 목적대로 작동할지를 확실히 하여야 한다. 이 문서 Section 6.3에 제조 또는 품질시스템에 사용되는 OTS 소프트웨어에 대한 추가 지침이 있다. 의료기기 소프트웨어에 유용한 추가 정보는 FDA의 'Guidance for Industry, FDA Reviewers, and Compliance on Off-The-Shelf Software Use in Medical Devices'를 참조하기 바란다.

2.5. 품질시스템 규정과 시판전 제출자료

이 문서는 소프트웨어 밸리데이션의 이행을 포함한 품질시스템규정(QSR) 관련 사항을 언급하고 있다. 소프트웨어 밸리데이션 과정의 관리를 위한 지침을 제시한다. 소프트웨어 밸리데이션 과정의 관리와 자동화 공정에서의 공정 밸리데이션과 같은 다른 밸리데이션 요구사항을 혼동해서는 아니 된다.

의료기기 제조업자는 FDA에 제출하는 시판전 자료처럼 품질시스템 및 설계관리 요구사항에 적합함을 증명하는 동일한 절차 및 기록을 사용할 수 있다. 이 문서는 소프트웨어 밸리데이션과 관련하여 특정한 안전성 또는 유효성 사항을 언급하지 않으며, 규제되는 소프트웨어의 시판전 제출 자료를 위한 설계사항 및 문서화 요구사항에 대하여 다루고 있지 않다. 시판전 제출 자료에서 요구되는 안전성, 유효성 및 문서화 요구사항과 관련된 특정 사항은 Office of Device Evaluation (ODE), Center for Devices and Radiological Health(CDRH) 또는 Office of Blood Research and Review, Center for Biologics Evaluation and Research(CBER)에서 다루어진다. 시판전 제출 자료를 위한 FDA 지침은 Appendix A를 참조하기 바란다.

Section 3. 소프트웨어 밸리데이션의 개관

많은 사람들이 소프트웨어 밸리데이션과 관련하여 품질시스템 규정에 적합함을 증명하기 위하여 하여야 할 특별한 지침을 문의한다. 이 문서에 제시된 소프트웨어 밸리데이션 관련 정보는 새로운 것이 아니다. Section 4, 5에 제시된 원칙과 작업(tasks)을 이용한 소프트웨어 밸리데이션은 소프트웨어 산업계에서는 20여년 동안 실시되어 온 것이다.

의료기기의 종류, 공정 및 제조시설의 다양성 때문에 적용 가능한 모든 밸리데이션 요소들을 이 문서 하나에서 다룰 수는 없다. 그러나 포괄적 다수 개념의 일반적인 적용이 밸리데이션을 위한 지침에서는 유용하게 적용될 수 있다. 이러한 포괄적 개념은 밸리데이션을 위한 전반적 접근법을 설계하는 체계를 제공할 것이다. 추가적인 특정정보는 부속서 A에 수록된 참고문헌에서 참고할 수 있다.

3.1. 용어의 정의

품질시스템 규정 또는 기타 하위 규정에서 정의되지 않은 이 지침에서 사용된 모든 다른 용어들은 “FDA Glossary of Computerized System and Software Development Terminology” 최신판에 정의되어 있다.

의료기기 품질시스템규정(21 CFR 820.3(k))에서 “establish”는 “정의하다(define), 문서화하다(document), 수행하다(implement)”로 정의하고 있다. 이 지침에서 “establish” 및 “established”는 위와 같은 의미로 해석되어야 한다.

의료기기 품질시스템규정에서 일부 정의는 소프트웨어 산업계에서 사용되는 일반적 용어와 비교했을 때 혼동될 수 있다. 일례로 요구사항(requirements), 규격(specification), 검증(verification) 및 밸리데이션(validation)을 들 수 있다.

3.1.1 요구사항과 규격

품질시스템 규정은 설계 입력 요구사항을 반드시 문서화하고 특정 요구사항은 반드시 검증하도록 하고 있으나 '요구사항'과 '규격'의 차이점을 명확히 설명하고 있지 않다. **'요구사항(requirement)'**은 시스템 또는 소프트웨어에 필요하거나 예상될 수 있는 것을 말한다. 요구사항은 명시되거나 함축된 고객의 요구를 반영하며 조직의 내부, 시장의 요구(market-based), 계약상 또는 법적 요구사항일 수도 있다. 다양한 종류의 요구사항이 있을 수 있다(예: 설계, 기능(functional), 실행, 인터페이스, 성능 또는 물리적 요구사항). 소프트웨어 요구사항은 전형적으로 소프트웨어에 배정된 시스템의 기능적인 측면에 대한 요구사항으로부터 기인한다. 소프트웨어 요구사항은 일반적으로 개발과정에서 기능적인 용어로 규정, 정의, 수정 및 갱신된다. 소프트웨어 요구사항을 정확하고 완벽하게 문서화하는데 성공하는 것은 결과물인 소프트웨어의 성공적인 밸리데이션을 위하여 결정적 요소이다.

규격(specification)'은 "요구사항을 설명한 문서"로 정의되어 있다(21 CFR 820.3(y)). 이는 도면(drawing), 도안(pattern) 또는 그 밖의 관련 문서를 참조하거나 포함할 수 있고 요구사항에 적합함을 확인할 수 있는 방법 및 기준을 가리킨다. 이 규격은 시스템 요구사항 규격, 소프트웨어 설계 규격, 소프트웨어 시험 규격서, 소프트웨어 통합 규격 등과 같이 여러 종류가 있다. 이러한 문서들은 "특정 요구사항"을 설정하고 있으며 검증이 필요한 다양한 형태의 설계 출력이다.

3.1.2 검증과 밸리데이션

품질시스템 규정은(QSR) ISO 8402:1994와의 조화를 이루어 "검증"과 "밸리데이션"을 분리된 별개의 용어로서 사용한다. 반면, 많은 소프트웨어 공학 저널의 논설 및 교과서는 "검증"과 "밸리데이션"을 번갈아 사용하거나 어떤 경우에는 소프트웨어 "검증, 밸리데이션 및 시험(VV&T)"을 단일 개념으로 사용하고 있다.

소프트웨어 검증은 소프트웨어 개발 life cycle의 특정 단계의 설계 출력이 그 단계를 위한 특정 요구사항을 모두 만족하는 것을 객관적으로 증명하는 것이다. 소프트웨어 검증은 소프트웨어 및 관련 문서의 일관성(consistency), 완전성(completeness), 정확성(correctness)을 평가하고 소프트웨어가 유효하다는 결론을 뒷받침할 수 있게 한다. 소프트웨어 시험은 소프트웨어 개발 출력이 그 입력 요구사항을 만족시킨다는 것을 확인하기 위하여 의도된 많은 검증 활동 중의 하나이다. 다른 검증활동에는 다양한 정적·동적 분석, 코드 및 문서검사, walkthroughs, 그리고 그 밖의 다른 기술사항들을 포함한다.

소프트웨어 밸리데이션은 완성된 의료기기를 위한 설계 밸리데이션의 한 부분이지만 품질시스템 규정 내에서 독립적으로 정의되어 있지 않다. 이 지침의 목적을 위하여 FDA는 소프트웨어 밸리데이션을 "**시험에 의한 확인과 사용자의 요구사항·의도된 사용 목적을 위한 소프트웨어 규격확인 및 소프트웨어에 의해 실시된 특정 요구사항이 일관되게 실시될 수 있음을 보여주는 객관적 증거의 제공**"으로 간주한다.

모든 요구사항이 충족되었음을 보장하기 위하여 소프트웨어 밸리데이션은 소프트웨어 개발 life cycle의 마지막 단계뿐만 아니라 그 과정에서 실시될 수 있다. 일반적으로 소프트웨어는 종합적인 하드웨어 시스템의 일부분이기 때문에 소프트웨어 밸리데이션은 모든 소프트웨어 요구사항이 적절하고 완벽하게 실시되었고 시스템 요구사항에 추적성이 가능하도록 증빙자료를 포함하고 있다. 소프트웨어가 유효하다는 결론은 소프트웨어 개발 life cycle의 각 단계에서 실시되는 종합적 소프트웨어 시험, 검사, 분석 및 다른 검증 활동에 크게 영향을 받는다. 모의사용 환경(simulated use environment)에서 의료기기 소프트웨어 성능시험과 사용자 현장 시험은 자동화된 소프트웨어 의료기기의 전체적인 설계 밸리데이션 프로그램의 요소로써 포함된다.

소프트웨어 검증과 밸리데이션은 개발자가 계속 시험할 수도 없고 어느 정도의 정보로 충분한지 알기 어렵기 때문에 어렵다. 넓은 범위에서 소프트웨어 밸리데이션은 소프트웨어에 의하여 자동화된 의료기기의 성능과 특성에 대하여 모든 요구사항 및 사용자 기대사항이 만족되었는지에 대한 "신뢰의 수준(level of confidence)" 문제이다. 규격 문서, 잠재적인 결함의 평가, 시험범위 및 다른 기법에서 나타나는 결함에 대한 측정들은 제품을 인도하기 전에 수용 가능한 신뢰의 수준을 개발함에 있어 모두 활용된다.

신뢰의 수준, 필요한 소프트웨어 밸리데이션·검증·시험의 수준은 기기의 자동화 기능에 의해 제기되는 안전 위험(safety risk) 또는 위해요소(hazard)에 따라 달라질 수 있다. 소프트웨어의 안전한 위험관리를 위한 추가 지침은 FDA의 Guidance for the Content of Pre-market Submissions for Software Contained in Medical devices의 Section 4, Appendix A에 있는 국제기준인 ISO/IEC 14971-1 및 IEC 60601-1-4를 참조하기 바란다.

3.1.3 IQ/OQ/PQ

오랫동안 FDA와 규제 대상 업체들은 소프트웨어 밸리데이션을 공정 밸리데이션의 범위에서 이해하고 정의하려 하였다. 예를 들어 업계의 문서 및 다른 FDA의 밸리데이션 지침에서 사용자 환경 소프트웨어 밸리데이션을 설치 적격성확인(installation

qualification, IQ), 운영 적격성확인(operational qualification, OQ), 성능 적격성확인(performance qualification, PQ)의 용어로 정의하고 있다. IQ/OQ/PQ의 용어 정의와 관련 추가 정보는 FDA의 Guideline on General Principles of Process Validation(1987.5.11) 및 Glossary of Computerized System and Software Development Terminology(1995.8)에서 나타나 있다.

IQ/OQ/PQ 용어가 그 목적에 적절하게 사용되고 사용자 환경에서 소프트웨어 밸리데이션 작업을 체계화하는 많은 합법적인 방법 중 하나이지만, 이 용어는 많은 소프트웨어 전문가들 사이에서 충분히 이해되지 못하고 이 문서의 다른 부분에서는 사용되고 있지 않다. 그러나 FDA 직원 및 의료기기 제조업자는 소프트웨어 밸리데이션과 관련하여 질문하고 정보를 제공함으로써 이 용어의 차이점을 인식할 필요가 있다.

3.2. 시스템 설계의 일부로서 소프트웨어 개발

소프트웨어 사용에 의한 시스템 기능 실행에 대한 결정은 일반적으로 시스템 설계과정에서 이루어진다. 소프트웨어 요구사항은 전반적인 시스템 요구사항과 소프트웨어를 이용하여 실행되는 시스템 측면의 설계에서 기인한다. 이는 완성품에 대한 사용자 요구사항 및 의도된 사용목적이지만, 사용자는 이러한 요구사항이 하드웨어, 소프트웨어 또는 이 두 가지의 복합에 의해 만족되는지의 여부를 일일이 언급하지 않는다. 따라서 소프트웨어 밸리데이션은 시스템을 위한 전체적인 설계 밸리데이션의 맥락에서 고려되어야 한다.

개발 제품의 문서화된 요구사항 규격은 사용자 요구사항 및 사용목적을 나타낸다. 소프트웨어 밸리데이션의 중요한 목표는 모든 완성된 소프트웨어 제품이 모든 문서화된 소프트웨어 및 시스템 요구사항에 적합함을 증명하는 것이다. 시스템 요구사항과 소프트웨어 요구사항에 있어 정확성, 완전성은 설계 밸리데이션의 일부로서 반드시 검토하여야 한다. 밸리데이션은 모든 소프트웨어 규격에 적합함과 모든 소프트웨어 요구사항이 시스템 규격에 소급 가능함을 확인(confirmation)하는 것을 포함한다. 확인(confirmation)은 의료기기의 모든 측면이 사용자 요구사항과 사용목적에 적합함을 보증하기 위한 전반적인 설계 밸리데이션의 중요한 부분이다.

3.3. 하드웨어와 소프트웨어의 차이점

소프트웨어는 하드웨어처럼 유사한 기술을 많이 공유하고 있으나 중요한 차이점이 있다. 예를 들면,

- 대다수 소프트웨어 문제는 설계 및 개발과정동안 발생한 오류(error)로 추적가능하다. 하드웨어 제품 품질은 설계 및 개발, 제조에서 크게 영향을 받는 반면, 소프트웨어 제품 품질은 소프트웨어 제조를 위한 최소한의 고려사항과 함께 주로

설계 및 개발에서 영향을 받는다. 소프트웨어 제조는 쉽게 검증될 수 있는 재생산으로 이루어진다. 원본과 동일한 기능을 가지는 수천 개의 프로그램 복사본을 생산하는 것은 어렵지 않다; 어려운 것은 원본 프로그램이 모든 규격을 만족하도록 하는 것이다.

- 소프트웨어의 중요한 특징 중 하나는 브랜칭(branching, 다른 입력 기반의 명령어들을 실행시키는 능력)이다. 이러한 특징은 소프트웨어의 복잡성(complexity)과 같은 다른 특징에도 중요하게 작용한다. 간단한 프로그램이라 할지라도 매우 복잡하고 충분히 이해하는데 어려울 수 있다.
- 소프트웨어가 완벽하고 정확한지, 시험만으로 완벽히 검증할 수 없다. 종합적인 밸리데이션 접근을 확실히 하기 위하여 다른 검증 기술 및 체계화되고 문서화된 개발 공정이 병행되어야 한다.
- 하드웨어와는 달리 소프트웨어는 실체가 있는 것이 아니고 없어지는 것도 아니다. 소프트웨어는 잠재적인 결함이 발견되고 삭제되는 것처럼 시간이 지나면서 진화할 수 있다. 그러나 소프트웨어가 업데이트 및 변경되는 동안 새로운 결함이 소프트웨어에 발생함으로 그러한 개선 활동이 무효화되는 경우도 있다.
- 하드웨어 결함과는 달리 소프트웨어 결함은 사전 경고 없이 발생한다. 실행되는 동안 경로들을 다르게 따라가도록 하는 브랜칭은 소프트웨어 제품이 시장에 출시된 후에도 약간의 잠재적인 결함을 내재하고 있을 수 있다.
- 소프트웨어와 관련한 다른 특징은 변화 속도 및 용이성이다. 이로 인하여 소프트웨어 전문가와 비(非)전문가들은 소프트웨어 문제가 쉽게 수정될 수 있다는 믿음을 야기시킨다. 소프트웨어에 대한 이해부족은 관리자들이 하드웨어를 위한 엄격한 관리기술이 소프트웨어에는 필요 없다고 생각하도록 할 수 있다. 사실은 이와 반대이다. **그 복잡성 때문에 개발 프로세스에서 추후에 쉽게 발견될 수 없는 문제의 발생을 예방하기 위하여 소프트웨어는 하드웨어보다 더 엄격하게 관리되어야 한다.**
- 소프트웨어에서 중요하지 않은 것처럼 보이는 변경이 소프트웨어 프로그램과 다른 곳에서 예상하지 못한 심각한 문제를 발생시킬 수 있다. 소프트웨어 개발 프로세스는 소프트웨어 변경으로 인한 예상치 않은 결과를 발견하고 수정하기 위하여 충분히 계획, 관리 및 문서화 되어야 한다.
- 소프트웨어 전문가 및 첨단 모바일 종사자에게는 높은 요구사항이 주어지는 반면 소프트웨어 변경을 유지·보수하는 소프트웨어 종사자는 원본 소프트웨어 개발과정에 참여하지 않을 수 있다. 그래서 정확하고 전체적인 문서화가 필요하다.
- 소프트웨어 구성요소는 하드웨어 구성요소처럼 자주 규격화, 호환(interchangeable) 되지 않았다. 그러나 의료기기 소프트웨어 개발자들은 구성요

소 기반(component-based)의 개발도구와 기술을 사용하기 시작했다. 목적-지향 방법론(Object-oriented methodologies)과 OTS 소프트웨어 구성 요소의 사용으로 소프트웨어 개발은 빨라지고 비용이 절감되었다. 그러나 구성요소 기반의 접근은 통합(integration)할 때 매우 주의하여야 한다. 통합 전에 재사용 소프트웨어 코드를 충분히 정의, 개발하고 규격 구성요소의 기능을 충분히 이해할 수 있는 시간이 필요하다.

이러한 이유로 소프트웨어 기술은 하드웨어 기술보다 더 높은 단계의 관리·감독이 필요하다.

3.4. 소프트웨어 밸리데이션의 장점(BENEFITS)

소프트웨어 밸리데이션은 의료기기 소프트웨어 및 소프트웨어 자동화공정의 품질을 보증하는데 사용되는 중요한 수단이다. 소프트웨어 밸리데이션은 의료기기의 유용성과 신뢰도를 증가시킴으로 결함 발생율, 리콜 및 시정조치 감소, 환자 사용자에 대한 위험 감소, 의료기기 제조업자에 대한 부담 경감을 가져올 것이다. 소프트웨어 밸리데이션은 소프트웨어의 변경 및 변경의 유효성 재확인을 쉽고 적은 비용으로 가능하게 하여 장기적으로는 비용을 감소시킬 수 있다. 소프트웨어 life cycle에서의 전체 비용에 있어 소프트웨어 유지·보수는 많은 부분을 차지할 수 있다. 수립된 전체 소프트웨어 밸리데이션 프로세스는 추후 소프트웨어의 출고(release)를 위한 밸리데이션 비용을 절감함으로써 소프트웨어의 장기 비용을 줄이는데 도움이 된다.

3.5 설계 검토

설계 검토는 설계요구사항의 적절성(adequacy)을 평가하고 이런 요구사항에 부응하는 설계 능력평가와 문제점 식별을 위한 문서화된 전반적·체계적인 설계의 시험이다. 소프트웨어 개발과정에서 개발팀 내부의 많은 약식(informal) 기술검토가 있는 반면 정식적인 설계 검토는 좀 더 체계화되고 개발팀 외부인원이 참여한다. 정식적인 설계 검토는 다른 정식·약식 검토에서 나온 결과를 참조 또는 포함할 수 있다. 소프트웨어가 하드웨어와 함께 시스템에 통합된 후에 소프트웨어를 위해서 별도로 설계 검토를 실시할 수 있다. 설계 검토는 개발 계획의 시험, 요구사항 규격, 설계 규격, 시험 계획·공정, 프로젝트 관련 문서·활동, 규정된 life cycle의 각 단계의 검증결과 및 의료기기를 위한 전반적 밸리데이션 결과를 포함해야 한다.

설계 검토는 개발 프로젝트를 관리하고 평가하기 위한 주요도구이다. 예를 들어 정식 설계 검토는 경영자가 소프트웨어 밸리데이션 계획에서 정의된 모든 목표가 수행되었음을 확인하도록 한다.

품질시스템규정은 최소 1회의 정식 설계 검토가 의료기기 설계과정에서 수행 되도록 요구하고 있다. 그러나 각 소프트웨어 life cycle 활동의 마지막 단계 또는 다음 활동 진행 전과 같이 다양한 설계 검토를 실시하도록 권장한다. 정식 설계 검토는 주요 자원(resource)이 특정 설계 해결(solutions)로 되기 전인 요구사항 충족활동의 마지막 단계에서 매우 중요하다. 이 시점에서 발견된 문제점들은 보다 쉽게 해결되고 시간과 비용의 절감, 중요한 문제점을 놓치는 것과 같은 실수를 감소시킬 것이다.

다음과 같이 중요한 질문사항에 대한 대답을 정식 설계 검토과정에서 문서화하여야 한다.

- 각 소프트웨어 life cycle 활동별 적절한 작업, 예상 결과, 출력물 또는 제품을 설정하였는가?
- 각 소프트웨어 life cycle 활동의 작업, 예상 결과, 출력물 또는 제품은 다음을 만족 하는가;

√ 정확성, 완전성, 일관성 및 정확성의 관점에서 다른 소프트웨어 life cycle 활동의 요구사항에 적합한가?

√ 그 활동의 표준, 규정 및 약정을 만족하는가?

√ 추후 소프트웨어 life cycle 활동에 대한 초기 작업을 위하여 적절한 근거를 설정하였는가?

Section 4. 소프트웨어 밸리데이션의 원칙

이번 장에서는 소프트웨어 밸리데이션에 고려되어야 할 일반적인 원칙을 제시한다.

4.1. 요구사항

문서화된 소프트웨어 요구사항의 규격은 밸리데이션 및 검증을 위한 기준을 제공한다. 소프트웨어 밸리데이션 프로세스는 설정된 소프트웨어 요구사항의 규격없이 완료될 수는 없다(21 CFR 820.3(z), (aa) 및 820.20(f), (g)).

4.2. 결함 예방

소프트웨어 품질보증은 소프트웨어 개발 프로세스에 있어 결점 발생을 방지하고 코딩 후 소프트웨어 코드에 시험 품질을 시도하지 않는데 초점을 둘 필요가 있다. 소프트웨어 시험은 소프트웨어 코드 내에 잠재하는 모든 결점을 확인함에 있어 매우 제한적이다. 예를 들어 대부분 소프트웨어가 복잡하여 시험검사를 통하여 소프트웨어가 철저하게 테스트되지 못하게 한다. **소프트웨어 시험은 꼭 필요한 작업이다. 그러나 대부분의 경우 자체적인 소프트웨어 시험은 소프트웨어가 그 사용목적에 적합한지 나**

타내기 위해서는 충분하지 않다. 이 신뢰성을 수립하기 위하여 소프트웨어 개발자는 소프트웨어 오류를 방지하고 발생하는 소프트웨어 오류를 발견하기 위하여 방법 및 기법을 병행하여야 한다. 이 방법들의 "최상의 병행(best mix)"은 개발환경, 응용, 프로젝트 규모, 언어 및 위험을 포함한 많은 요소들에 달려있다.

4.3. 시간과 노력(EFFORT)

소프트웨어가 유효한지를 판단하기 위해서는 시간과 노력이 요구된다. 소프트웨어 밸리데이션을 위한 준비는 설계·개발 계획과 설계 입력과 같이 초기에 실시해야 한다. 소프트웨어 밸리데이션에 대한 최종 결정은 소프트웨어 life cycle 동안 수행된 계획적인 노력으로부터 수집한 증빙자료를 기초로 한다.

4.4. 소프트웨어 life cycle(LIFE CYCLE)

소프트웨어 밸리데이션은 설정된 소프트웨어 life cycle의 환경 내에서 실시한다. 소프트웨어 life cycle는 소프트웨어 기술 작업과 소프트웨어 밸리데이션 노력을 제공하기 위하여 필요한 문서화를 포함한다. 소프트웨어 life cycle는 소프트웨어 사용 목적에 적합한 특별한 검증 및 밸리데이션 작업을 포함한다. 이 지침은 소프트웨어 개발 프로젝트를 위해서 선택되고 사용돼야만 하는 특정 life cycle 모델을 제시하지는 않는다.

4.5. 계획

소프트웨어 밸리데이션 프로세스는 계획의 활용을 통해서 정의되고 관리된다. 소프트웨어 밸리데이션 계획은 소프트웨어 밸리데이션 노력(effort)을 통해 "무엇(what)"을 수행할 것인지를 정의한다. 소프트웨어 밸리데이션 계획은 중요한 품질시스템 수단이다. 소프트웨어 밸리데이션 계획은 적용범위, 접근(approach), 자원, 일정, 활동의 형태·범위(extent), 작업 및 업무요소(items)를 구체화 한다.

4.6. 공정

소프트웨어 밸리데이션 프로세스는 절차의 활용을 통해서 실행된다. 이러한 절차는 "어떻게(how)" 소프트웨어 밸리데이션 노력을 실시할 것인지를 설정한다. 이 절차는 개별적인 밸리데이션 활동, 작업 및 업무 사항을 완성하기 위하여 수행되어야 하는 특정 활동 또는 활동의 연속(sequence)을 명백하게 해준다.

4.7. 변경후 소프트웨어 밸리데이션

소프트웨어의 복잡성으로 인하여 좁은 범위의(local) 변경으로 인해 시스템에 광범위하게 영향(impact)을 미칠 수 있다. 소프트웨어에 어떠한 변경(매우 작은 변화라

할지라도)이 발생한 경우 소프트웨어의 밸리데이션 상태가 재설정되어야할 필요가 있다. **소프트웨어가 변경될 때마다 개별적인 변경에 대한 밸리데이션뿐만 아니라 전체 소프트웨어 시스템에 미치는 변경의 정도 및 영향을 결정하기 위하여 밸리데이션 분석이 실시되어야 한다.** 이 분석에 기초하여 소프트웨어 개발자는 변경되지 않는(unchanged) 시스템의 취약한 부분이 악영향을 받지 않는다는 것을 입증하기 위하여 적절한 소프트웨어 회귀시험(regression testing)의 수준을 결정하여야 한다. 설계 관리 및 적절한 회귀시험은 소프트웨어 변경 후에 소프트웨어가 유효하다는데 대한 신뢰성을 제공한다.

4.8. 밸리데이션 적용범위(COVERAGE)

밸리데이션 적용범위는 소프트웨어의 복잡성과 안전 위험(safety risk)에 기초하여야 한다. - 업소 규모 또는 자원제약에 기초하지 않음. 밸리데이션 활동, 작업 및 업무사항의 선택은 소프트웨어 설계의 복잡성과 특정한 사용목적을 위한 소프트웨어의 사용에 관련 있는 위험에 상응하여 이루어져야 한다. 저위험도 의료기기에 대해서는 기본적인 밸리데이션 활동이 실시될 수 있다. 위험증가에 따라 추가적인 밸리데이션 활동이 추가적인 위험처리를 위하여 확대되어야(added) 한다. 밸리데이션 문서화는 모든 소프트웨어 밸리데이션 계획과 절차가 적절하게 실시되었음을 증명하기에 충분해야 한다.

4.9. 검토의 독립성

밸리데이션 활동은 "검토의 독립성(independence of review)"이라는 기본적인 품질보증규칙(precept)에 의하여 실시되어야 한다. 자체 밸리데이션(self-validation)은 매우 어렵다. 가능한 독립적인 평가가 바람직하며, 고도의 위험성이 있는 경우 더욱 그러하다.

제3자에 의한 독립적인 검증 및 밸리데이션을 위하여 위탁을 하는 업체들도 있으나 이것이 항상 가능한 것은 아니다. 또 다른 방법은 특정 계발 또는 실행부서 소속되지 않은 프로젝트 평가와 검증・밸리데이션 실시에 대하여 역량이 있는 내부 직원을 선정하는 것이다. 소규모 업체일수록 어떻게 작업을 구성하고 내부적인 검토의 독립성을 유지할 것인가에 대하여 독창적일 필요가 있다.

4.10. 유연성 및 신뢰도

이러한 소프트웨어 밸리데이션 원칙은 구체적으로 실행함에 있어 매우 다양하게 적용된다. 의료기기 제조업자는 이러한 밸리데이션 원칙을 어떻게 적용할 것인지를 선택하는데 유연성을 가지되 소프트웨어기 유효하다는 것을 입증함에 있어 최종적인 책임이 있다.

소프트웨어는 광범위한 환경조건 및 다양한 위험수준을 가진 의료기기를 고려하여 설계, 개발, 밸리데이션 및 규제되어야 한다. FDA는 다음과 같은 소프트웨어를 포함하는 의료기기 응용프로그램을 규제하고 있다.

- 의료기기의 구성 요소, 부분품, 또는 부속품(accessory)인 소프트웨어
- 그 자체가 의료기기인 소프트웨어
- 제조, 설계, 개발, 또는 품질시스템의 다른 부분에 사용되는 소프트웨어

각 환경에서 여러 가지 원인(source)으로 인하여 소프트웨어 구성요소는 응용 프로그램을 만드는데 사용될 수 있다(예 : 사내 개발된 소프트웨어(in-house developed software), OTS 소프트웨어(off-shelf-software), 계약 소프트웨어, 세어웨어). 또한 소프트웨어 구성요소는 다양한 형태로 나타난다(예 : 응용프로그램 소프트웨어, 운영시스템, 컴파일러(compiler), 디버거(debugger), 구성 관리도구(configuration management tool) 등). 이런 조건에서 소프트웨어 밸리데이션은 복합해질 수 있다. 따라서 이러한 모든 소프트웨어 밸리데이션 원칙은 소프트웨어 밸리데이션 프로세스를 설계할 때 고려되는 것이 적절하다. 결과적으로 소프트웨어 밸리데이션 프로세스는 시스템, 의료기기 또는 프로세스와 관련된 안정 위험에 상응하여 이루어져야 한다.

소프트웨어 밸리데이션 활동 및 작업은 다른 장소에서 발생하고 다른 조직으로 전파되어 분산될 수 있다. 그러나 작업분배・계약관계・구성요소 원인 또는 개발환경에 관계없이 의료기기 제조업자 또는 특정 개발자는 소프트웨어가 유효하다는 것을 입증함에 있어 최종적인 책임이 있다.

Section 5. 활동과 작업(TASKS)

소프트웨어 밸리데이션은 소프트웨어 개발 life cycle의 다양한 단계에서 계획・실행되는 연속적인 활동과 작업을 통하여 실시되어야 한다. 이런 작업은 소프트웨어 프로젝트 프로세스로서 사용되는 life cycle 모델 및 발생되는 변경 범위에 영향을 받아서 단발적 또는 반복적으로 실시될 수 있다.

5.1. 소프트웨어 life cycle 활동

이 지침은 특정 소프트웨어 life cycle 모델을 권장하지는 않는다. 소프트웨어 개발자는 해당 제품 및 조직에 적절한 소프트웨어 life cycle 모델을 설정하여야 한다. 선택된 소프트웨어 life cycle 모델은 제품 초기부터 폐기 때까지 적용되어야 한다. 일반적인 소프트웨어 life cycle 모델의 활동은 다음과 같다.

- 품질 계획
- 시스템 요구사항 정의
- 구체적인 소프트웨어 요구사항에 대한 규격(specification)
- 소프트웨어 디자인 규격
- 구성 또는 코딩(coding)
- 시험
- 설치
- 운영 및 유지(support)
- 유지・보수(maintenance)
- 폐기(retirement)

소프트웨어 밸리데이션을 증명하는 검증, 시험 및 다른 작업들이 이런 활동의 각 단계에서 발생한다. life cycle모델은 다양한 방식으로 소프트웨어 개발 활동을 구성하고 소프트웨어 개발 프로젝트를 모니터링하고 관리하기 위한 체계를 제공한다. 몇 개의 소프트웨어 life cycle 모델(아주 빠른(waterfall), spiral, 빠른(rapid prototyping), 단계적인 개발(incremental development) 등)은 FDA의 Glossary of Computerized System and Software Development Terminology(1995.8)에 정의 되어있다. 이런 life cycle 모델은 Appendix A의 참고문헌에 설명되어 있다.

5.2. 밸리데이션을 위한 작업

각 소프트웨어 life cycle 활동은 소프트웨어가 유효하다는 결론을 뒷받침하는 "일반적인" 작업이다. 그러나 구체적인 수행 작업, 수행 작업순서 및 수행 작업의 반복・시간성(timing)이 선택된 특정 소프트웨어 life cycle 모델과 소프트웨어 적용과 관련한 안전 위험에 의하여 서술(dictated)되어야할 것이다. 저위험도의 응용프로그램은 특정 작업이 전혀 필요하지 않을 수 있다. 그러나 소프트웨어 개발자는 최소한 이러한 각 작업들을 고려하고 해당되는 특정한 적용에 적절한지의 여부를 정의하고 문서화하여야 한다.

다음의 논의는 특정 소프트웨어 life cycle 모델 또는 수행되는 특정 작업을 설명한 것이 아니라 일반적인 것이다.

5.2.1. 품질 기획

설계 및 개발 계획은 필요한 작업, 예외사항(anomaly) 보고 및 해결절차, 필요한 자원 그리고 정식 설계 검토를 포함하여 경영검토 요구사항을 규정한 계획을 통하여 완성된다. 각 소프트웨어 life cycle 활동을 위하여 필요한 작업뿐만 아니라 소프트웨어 life cycle 모델 및 관련 활동도 정의되어야만 한다. 이러한 계획은 다음 사항을 포함한다.

- 각 life cycle 활동을 위한 구체적 작업
- 중요한 품질요소의 열거(예 : 신뢰성, 유지성(maintainability) 및 유용성(usability)
- 각 작업을 위한 방법 및 절차
- 작업 수용 기준
- 입력 요구사항 확인을 평가할 수 있는 출력 정의 및 문서화에 대한 기준
- 각 작업을 위한 입력
- 각 작업의 출력
- 각 작업을 위한 규칙(role), 자원 및 책임
- 위험과 가정(assumptions)
- 사용자 요구사항의 문서화

관리부서는 적절한 소프트웨어 개발환경 및 자원을 파악하고 제공하여야 한다(21 CFR 820.20(b)(1) and (2)). 일반적으로 각 작업은 물적 자원뿐만 아니라 인적 자원(personnel)을 필요로 한다. 이런 계획에는 인력, 각 작업을 위한 시설·장비자원 및 위험관리를 위한 역할(role)을 명시하여야 한다. 형상관리 계획(configuration management plan)은 병렬적으로 수행되는 다양한 개발 활동을 관리하고 적절한 의사소통 및 문서화를 확실히 하도록 개발되어야 한다. 관리는 구체적인 문서, 소스 코드(source code), 오브젝트 코드(object code) 및 소프트웨어 시스템을 포함하는 시험원(test suites)이 승인된 모든 version 내에서 정확하게 일치됨을 확실히 하는 것이 필요하다. 또한 접근 가능한 승인된 version을 정확하게 식별하여야 한다.

절차는 밸리데이션 또는 다른 활동을 통하여 발생하는 소프트웨어 예외사항을 보고·해결하기 위하여 만들어야 한다. 관리는 기록을 식별하고 내용(contents), 형식 및 각 보고를 위한 책임 조직의 구성요소를 명시하여야 한다. 또한 검토 및 승인을 위하여 책임있는 조직의 구성원을 포함한 소프트웨어 개발 결과의 검토·승인을 위하여 절차가 필요하다.

<u>전형적 작업 - 품질 계획</u>
- 위험 관리 계획
- 형상 관리 계획(Configuration Management Plan)
- 소프트웨어 품질 승인 계획(Software Quality Assurance Plan)
  - 소프트웨어 검증 및 밸리데이션 계획
    · 검증 및 밸리데이션 작업 및 수용 기준
    · 소프트웨어 검증 및 밸리데이션 활동을 위한 일정 및 자원 배분

· 보고 시 요구사항
- 정규 설계 검토 요구사항
- 그 밖의 기술 검토 요구사항
- 문제점 보고 및 해결 절차
- 그 밖의 지원 활동

5.2.2. 요구사항

요구사항 개발은 의료기기 및 사용목적에 대한 정보의 식별, 분석 및 문서화를 포함한다. 특별히 중요한 범위는 하드웨어/소프트웨어, 운영 조건, 사용자 특성, 잠재적 위해요소(hazard) 및 예상 작업에 대한 시스템 기능의 배분(allocation)을 포함한다. 또한 요구사항은 소프트웨어의 사용목적을 정확하게 명시하여야 한다.

소프트웨어 요구사항을 규정한 문서는 소프트웨어 기능에 대한 문서화된 규정을 포함해야 한다. 이는 미리 확정되고 문서화된 소프트웨어 요구사항 없이는 소프트웨어의 밸리데이션을 수행하는 것은 불가능하다. 일반적으로 소프트웨어 요구사항은 다음을 명확히 한다.

- 모든 소프트웨어 시스템 입력
- 모든 소프트웨어 시스템 출력
- 소프트웨어가 수행할 모든 기능
- 소프트웨어가 만족시켜야 하는 모든 수행 요구사항(데이터 처리량(throughput), 신뢰성(reliability) 및 시간성(timing))
- 모든 내부의 소프트웨어와 시스템간의 인터페이스뿐만 아니라 모든 외부와 사용자 인터페이스의 정의
- 사용자가 시스템에 상호작용 할 수 있는 방법
- 오류 생성 요인과 처리방법
- 요구되는 응답시간(response time)
- 설계 제약이 있을시 소프트웨어를 위한 의도된 사용 환경(예 : 하드웨어 기반(platform), 운영시스템)
- 소프트웨어가 수용할 수 있는 모든 범위, 한계, 결함 및 특정 값(value)
- 소프트웨어에서 수행될 요구사항, 규격, 특징(feature) 또는 기능과 관련한 모든 안전성

소프트웨어 안정성 요구사항은 시스템 요구사항 개발 프로세스와 밀접하게 결합된 기술적인 위험관리 프로세스에서 기인한다. 소프트웨어 요구사항 규격은 소프트웨어에서 수행되는 모든 안정성 요구사항뿐만 아니라 시스템에서 소프트웨어 결함으로 발생할 수 있는 잠재적인 위해요소도 명확히 규정하여야 한다. 소프트웨어 결함의 영향은 결함을

완화시키는 방법과 함께 평가되어야 한다(예 : 하드웨어 완화, 방어 프로그래밍 등). 이런 분석으로부터 위험(harm)을 예방하는데 필요한 가장 적절한 방법을 명시하는 것이 가능하다.

품질시스템 규정은 불완전하고 불확실(ambiguous) 또는 일치하지 않는(conflicting) 요구사항을 검토하기 위한 메커니즘을 요구한다(21 CFR 820.30(c)). 소프트웨어 요구사항 규격에 명시된 각 요구사항(예 : 하드웨어, 소프트웨어, 사용자 운영 인터페이스 및 안전성)은 정확성(accuracy), 완료성(completeness), 일관성(consistency), 시험가능성(testability), 정확성(correctness) 및 명확성(clarity))을 위하여 평가되어야 한다. 예를 들어 소프트웨어 요구사항은 다음 사항을 검증하기 위해 평가되어야 한다.

- 요구사항 내에서 내부적으로 모순된 점(inconsistency)이 없어야 한다;
- 시스템을 위한 모든 수행 요구사항은 명쾌하게 설명되어야 한다;
- 결함의 허용오차, 안전성 및 보안 요구사항은 완벽하고 정확하다;
- 소프트웨어 기능의 할당은 정확하고 완전하다;
- 소프트웨어 요구사항은 시스템 위해요소(hazard)에 대하여 적절하다;
- 모든 요구사항은 측정 가능하거나 객관적으로 검증할 수 있는 용어로 설명 되어야 한다.

소프트웨어 요구사항에 대한 추적성 분석은 시스템 요구사항 및 위험 분석결과에 대하여 소프트웨어 요구사항을 추적하기 위하여 실시되어야 한다. 소프트웨어 요구사항을 검증하기 위하여 사용되는 모든 분석 및 문서화 외에, 요구사항이 충분히 명시화되어 있고 적절하다는 것을 확인하기 위하여 전체적인 소프트웨어 설계 작업이 시작되기 전에 정식 설계 검토과정이 권장된다. 요구사항은 계속적으로 승인되고 배포될 수 있으나 소프트웨어(및 하드웨어) 요구사항간의 상호영향 및 인터페이스가 적절하게 검토, 분석 및 관리하는 것에 주의를 기울여야 한다.

<u>전형적 작업 - 요구사항</u>

- 사전 위험 분석
- 추적성 분석
  - 시스템 요구사항에 대한 소프트웨어 요구사항(반대의 경우도 포함)
  - 위험 분석에 대한 소프트웨어 요구사항
- 사용자 특성에 대한 서술
- 1차 및 2차 메모리의 특성과 제한에 대한 목록

- 소프트웨어 요구사항 평가
- 소프트웨어 사용자 인터페이스 요구사항 분석
- 시스템 시험 계획의 생성(generation)
- 승인 시험 계획의 생성
- 모호성(ambiguity)에 대한 검토 또는 분석

5.2.3. 설계

설계 과정에서 소프트웨어 요구사항 규격은 소프트웨어가 수행할 수 있는 논리적이고 물리적인 표현으로 변환된다. 소프트웨어 설계 규격은 소프트웨어가 어떤(what) 기능을, 어떻게(how) 수행할 것인지에 대하여 설명한 것이다.

프로젝트의 복잡성 또는 다양한 기술적 책임의 수준을 지닌 사람들이 설계 정보를 명확하게 이해하도록 하기 위하여 설계 규격은 설계 및 세부 설계 정보에 대하여 높은 수준의 요약을 포함할 수 있다. 완성된 소프트웨어 설계 규격은 동의된 요구사항 및 설계 목적 내에서 프로그래머/코더(coder)가 작업하도록 한다. 완성된 소프트웨어 설계 규격은 프로그래머가 임기응변(ad hoc)으로 설계 결정을 내려야 하는 필요성을 경감시킨다.

소프트웨어 설계는 인적요소(human factor)를 명확히 하는 것이 필요하다. 지나치게 복잡하거나 동작(operation)에 대한 사용자의 직관적인 예상에 반대되는 설계로 인한 사용상의 오류는 FDA에 보고되는 가장 지속적이고 중대한 문제점 중의 하나이다. 소프트웨어 설계는 이러한 사용상 오류에 있어서 한 가지 요소이다. 인적요소공학(human factors engineering)은 의료기기 설계 요구사항, 분석 및 시험을 포함한 전체적인 설계 및 개발 프로세스에 포함되어야 한다. 의료기기 안전성 및 유용성(usability) 문제는 설계 흐름도(flowchart), 상태도, 표준(prototyping) 도구 및 시험계획을 할 때 고려되어야 한다. 또한 작업과 기능에 대한 분석, 위험 분석, 표준 시험, 검토 및 충분한(full) 유용성 시험이 수행되어야 한다. 사용자 집단(population)으로부터의 참가자는 이러한 방법론을 적용할 때 포함되어져야 한다.

소프트웨어 설계 규격에는 아래의 사항들을 포함해야만 한다.:

- 소프트웨어 수용에 있어 결정된 기준을 포함한 소프트웨어 요구사항 규격
- 소프트웨어 위험 분석
- 개발 절차 및 코딩 지침(또는 다른 프로그래밍 절차)

- 하드웨어, 소프트웨어 및 물리적 환경과의 관계를 포함하여 프로그램이 동작하기로 의도한 시스템의 개관(context)을 설명한 시스템 문서(예 : 서술 또는 개관도)
- 사용되는 하드웨어
- 측정 또는 기록되는 변수
- 논리적 구조(제어논리(control logic) 포함)와 논리적 처리단계(예: 알고리즘)
- 데이터 구조와 데이터 흐름도
- 변수(제어와 데이터)의 정의 및 사용되는 곳에 대한 기술
- 오류, 경보(alarm) 및 경고 메시지
- 보조 프로그램(예 : 운영 시스템, 드라이버, 다른 응용 소프트웨어)
- 전달 연결(communication links)(소프트웨어 내부 모듈 사이의 연결, 보조 소프트웨어와의 연결, 하드웨어와의 연결 및 사용자와의 연결)
- 보안 수단(물리적 및 논리적인 보안)
- 상기 요소들에서 언급되지 않은 모든 추가적인 constraints

위의 요소 중 처음의 4개는 소프트웨어 설계규격에 참고문헌으로 포함되는 것으로 별도의 미리 존재하는 문서이다. 소프트웨어 요구사항 규격은 소프트웨어 위험 분석으로서 이전 Section에서 언급하였다. 문서화된 개발 절차는 조직의 지침으로 제공되고 문서화된 프로그래밍 절차는 각 프로그래머에 대한 지침으로서 제공된다. 소프트웨어가 기능하는 시스템의 맥락에 대한 지식이 없이는 소프트웨어의 밸리데이션이 될 수 없기 때문에 문서가 참고 되어진다.

위의 요소 중 몇몇이 소프트웨어에 포함되어 있지 않을 경우 명확히 언급될 때 소프트웨어에 대한 향후 검토자 및 유지·보수하는 사람에게 도움이 될 것이다.(예 : 이 프로그램에 오류 메시지가 없음)

소프트웨어 설계과정에서 발생하는 활동은 여러 가지 목적이 있다. 소프트웨어 설계 평가는 설계의 완전성, 정확성, 일관성, 불확실성, 실행가능성 및 지속성 여부를 결정하기 위하여 수행된다. 설계과정에서 소프트웨어 구성(architecture, 예 : 모듈구조)에 대한 적절한 고려는 소프트웨어 변경이 필요할 때 향후 밸리데이션 노력의 양을 줄일 수 있다. 소프트웨어 설계 평가는 제어 흐름, 데이터 흐름, 복잡성, 시간성, 크기, 메모리 할당, 임계(criticality) 분석 및 다른 설계측면에 대한 분석을 포함할 수 있다. 추적성 분석(traceability analysis)은 소프트웨어 설계가 모든 소프트웨어 요구사항을 수행하는지를 검증하기 위하여 수행되어야 한다. 요구사항에 대한 식별을 하기 위한 기술이 불충분할 경우 추적성 분석은 모든 설계측면이 소프트웨어 요구사항에 대하여 추

적가능하다는 것을 검증할 수 있어야 한다. 상호의사소통 연결의 분석은 하드웨어, 사용자 및 관련 소프트웨어 요구사항을 고려하여 제안된 설계인지를 평가하기 위하여 수행되어진다. 소프트웨어 위험분석은 모든 추가적인 위해요인의 식별 여부 및 설계에 의하여 새로운 위해요인이 유도되었는지의 여부를 결정하기 위하여 재시험되어야 한다.

소프트웨어 설계 활동의 최종 단계에서 정규 설계 검토(Formal Design Review)는 설계를 수행하기 전에 설계의 옳음, 일관성, 완전성, 정확성 및 시험가능성을 검증하기 위하여 수행되어야 한다. 설계의 부분들은 수행을 위하여 단계적으로 승인 및 배포될 수 있다. 그러나 다양한 요소사이의 상호작용(interaction) 및 의사소통연결을 적절히 검토, 분석 및 제어하여야 한다.

대부분의 소프트웨어 개발 모델은 반복될 것이다. 이는 소프트웨어 요구사항 규격과 소프트웨어 설계 규격이 여러 가지 version으로 나오는 것과 같다. 모든 승인된 버전은 수립된 형상관리절차에 따라서 구성 및 제어되어야 한다.

전형적 작업 - 설계

- 업데이트 된 소프트웨어 위험분석
- 추적성 분석 - 소프트웨어 요구사항에 대한 설계 규격(설계 요구사항에 대한 소프트웨어 규격)
- 소프트웨어 설계 평가
- 설계 상호의사소통 연결 분석
- 모듈 시험 계획 시기(generation)
- 통합 시험 계획 시기
- 설계 시험 시기(모듈, 통합, 시스템 및 수용)

5.2.4 구조Construction) 또는 코딩(Coding)

소프트웨어는 새로운 응용프로그램 사용을 위하여 코딩(프로그래밍)하거나 미리 코딩된 소프트웨어 구성요소들(예 : 코드 라이브러리, OTS 소프트웨어 등)을 같이 조합하여 구성될 수 있다. 코딩은 세부적인 설계 규격이 소스 코드(source code)로서 수행되는 소프트웨어 활동이다. 코딩은 소프트웨어 개발 프로세스 중 가장 낮은 수준의 작용(abstraction)이다. 모듈 규격이 프로그래밍 언어로 변환(translation)되는 것은 소프트웨어 요구사항의 분해(decomposition)에 있어서는 최종 단계 이다.

코딩은 보통 높은 수준의 프로그래밍 언어의 사용을 필요로 하지만 임계시간(time-critical) 동작을 위하여 어셈블리 언어(assembly language)(또는 마이크로코드)의 사용을 수반할 수도 있다. 소스 코드는 대상 하드웨어 기반에서 사용을 위하여 컴파일되거나 변환(interpreted)될 수 있다. 프로그래밍 언어와 소프트웨어 빌드 도구(어셈블러(assembler), 링커(linker), 컴파일러(compiler))의 선택에 대한 결정은 결과적인 품질 평가 작업(예 : 선택된 언어를 위한 디버깅 및 시험도구의 가능성)에 미치는 영향을 고려하여야 한다. 몇몇 컴파일러는 코드 디버깅시 보조적으로 오류확인을 위한 선택 수준(optional level) 및 명령어를 제공한다. 오류 확인의 여러 가지 수준은 코딩과정을 통하여 이용될 수 있고 컴파일러에서 오는 경고 또는 다른 메시지는 기록되거나 그렇지 아니할 수 있다. 그러나 코딩 및 디버깅 프로세스의 최종 단계에는 무슨 컴파일레이션(compilation) 오류가 소프트웨어에 여전히 남아있는지를 문서화하기 위하여 가장 엄격한 오류 확인 수준이 이용되어진다. 가장 엄격한 오류 확인 수준이 최종 소스 코드 전환시 사용되지 않을 경우 덜 엄격한 전환 오류 확인의 사용을 위한 조정(justification)이 문서화되어야 한다. 또한 최종 컴파일레이션을 위하여 모든 경고 또는 컴파일에서 오는 메시지를 포함한 컴파일레이션 과정 및 그 결과 또는 미결된 문제점을 남기는 결정을 위한 조정에 대하여 문서화 하여야 한다.

업소는 소프트웨어 코딩 프로세스와 관련하여 품질방침 및 절차를 설정하는 규격 코딩 지침을 채택한다. 소스 코드는 규격화된 코딩 지침에 적합함을 검증하기 위하여 평가되어야 한다. 이러한 지침은 명확성(clarity), 형식, 복잡성, 관리 및 해설과 관련하여 코딩 합의사항을 포함하여야 한다. 코드 해설은 예상 입·출력, 참조 변수, 예상 데이터 형식 및 수행 동작을 포함한 모듈을 위해 유용하고 기술적인 정보를 제공하여야 한다. 소스 코드는 세부적인 설계 규격에 적합함을 검증하기 위하여 평가되어야 한다. 통합 및 시험을 위한 모듈은 코딩 지침 및 다른 적용되는 품질방침·절차에 적합하도록 문서화 하여야 한다.

소스 코드 평가는 코드 검사 및 코드 walkthrough로서 수행될 때도 있다. 이런 통계적 분석은 코드의 실행 전에 오류를 파악할 수 있는 매우 효과적인 방법이다. 이는 격리된 상황에서 각 오류의 시험을 가능하게 하고 추후 소프트웨어의 동적인 시험에 집중할 수 있도록 해준다. 업소는 일관성과 독립성을 확실히 하기 위한 적절한 관리 기능을 가진 수동 확인을 사용할 수 있다. 소스 코드 평가는 모듈과 레이어(수평적, 수직적 인터페이스) 및 그 설계 규격에 적합성 사이의 내부적인 연계에 대한 검증까지 확대되어야 한다. 사용되는 절차의 문서화 및 소스코드 평가의 결과는 설계 검증의 일부분으로써 관리되어야 한다.

소스 코드의 추적성 분석은 모든 코드가 설정된 규격 및 시험 절차와 연결이 되어있는지를 검증하기 위한 중요한 도구이다. 소스 코드 추적성 분석은 다음과 같은 사항을 검증하기 위하여 수행 및 문서화 되어야 한다.

- 소프트웨어 설계 규격의 각 요소들은 코드로 구현되었다
- 코드로 구현된 모듈 및 기능은 소프트웨어 설계 규격의 요소와 위험분석까지 추적될 수 있다.
- 모듈 및 기능에 대한 시험은 소프트웨어 설계 규격의 각 요소와 위험분석까지 추적될 수 있다.
- 모듈 및 기능에 대한 시험은 동일한 모델과 기능을 위한 소스코드까지 추적될 수 있다.

전형적 작업 - 구조 또는 코딩

- 추적성 분석
  - 설계 규격을 위한 소스코드(및 소스코드를 위한 설계 규격)
  - 소스코드 및 설계 규격을 위한 시험 사례(cases)
- 소스코드 및 소스코드 문서화 평가
- 소스코드 인터페이스 분석
- 시험 절차 및 시험 예제 시기(모듈, 통합, 시스템 및 수용)

5.2.5 소프트웨어 개발자에 의한 시험

소프트웨어 시험은 미리 규정된 예상과 비교될 수 있는 규정된 입력 및 문서화된 출력의 조건하에서 소프트웨어 제품을 실행을 하는 것이다. 이는 시간 소모적이며 어렵고 불완전한 활동이다. 또한 효과적이고 효율성 있기 위해서는 초기에 계획되는 것이 요구된다.

시험 계획과 시험 사례는 가능한 소프트웨어 개발 프로세스 초기에 수행되어야 한다. 이것들은 일정, 환경, 자원(인력, 도구 등), 방법론, 사례(입력, 절차, 출력 예상 결과), 문서화 및 보고 기준을 명확히 하여야 한다. 시험프로세스를 통하여 적용되는 노력의 크기는 복잡성, 임계(criticality), 신뢰성 및/또는 안전성의 문제(예 : 요구 기능 또는 그 결함 허용(tolerance) 형상의 중요한 시험에 적용되는 특별한 결과를 생성하는 모듈)와 연결될 수 있다. 소프트웨어 및 소프트웨어 시험 노력의 범주에 대한 설명은 여러 문헌에 나타나 있다. 예를 들면 다음과 같다.

- NIST Special Publication 500-235, **Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric**;
- NUREG/CR-6293, **Verification and Validation Guidelines for High Integrity**

Systems; and

- IEEE Computer Society Press, **Handbook of Software Reliability Engineering**.

소프트웨어 시험 계획은 각 개발 단계에서 수행되는 특별한 작업을 명확히 하고 관련 완전성 기준에 의하여 나타내어지는 노력 수준의 정당화를 포함하여야 한다.

소프트웨어 시험은 특정 소프트웨어 제품의 시험을 계획할 때에 인지하고 고려해야 하는 한계를 가지고 있다. 가장 간단한 프로그램을 제외하고 소프트웨어는 철저하게 시험될 수는 없다. 일반적으로 모든 가능한 입력을 가지고 소프트웨어 제품을 시험하는 것은 불가능하고 프로그램 실행시 발생할 수 있는 모든 가능한 데이터 프로세싱 경로를 시험하는 것도 불가능하다. 특정 소프트웨어 제품이 완전하게 시험되었다고 확신할 수 있는 단 한 가지 시험 형식 또는 시험방법론이 있는 것은 아니다. 한 프로그램의 모든 코드를 시험하는 것이 프로그램에 모든 필요한 기능이 있다는 것을 의미하는 것은 아니다. 모든 프로그램 기능 및 코드에 대한 시험이 프로그램이 100% 정확하다는 것을 의미하는 것은 아니다. 오류가 없음을 나타내는 소프트웨어 시험결과가 소프트웨어 제품에 오류가 없다는 것으로 해석되어서는 아니 된다. 이는 시험이 피상적이라는 것을 의미할 수 있다.

소프트웨어 시험사례의 필수적 요소는 결과 이다. 실제적인 시험 결과의 객관적인 평가를 허용하는 것이 중요 사항이다. 필수 시험 정보는 관련성이 있고 미리 정의된 정의 또는 규격으로부터 획득된다. 소프트웨어 규격 문서는 무엇을(what), 언제(when), 어떻게(how), 왜(why) 등을 명확히 하여야 하고 그것이 시험을 통하여 확인되기 위하여 구체적인 공학(예 : 측정가능하거나 객관적으로 검증이 가능한)수준으로 수행되어야 한다. 효과적인 소프트웨어 시험의 실제적인 노력은 시험의 결과보다는 무엇을 시험해야 하는지 규정하는 것이다.

소프트웨어 시험 프로세스는 소프트웨어 제품의 효과적인 시험을 촉진하려는 원칙을 기반으로 하고 있다. 응용 소프트웨어 시험은 다음의 사항을 포함한다.

- 예상되는 시험 결과는 미리 정의한다.
- 우수 시험사례는 오류 노출(exposing an error)의 가능성이 높다.
- 성공적인 시험은 오류를 발견하는 것이다
- 코딩으로부터 독립적이다.
- 사용자 및 소프트웨어(프로그래밍) 전문가가 고용되어진다.

- 시험자는 코더와는 다른 도구를 사용한다.
- 단지 일반적인 사례를 시험하는 것은 불충분하다
- 시험 문서화는 재사용이 가능하고 추후 검토과정에서 시험결과의 적합(pass)/부적합(fail) 상태에 대한 독립적인 확인이 가능하다

전제가 되는 작업(예 : 코드 검사)이 성공적으로 완료되면 소프트웨어 시험은 시작된다. 이는 단위 수준 시험에서 시작하고 시스템 수준의 시험에서 종결된다. 여기에는 시험수준의 통합이 별도로 있을 수 있다. 소프트웨어 제품은 그 내부적인 구조에 기초한 시험사례 및 외부의 규격에 기초한 시험사례를 가지고 수행되어야 한다. 이 시험은 소프트웨어의 기능성, 실행 및 인터페이스 규정 및 요구사항의 적합성에 대한 전체적이고 엄격한 시험을 제공하여야 한다.

코드 기반(code-based) 시험은 구조 시험 또는 “white-box" 시험[27]으로도 알려져 있다. 이는 소스코드, 구체적인 설계 규격 및 다른 개발 문서로부터 획득된 지식을 기초로 하는 시험 사례들을 명확히 한다. 이 시험사례들은 프로그램에서 만들어진 관리결정(control decision) 및 형상(configuration) 표를 포함하는 프로그램의 데이터 구조를 수행한다. 구조적인 시험은 프로그램이 실행될 때 한번도 수행되지 않는 dead 코드를 명확히 할 수 있다.

구조적인 시험은 단위(모듈) 수준 시험으로 우선적으로 수행되어지지만 소프트웨어 시험의 다른 수준까지 확대될 수는 없다.

구조적인 시험의 수준은 구조적인 시험과정에서 평가되는 소프트웨어 구조의 비율이 얼마인지를 보여주기 위한 측정기준(metrics)을 사용하여 평가될 수 있다. 이런 측정기준은 일반적으로 “범위(coverage)"로 참고 되고 시험 선택기준을 고려한 완전성의 측정이다. 구조적인 범위의 크기는 소프트웨어에 의하여 제기된 위험의 수준에 동일하여야 한다. ”범위“라는 용어의 사용은 일반적으로 100% 커버리지(coverage)를 의미한다. 예를 들면 시험 프로그램이 ”명령문 커버리지(statement coverage)"를 수행하였다면 이는 소프트웨어의 명령문 100%가 적어도 한번은 실행되었다는 것을 의미한다. 일반적인 구조적인 측정기준은 다음의 사항을 포함한다.

- **명령문 범위(Statement Coverage)** - 이 기준은 최소 한번은 각 프로그램 명령

27) 제품의 내부수행동작을 알고 있을 때 명세화에 따른 내부수행 동작을 시험하는 것으로 순차적이고 세부적이며 소프트웨어의 논리적인 경로, 조건 반복과 같은 부분들을 시험하는 것이다. 발생가능한 모든 경우 및 경로를 시험하는 것을 특히 소모적 시험(Exhaustive test)이라 하는데 이는 현실적으로 어려워 중요부분을 대상으로 White-box test가 이루어진다.

어가 수행되기 위하여 충분한 시험사례들을 요구한다. 그러나 이들의 수행은 소프트웨어 프로그램의 반응에서 신뢰성을 제공하는 데는 불충분하다.

- **결정(Decision)(분기, Branch) 범위** - 이 기준은 각 가능한 결과가 최소 한번은 발생할 수 있도록 실행되어지는 각 프로그램 결정 또는 분기를 위한 충분한 시험사례를 요구한다. 이는 대부분의 소프트웨어 제품을 위한 범위의 최소 수준으로 고려되어지나 결정 범위 단독으로는 높은 수준의 통합 적용(high-integrity application)을 위해서는 불충분하다.

- **조건 범위** - 이 기준은 최소 한번은 모든 가능한 결과에서 수행하여야 하는 프로그램 결정에서 각 조건을 위한 충분한 시험사례를 요구한다. 이는 단지 다양한 조건이 결정에 이르기 위하여 평가되어져야 할 때의 분기 범위와는 다르다.
- **다양한 조건 범위** - 이 기준은 프로그램 결정에서 모든 가능한 조건 조합을 수행하기 위한 충분한 시험사례를 요구한다.

- **루프(loop) 범위** - 이 기준은 초기화(initialization), 일반적인 실행 및 종료(한계, boundary) 조건을 포함하는 0, 1, 2 및 많은 반복을 위하여 실행되는 모든 프로그램 루프를 위한 충분한 시험사례를 요구한다.

- **경로 범위** - 이 기준은 정의된 프로그램 구획의 시작부터 종료까지 최소 한번은 실행되는 각 실행 가능한 경로, 기본 경로 등을 위하여 충분한 시험사례를 요구한다. 소프트웨어 프로그램을 통틀어 가능한 경로의 수가 매우 많기 때문에 일반적으로 경로 범위는 전수될 수는 없다. 보통 경로 범위의 크기(amount)는 시험하에서 소프트웨어의 위험 또는 임계(criticality)에 기초하여 설정된다.

- **데이터 흐름 범위(Data Flow Coverage)** - 이 기준은 최소 한번은 수행되는 각 실행 가능한 데이터 흐름을 위하여 충분한 시험사례를 요구한다. 많은 데이터 흐름 시험전략이 가능하다.

정의기반(definition-based) 또는 규격기반(specific-based) 시험은 또한 기능적 시험 또는 "black-box"시험[28]으로 알려져 있다. 이는 소프트웨어 제품(한 개의 단위(unit)((모듈)

28) 제품의 수행기능을 알고 있을 때 각 기능의 완전한 동작을 증명해 보이는 것으로 소프트웨어 인터페이스에서 실시되는 시험을 의미한다. 즉, 소프트웨어 기능의 작동과 입력의 적합성, 출력의 정확성, 자료파일과 같은 외부 정보의 무결성을 입증하는 것이다. 이는 소프트웨어 내부 논리적 구조는 고려하지 않고 기본적인 시스템 모델의 기능을 시험하는 것이다.

또는 완전한 프로그램)이 목적으로 한 동작에 대하여 정의내린 것에 기초한 시험사례에 대하여 명확히 하여야 한다. 이 시험사례는 사용목적 또는 프로그램의 기능성 및 프로그램의 내·외적 인터페이스를 수행한다. 기능적 시험은 각 단위에서부터 시스템 수준의 시험까지 소프트웨어 시험의 전 수준에 적용될 수 있다.

다음의 기능적 시험의 형식은 일반적으로 노력의 수준 향상을 포함한다.

- **일반적 사례** - 일반적인 입력에 대한 검사가 필요하다. 그러나 예상되는 확실한 입력만을 가지는 소프트웨어 제품에 대한 시험은 그 소프트웨어 제품에 대한 완벽한 시험은 아니다. 그것만으로는 일반적인 사례 시험은 소프트웨어 제품의 확실성(dependability)에 있어 충분한 신뢰를 제공할 수 없다.
- **출력 forcing** - 선택된(또는 모든) 소프트웨어 출력을 확실히 하기 위한 시험 입력을 선택하는 것은 시험에 의하여 이루어진다.
- **Robustness** - 소프트웨어 시험은 예상하지 못한 무효한(invalid) 입력이 주어진 경우 소프트웨어 제품이 정확하게 동작한다는 것을 증명하여야 한다. 이런 충분한 일련의 시험 사례를 명확히 하기 위한 방법으로 동일 등급 구분(Equivalence Class Partitioning), 경계값 분석(Boundary Value Analysis) 및 특별 사례 확인(Special Case Identification)(Error Guessing)이 있다.
- **입력의 조합** - 위에서 명시된 모든 기능적 시험 방법이 개별적 또는 단일 시험 입력을 강조하고 있다. 대부분의 소프트웨어 제품은 그 사용 환경에서 다양한 입력을 가지고 동작한다. 소프트웨어 제품 시험은 소프트웨어 단위 또는 시스템이 동작과정에서 일으키는 입력의 조합을 고려하여야 한다.

기능 및 구조적인 소프트웨어 시험사례 확인(identification) 기술은 임의의(random) 시험 입력보다 시험을 위한 특정 입력을 제공한다. 이 기술에서의 결점(weakness)은 소프트웨어 제품의 신뢰도(reliability)를 위한 구조·기능적 시험 완료 기준 연결(linking)에 있어 어려움이 있다. 통계적 시험과 같이 향상된 소프트웨어 시험방법은 소프트웨어의 신뢰성을 좀 더 확실히 하기 위하여 이용될 수 있다. 통계적 시험은 수행도(operational profile)(예 : 소프트웨어의 예상된 사용, 위험한 사용 또는 부당한 사용)에 기초하여 정의된 분포(distribution)로부터 생성된 시험 데이터를 임의로 사용한다. 시험 데이터의 많은 부분은 소프트웨어 제품의 개발자 또는 시험검사자가 예상하지 못한 개별적이고 다양한 예외적인 동작 환경을 확인하기 위한 증가된 가능성을 제공하는 특정한 영역 또는 문제점을 방어하기 위하여 수행되고 목표로 한다. 통계적 시험은 또한 높은 구조적 적용(coverage)을 제공한다. 이는 안정된 소프트웨어 제품을 요구하는 것은 아니다. 그러므로 구조·기능적시험은 소프트웨어 제품의 통계적 시험을 위하여 필수적이다.

소프트웨어 시험의 다른 측면은 소프트웨어 변경의 시험이다. 변경은 소프트웨어 개발 과정에서 자주 발생한다. 이러한 변경은 1) 오류 발견 및 수정하는 디버깅(debugging), 2) 신규 또는 변경된 요구사항(“requirements creep") 및 3) 더 효과적이고 효율적인 실행이 파악될 경우 변경된 설계의 결과이다. 한 소프트웨어 제품이 승인되어지면 그 제품을 위한 모든 변경은 시험을 포함한 자체의 ”최소 life cycle(mini life cycle)"를 가져야 한다. 변경된 소프트웨어 제품의 시험은 추가적인 노력을 요구한다. 이는 그 변경이 정확하게 수행되어진다는 것뿐만 아니라 시험은 또한 변경사항이 소프트웨어 제품의 다른 영역에 부작용을 일으키지 않는다는 것을 증명하여야 한다. 회귀분석(Regression analysis)[29] 및 시험은 변경이 소프트웨어 제품의 다른 곳에서 문제를 발생하지 않는다는 것을 확실히 하기 위하여 이용되어진다. 회귀분석은 실행되는 필요한 회귀 시험을 확실히 하기 위하여 관련 문서(예 : 소프트웨어 요구사항 규격, 소프트웨어 설계 규격, 소스 코드, 시험 계획, 시험 사례, 시험설명서 등)의 검토에 기초한 변경의 영향에 대한 문서화이다. 회귀검사는 프로그램이 본래 정확하게 수행해온 시험사례의 재실행(rerunning)이고 의도하지 않은 소프트웨어 변경의 효과를 파악하기 위하여 현재의 결과를 이전의 결과와 비교하는 것이다. 또한 회귀분석 및 회귀검사는 신규로 통합된 모듈이 이전에 통합된 모듈의 동작에 반대 영향을 주지 않는다는 것을 확실히 하기 위한 소프트웨어 제품을 설계하기 위하여 통합방법을 사용하여 소프트웨어 제품을 설계할 때 도입된다.

철저하고 엄격한 소프트웨어 제품의 검사(examination)를 위하여 개발 시험은 일반적으로 단계적(level)으로 설정되어 있다. 예를 들어 소프트웨어 제품의 시험은 단위, 통합 및 시스템 시험의 단계로 체계화될 수 있다.

1) 단위(모듈 또는 구성요소) 수준의 검사는 보조 프로그램(sub-program) 기능에 대한 초기 시험과 시스템 단계에서 보이지 않는 기능이 시험에 의해 검사된다는 것을 확실히 하는데 초점을 두고 있다. 단위 시험(unit testing)은 품질 소프트웨어 단위가 최종 소프트웨어 제품으로의 통합을 위하여 제공된다는 것을 확실히 한다.

2) 통합 단계 시험(integration level testing)은 프로그램의 내·외부 인터페이스에 걸친 데이터 및 제어(control)의 이전에 초점을 두고 있다. 외부 인터페이스는 소프트웨어(운영시스템 소프트웨어 포함), 시스템 하드웨어 및 사용자가 있고 전달 연결(communication link)처럼 묘사될 수 있다.

29) 회귀분석(Regression analysis) 이란 변수들 사이의 관계를 조사하여 모형화 시키는 통계적 기법으로서 경제, 경영, 교육, 정치 등의 사회과학 그리고 물리, 화학, 생물, 공학, 농학, 의학 등 자연과학의 거의 모든 분야에서 널리 응용되고 있다. 즉, 변수들 간의 함수적인 관련성을 규명하기 위하여 수학적 모형(통계모형)을 가정하고, 관측된 자료로부터 이 모형을 추정하는 통계분석방법으로 주로 예측에 사용된다.

3) 시스템 단계 검사(system level testing)는 모든 규격화된 기능이 있고 소프트웨어 제품이 신뢰할 만하다는 것을 증명한다. 이 시험은 규격화된 운영 기반(operating platform)에 표시된 것처럼 소프트웨어 제품을 위한 요구사항을 고려한 완성된(as-built) 프로그램의 기능 및 성능을 검증한다. 시스템 단계 소프트웨어 시험은 기능적 문제(concern) 및 사용목적과 관련 의료기기 소프트웨어의 다음과 같은 요소들을 다룬다.
   - 성능 문제(performance issues)(예 : 반응시간, 신뢰도 측정)
   - 가혹조건(stress conditions)에 대한 반응. 예 : 최대 하중(load), 지속적인 사용하의 반응(behavior)
   - 내・외부 보안 특성(feature)의 운영
   - 손실 회복(disaster recovery)을 포함한 회복 절차의 유효성
   - 유용성(usability)
   - 다른 소프트웨어 제품에 대한 호환서(compatibility)
   - 정의된 각 하드웨어 형상(configuration)에서의 반응
   - 문서화의 정확성(accuracy)

제어 방법(control measures)(예 : 추적성 분석)은 의도한 범위가 수행되었는지를 확실히 하기 위하여 사용되어야 한다.

또한 시스템 단계 시험은 의도한 운영 환경에서 소프트웨어 제품의 반응을 나타낸다. 이러한 시험의 위치는 목표(target) 운영 환경을 위한 소프트웨어 개발자의 능력에 좌우된다. 환경에 따라 잠재적인 사용자 위치에서의 모의실험(simulation) 및/또는 시험은 이용될 수 있다.

계획된 시스템 단계 시험이 소프트웨어 개발자에 의하여 직접적으로 관리되지 않는 현장(site)에서 수행될 때 시험계획은 사용범위가 설정되고 적절한 문서화가 준비되었는지를 확실히 하기 위하여 필요한 관리(control)을 명확히 하여야 한다. 또한 FDA 승인(clearance) 전 인체에 사용되는 의료기기 또는 그 구성요소인 소프트웨어 제품에 대하여 인체 피실험자(human subject)를 포함하는 시험은 IDE(Investigational Device Exemption) 또는 IRB(Institutional Review Board) 승인을 요구할 수 있다.

시험절차, 시험 데이터 및 시험결과는 이루어진 객관적인 적합/부적합 결정을 수락하는 방식(manner)에서 문서화되어야 한다. 또한 검토 및 시험 실시를 수반하는 객관적인 결정을 내리는 것에 적절하여야 하고 추후에 발생하는 모든 회귀시험에서의 사용에 적절하여야 한다. 시험과정에서 발견되는 오류는 소프트웨어 배포(release)전에 기록

(log), 분류(classify), 검토 및 해결(resolve)되어야 한다. 개발life cycle(development life cycle)동안 수집 및 분석된 소프트웨어 오류 데이터는 상업적 판매를 위한 배포를 위하여 소프트웨어의 적합성을 결정하는데 사용될 수 있다. 시험 보고서는 관련 시험계획의 요구사항에 적합하여야 한다.

의료기기 또는 해당 기기의 생산에서 유용한 기능을 수행하는 소프트웨어 제품은 복잡한 것도 있다. 소프트웨어 시험도구는 이러한 소프트웨어 제품의 시험에서 일관성(consistency), 완벽성(thoroughness) 및 효율성(efficiency)을 확실히 하고 계획된 시험 활동 요구사항을 만족하기 위하여 자주 사용된다. 이러한 도구는 상업용 소프트웨어 시험도구 뿐만 아니라 단위(모듈) 시험 및 수반되는 통합시험(예 : 드라이버 및 stub)을 도와주기 위하여 자체 개발된(built-in-house) 보조 소프트웨어를 포함할 수 있다. 이러한 도구는 개발에 사용되는 소프트웨어 제품과 같은 품질의 등급(degree)을 가져야 한다. 그 사용목적을 위한 소프트웨어 도구의 밸리데이션 증거를 제공하는 적절한 문서화가 유지되어야 한다(동 지침의 Section 6 참조)

<u>전형적인 작업 - 소프트웨어 개발자에 의한 시험</u>

- 검사 계획
- 구조적 시험사례 확인(Structural Test Case Identification)
- 기능적 시험사례 확인(Functional Test Case Identification)
- 추적성 분석(Traceability Analysis) - 시험
  - 세부적인 설계를 위한 단위(모듈) 시험
  - 고수준(high level) 설계를 위한 통합 시험
  - 소프트웨어 요구사항을 위한 시스템 시험
- 단위(모듈) 시험 실행
- 통합시험 실행
- 기능적 시험 실행
- 시스템 시험 실행
- 수용 시험 실행(Acceptance Test Execution)
- 시험결과 평가
- 오류 평가/해결
- 최종 시험 보고

5.2.6. 사용자 현장 시험(User Site Testing)

사용자 환경에서의 시험은 소프트웨어 밸리데이션에서 필수적인 부분이다. 품질관리

규정은 적절한 설치를 증명하기 위한 심사(inspection) 및 시험의 문서화뿐만 아니라 설치 및 심사절차(적절한 장소에서의 시험 포함)를 요구한다(21 CFR 820.170 참고). 또한 제조장비는 규격화된 요구사항을 반드시 만족하여야 하고 자동화된 시스템은 그 사용목적을 위하여 반드시 밸리데이션되어야 한다(21 CFR 820.70(g) 및 820.70(i) 참고).

사용자 환경시험에 관한 전문용어는 혼란스러울 수 있다. 베타 시험(beta test), 환경 밸리데이션(site validation), 사용자 수용 시험(user acceptance test), 설치 검증(installation verification) 및 설치 시험(installation testing)과 같은 용어는 사용자 환경 시험을 설명하기 위하여 사용되어 왔다. 이 지침의 목적에 따라 "사용자 환경시험"이라는 용어는 이러한 모든 것을 포함하고 개발자 조정 환경 범위의 외부에서 일어나는 다른 모든 시험도 포함한다. 이 시험은 설치되는 시스템 형상(configuration)의 일부가 될 수 있는 실제 하드웨어 및 소프트웨어를 갖춘 사용자 환경에서 수행되어야 한다. 이 시험은 목적하는 기능 범위 내에서 시험되는 소프트웨어의 실제 또는 가상사용을 통하여 수행되어진다.

동 지침은 일반적이고 모든 사용자 환경시험에 적용될 수 있는 것을 포함하고 있다. 그러나 일부 범위(예 : 혈액 설정 시스템 등)에서는 사용자 환경시험 계획에서 고려되어져야할 특별한 환경 밸리데이션 문제가 있을 수 있다. 시험 계획자는 사용자 환경시험을 위하여 추가적인 법적 요구사항이 있는지의 여부를 결정하기 위한 유사한 제품 권한(product jurisdiction)에 대하여 FDA 센터에 확인하여야 한다.

사용자 환경시험은 정식 시험 요약 및 정식 수용(formal acceptance) 기록을 가진 미리 정의된 문서화된 계획을 따라야 한다. 모든 시험 절차, 시험 입력 데이터 및 시험 결과의 문서화된 증거는 유지되어야 한다.

하드웨어 및 소프트웨어가 규격대로 설치 및 구성되었다는 증거가 있어야 한다. 방법(measures)은 모든 시스템 구성요소가 시험과정에서 실행되고 이 구성요소의 버젼(version)이 규격화되어 있다는 것을 명확히 하여야 한다. 시험 계획은 운영 조건의 모든 범위에 걸친 시험을 규격화하고, 일반적인 활동에서 나타나지 않은 어떤 잠재적인 결점을 파악하기 위한 노력의 과정에서 시스템이 직면하는 다양한 사건과 조건에 대하여 충분한 시간동안 지속됨을 명확히 하여야 한다.

개발자 환경에서 소프트웨어 개발자에 의하여 초기에 운영된 일부 평가는 실제 사용자 환경에서 반복되어야 한다. 이는 데이터의 고용량(high volume), 부담(load or

stress), 보완, 결점 시험(fault testing)(무효(avoidance), 발견, 허용(tolerance) 및 해결), 오류 메시지 및 안전성 요구사항의 수행을 위한 시험을 포함할 수 있다. 개발자는 이러한 목적을 위하여 사용되는 시험 데이터세트(test data set)의 일부를 사용자에게 제공할 수 있다.

목적한 기능을 적절히 수행하기 위한 시스템의 능력 평가에 더하여 정확한 인터페이스 및 시스템 사용자 능력의 평가도 이루어져야 한다. 운영자(operator)는 목적한 기능을 수행하고 모든 경보, 경고 및 오류 메시지에 대하여 적절한 방법으로 처리할 수 있어야 한다. 사용자 환경 시험과정에서 기록은 적절한 시스템 성능 및 발생된 모든 시스템 결점을 유지하여야 한다. 이 사용자 환경 시험과정에서 발견된 결점을 보정하기 위한 시스템의 개정(revision)은 다른 소프트웨어 변경을 위하여 같은 절차 및 관리를 따라야 한다.

소프트웨어 개발자는 사용자 환경시험에 참여 또는 불참할 수 있다. 개발자 참여시 설계수준(design-level) 시스템 시험의 최종 부분을 사용자 환경을 위하여 수행할 수 있다. 그렇지 않은 경우 사용자는 철저한(careful) 시험계획의 중요성, 예상 시험결과의 정의 및 모든 시험 출력의 기록을 이해하는 사람을 필요로 한다.

전형적 작업 - 사용자 환경시험

- 수용 시험 실시
- 시험결과 평가
- 오류 평가/해결
- 최종 시험 보고

5.2.7. 유지보수와 소프트웨어 변경

소프트웨어에 적용시 유지보수(maintenance)라는 용어는 하드에어에 적용한 것과 같은 의미는 아니다. 하드웨어 및 소프트웨어의 부적합/오류 메커니즘(mechanism)이 다르기 때문에 유지보수의 운영방식도 다르다. 하드웨어 유지보수는 일반적으로 예방차원의 하드웨어 유지보수 활동, 구성요소 교체(replacement) 및 시정 변경(corrective change)을 포함한다. 소프트웨어 유지보수는 시정, 향상(perfective) 및 개조(adaptive)하는 유지보수를 포함하지만 예방차원의 유지보수 활동 또는 소프트웨어 구성요소 교체를 포함하지는 않는다.

소프트웨어의 오류 및 결점을 교정하기 위하여 수행된 변경은 시정(corrective) 유지보수이다. 성능, 지속성(maintainability) 또는 다른 소프트웨어 시스템의 속성(attribute)을

향상시키기 위하여 소프트웨어에 일어난 변경은 완성적(perfective) 유지보수이다. 변경된 환경에 소프트웨어 시스템이 사용가능하도록 하기 위한 소프트웨어 변경은 적응적(adaptive) 유지보수 이다.

소프트웨어 시스템에 변경이 발생하는 경우 초기 개발 또는 배포 후 유지보수 과정에서 변경되지 않은 소프트웨어의 부분이 반대영향을 받지 않는다는 것을 증명하기 위하여 충분한 회귀분석 및 시험이 수행되어야 한다. 이는 수행된 변경의 정확성(correctness)을 평가하는 시험에 추가적인 것이다.

각 소프트웨어 변경을 위하여 필요한 특정 밸리데이션 노력은 변경의 형태, 영향 받는 개발 제품 및 소프트웨어 동작시 그 제품의 영향에 따라 결정된다. 설계 구조의 철저하고 완벽한 문서화와 다양한 모듈, 인터페이스 등의 상호관계는 변경이 발생했을 때 필요한 밸리데이션 노력을 한정할 수 있다.

변경에 대한 충분한 밸리데이션을 위하여 필요한 노력의 수준은 원본 소프트웨어의 밸리데이션이 문서화되고 수행된 정도에 따라 좌우된다. 예를 들어 추후 회귀분석을 수행하기 위하여 필요할 경우에는 시험 문서화, 시험사례 및 이전 검증·밸리데이션 시험의 결과는 수행될 필요가 있다. 추후 사용을 위하여 이 정보를 보관하는데 실패하는 것은 노력의 수준 및 변경 수행 후에 소프트웨어 유효성 재확인(revalidation) 하는 비용을 크게 증가시킬 수 있다.

표준 소프트웨어 개발 프로세스의 일부분인 소프트웨어 검증 및 밸리데이션 작업에 추가하여 다음과 같은 유지보수 작업이 개정된 소프트웨어의 밸리데이션을 지원하기 위하여 설정되어야 한다.

- **소프트웨어 밸리데이션 계획 변경(Software Validation Plan Revision)** - 이전에 밸리데이션된 소프트웨어에 대하여 현재 소프트웨어 밸리데이션 계획은 개정된 소프트웨어의 밸리데이션을 제공하기 위하여 개정되어야 한다.

- **예외사항 평가(Anomaly Evaluation)** - 소프트웨어 조직은 발견된 소프트웨어 예외사항 및 각 예외사항을 해결하기 위하여 취해진 특정 시정조치를 설명하는 소프트웨어 문제 보고서와 같은 문서화를 지속한다. 그러나 소프트웨어 개발자는 문제의 근본 원인을 결정하고 문제의 재발(recurrence) 방지를 위하여 필요한 프로세스 및 절차상의 변경을 수행하기 위한 그 다음 단계를 취하지 않아서 실수가 자주 반

복된다. 소프트웨어 예외사항은 시스템 운영 및 안전성에서 심각성(severity)과 효과의 관점에서 평가되어야 한다. 그러나 이러한 예외사항은 품질시스템에서 부적합(deficiency) 프로세스의 증상으로 취급되어서는 아니 된다. 예외사항의 근본적인 원인 분석은 특정 품질시스템 부적합을 명확히 할 수 있다. 경향이 식별되는(예 : 유사 소프트웨어 예외사항의 재발) 곳에서 적절한 시정 및 예방조치가 반드시 수행되고 유사한 품질 문제의 재발을 방지하기 위하여 문서화되어야 한다(21 CFR 820.100 참조)

- **문제점 식별 및 해결책 기록(Problem Identification and Resolution Tracking)** - 소프트웨어 유지보수과정에서 발견된 모든 문제는 반드시 문서화되어야 한다. 각 문제점의 해결책은 참고이력 및 경향에 대하여 그 문제가 수정되었음을 확실히 하기 위하여 기록되어야 한다.

- **제시된 변경 평가(Proposed Change Assessment)** - 모든 제시된 수정(modification), 향상(enhancement) 또는 추가사항(addition)은 시스템에 대하여 각 변경이 가져올 영향을 결정하기 위하여 평가되어야 한다. 이런 정보는 반복되기 위하여 필요한 검증 및/또는 밸리데이션 작업의 범위를 결정하여야 한다.
- **작업 반복(Task Iteration)** - 승인된 소프트웨어 변경에 대하여 모든 필요한 검증 및 밸리데이션 작업은 계획된 변경이 정확하게 수행되고, 모든 문서화가 완료 및 최신으로 갱신(up to date)되고, 받아들이기 어려운 변경이 소프트웨어 성능에 일어나지 않았음을 확실히 하기 위하여 실시되어야 한다.

- **문서화 갱신(Documentation Updating)** - 문서화는 문서가 변경에 따라 여향을 받았음을 결정하기 위하여 주의 깊게 검토되어야 한다. 영향 받게 되는 모든 승인된 문서(예 : 규격, 시험절차, 사용자 설명서 등)는 형상관리 유지 프로세스(configuration management process)에 따라 갱신되어야 한다. 규격은 모든 유지보수 및 소프트웨어 변경이 수행되기 전에 갱신되어야 한다.

Section 6. 자동화 공정설비의 밸리데이션과 품질시스템 소프트웨어

품질 시스템 규정은 "컴퓨터 또는 자동화 데이터 프로세싱 시스템이 생산 또는 품질 시스템의 일부로서 사용될 시 의료기기 제조업자는 설정된 프로토콜에 따라 사용목적을 위하여 컴퓨터 소프트웨어를 밸리데이션해야 한다"고 요구하고 있다(21 CFR 820.20.70(i) 참조). 이는 1978년 이후 FDA의 의료기기 GMP 규정의 법적 요구사항이다.

위의 밸리데이션 요구사항에 추가하여 의료기기 제조업자의 생산 프로세스 또는 품질

시스템(또는 모든 다른 FDA 규정에 의하여 요구되는 기록을 생성 및 유지하기 위하여 필요한)의 일부분으로서 수행되는 컴퓨터 시스템은 전자기록(Electronic Record);전자서명(Electronic Signature) 규정을 필요로 한다(21 CFR Part 11 참조). 이 규정은 기록이 전자적으로 생성 또는 관리될 때에 추가적인 보안, 데이터 통합 및 밸리데이션 요구사항을 설정한다. 이 추가적인 Part 11 요구사항은 시스템 요구사항 및 시스템을 유지하는 모든 자동화된 기록에 대한 소프트웨어 요구사항은 신중하게 고려되고 포함되어야 한다. 시스템 밸리데이션 및 소프트웨어 밸리데이션은 모든 Part 11요구사항이 충족됨을 증명하여야 한다.

컴퓨터 및 자동화 장비는 의료기기 설계, 실험실 시험 및 분석, 제품 검사 및 수용, 생산 및 프로세스 관리, 환경 관리, 포장, 라벨링, 추적성(traceability), 문서 관리, 고객 불만 관리의 모든 측면 및 품질시스템의 많은 다른 측면을 통하여 광범위하게 사용되어진다. 자동화 공장설비 운영(automated plant floor operation)은 다음과 같은 사항에서 임베디드 시스템(embedded system)의 광범위한 사용을 포함한다.

- 프로그래머블 로직 컨트롤러(PLC, programmable logic controller)
- 디지털 기능 컨트롤러(digital function controller)
- 통계적 프로세스 제어(statistical process control)
- 슈퍼바이저(supervisory, 운영체제의 중심 부분에서 하드웨어의 능력을 최대한 활용할 수 있도록 체계를 감시・제어하는 프로그램) 제어 및 데이터 획득
- 로보틱스(robotics)
- 인간-기계 인터페이스(human-machine interface)
- 입/출력기기
- 컴퓨터 운영 시스템

소프트웨어 도구는 자동화된 의료기기에 들어가는 소프트웨어를 설계, 제작(build) 및 시험하기 위하여 자주 사용된다. 워드 프로세서, 스프레드시트, 데이터베이스 및 플로 차트 소프트웨어와 같이 많은 다른 상업용 소프트웨어 적용은 품질시스템을 수행하기 위하여 사용된다. 이 모든 적용은 소프트웨어 밸리데이션을 위한 요구사항을 필요로 하지만 각 적용을 위하여 필요한 밸리데이션 접근은 매우 광범위할 수 있다.

제품 또는 품질시스템 소프트웨어가 의료기기 제조업자에 의하여 자체적으로(in-house) 개발, 계약자에 의한 개발 또는 규격품(off-the-shelf) 구입일 경우 이 지침의 다른 부분에서 개략적으로 서술된 기본적인 원칙을 이용하여 개발되어야 한다. 의료기

기 제조업자는 그 소프트웨어의 밸리데이션이 어떻게 수행되었는지를 정의하는데 있어 범위 및 융통성(flexibility)을 가지지만 밸리데이션은 어떻게, 누구에 의하여 소프트웨어가 개발될 것인지 또는 누구로부터 구매할 것인지를 결정하는데 중요한 고려사항이어야 한다. 소프트웨어 개발자는 life cycle 모델을 정의하여야 한다. 밸리데이션은 일반적으로 다음과 같은 사항에 의하여 지원된다.

- 소프트웨어 개발 life cycle의 각 단계로부터 출력의 검증
- 의료기기 제조업자의 사용 목적에서 완성된 소프트웨어의 적절한 운영에 대한 확인

6.1.밸리데이션을 입증하는 자료는 얼마나 필요한가?(HOW MUCH VALIDATION EVIDENCE IS NEEDED?)

밸리데이션 노력의 수준은 자동화 운영에 의하여 제시된 위험에 적절하여야 한다. 위험성에 추가하여 소프트웨어 프로세스의 복잡성과 의료기기 제조업자가 안전하고 효과적인 의료기기를 생산하기 위한 자동화된 프로세스에 의존하고 있는지의 정도와 같은 다른 요인들은 밸리데이션 노력의 일부분으로서 필요한 시험의 특징 및 범위를 결정한다. 자동화 프로세스의 문서화된 요구사항 및 위험 분석은 소프트웨어가 사용목적을 위하여 유효하다는 것을 증명하기 위하여 필요한 증거의 범위를 결정한다. 예를 들어 의료기기 제조업자가 운영의 출력이 배포전의 규격에 반하여 충분히 검증됨을 보여줄 수 있다면 자동화 밀링(milling) 기계는 아주 적은 시험을 요구할 수 있다. 반면에 광범위한 시험은 다음의 사항을 위하여 필요할 수 있다.

- plant-wide 전자기록 및 전자서명 시스템
- 멸균 주기를 위한 자동화 컨트롤러
- 생명유지/보조 의료기기의 완성된 회로보드의 검사 및 수용을 위하여 사용되는 자동화 시험장비

수많은 상업용 소프트웨어 적용은 품질시스템(예 : 품질시스템 추측(calculation)을 위하여 사용되는 스프레드시트 또는 통계 패키지, 경향 분석을 위하여 사용되는 그래픽 패키지, 의료기기 이력 또는 고객 불만 관리 기록을 위하여 사용되는 상업용 데이터베이스)의 일부로서 사용될 수 있다. 이러한 소프트웨어를 위하여 필요한 밸리데이션 증거의 범위는 제조업자의 문서화된 소프트웨어의 사용목적에 따른다. 예를 들어 판매업체가 제공한 소프트웨어의 모든 기능을 사용하지 않는 의료기기 제조업자는 사용될 소프트웨어 기능과 생산 또는 품질시스템의 일부로서 소프트웨어 결과에 의존하는 의료기기

제조업자를 위해서만 밸리데이션이 필요하다. 그러나 고위험도의 응용 프로그램은 비록 그 소프트웨어 기능이 사용되지 않더라도 밸리데이션되지 않은 소프트웨어 기능을 가지고 동일한 운영 환경에서 실행되어서는 아니 된다. 메모리 분할(memory partitioning) 또는 자원(resource) 보호 접근과 같은 위험 감소 기술(risk mitigation technique)은 고위험도 응용 프로그램 및 저위험도 응용프로그램이 동일 운영 환경에서 사용될 때 고려될 필요가 있을 수 있다. 소프트웨어가 갱신 또는 어떤 변경이 소프트웨어에 발생한 경우 의료기기 제조업자는 이 변경을 소프트웨어의 “사용되던 부분(used portion)"에 어떤 영향을 미치는지 고려하고 사용되는 소프트웨어의 그 부분의 유효성을 반드시 재확인 하여야 한다(21 CFR 820.70(i) 참조)

6.2. 규정된 사용자 요구사항

소프트웨어 밸리데이션을 위하여 아주 중요한 관건은 다음과 같은 사항을 정의하는 규격화된 사용자 요구사항 규격이다.

- 소프트웨어 또는 자동화 장비의 “사용 목적”
- 우수한(quality) 의료기기의 생산을 위하여 의료기기 제조업자가 소프트웨어 또는 장비에 의존하는 범위

의료기기 제조업자(사용자)는 모든 요구된 하드웨어 및 소프트웨어 형상(configuration), 소프트웨어 버전, 유틸리티(utility) 등을 포함한 예상되는 운영 환경을 정의하는 것이 필요하다. 또한 사용자는 다음과 같은 것이 필요하다.

- 시스템 성능, 품질, 오류 처리(handling), 시작(startup), 종료(shutdown), 보안 등을 위한 문서 요구사항
- 센서, 경보, 연동장치(인터락(interlock) : 진행 중인 동작이 끝날 때까지 다음 동작이 개시되지 않도록 하는 일(장치)), 논리적 프로세싱 단계 또는 명령어 배열(command sequence)과 같은 기능 또는 특징과 관련된 모든 안전성의 확인
- 수용 가능한 성능을 결정하기 위한 객관적인 기준의 정의

밸리데이션은 반드시 문서화된 프로토콜에 따라 수행되어야 하고 밸리데이션 결과는 반드시 문서화 되어야 한다(21 CFR 820.70(i) 참조). 시험사례는 미리 정의된 기준(그 대부분의 위기(critical) 변수들을 위하여)에 대하여 그 성능을 조사하기 위한 시스템을 수행할 것을 문서화하여야 한다. 시험사례는 오류 및 경보 조건, 시작, 종료, 모든 적용 가능한 사용자 기능 및 운영자 제어, 잠재적인 운영자 오류, 인정되는 값의 최대 및 최소 범위 및 장비의 사용목적을 위하여 적용될 수 있는 가혹(stress) 조건을 확인하여

야 한다. 시험사례는 실행되어야 하고 소프트웨어가 그 사용목적을 위하여 유효하다는 결론에 대하여 그 결과가 지지하는지의 여부를 결정하기 위하여 기록 및 평가되어야 한다.

의료기기 제조업자는 자체적인 인력을 이용하여 밸리데이션을 실시할 수 있고 장비/소프트웨어 판매업자 또는 컨설턴트와 같은 제3자(소프트웨어나 주변장치 등의 제조자)에 의지할 수도 있다. 어떤 경우든 의료기기 제조업자는 생산 및 품질시스템 소프트웨어가 다음과 같음을 확실히 하기 위한 최종적인 책임을 유지하여야 한다.

- 특정 사용목적을 위한 문서화된 절차에 따라 밸리데이션되는가;
- 선택된 응용 프로그램에서 의도한대로 운영할 것인가

의료기기 제조업자는 그 사용목적을 위하여 소프트웨어가 밸리데이션되었음을 객관적으로 확인할 수 있도록 다음 사항을 포함하여 문서화 하여야 한다.

- 정의된 사용자 요구사항
- 사용되는 밸리데이션 프로토콜
- 수용 기준
- 시험사례 및 결과
- 밸리데이션 요약

6.3. OTS 소프트웨어 및 자동화 설비의 밸리데이션

의료기기 제조업자들에 의해 사용된 대부분의 자동화 설비와 시스템은 제 3의 판매업체에 의해 제공되는 기성품(OTS; off-the-shelf)으로 구입되어 진다. 기기 제조업자는 OTS소프트웨어 개발자에 의해 사용되는 제품개발방법론(product development methodologies)이 기기제조업자의 의도된 사용에 적합하고 충분한지 확인을 위한 책임이 있다. OTS소프트웨어와 장비를 위해 기기제조업자는 판매업체의 소프트웨어 확인 문서에 접근할 수도 있다. 만약 판매업체가 그들의 시스템 요구사항, 확인프로세스와 확인결과에 관한 정보를 제공할 수 있다면, 의료기기 제조업자는 밸리데이션을 위한 문서화에 있어 출발점이 되는 정보를 활용할 수 있다. 검사프로토콜과 검사결과, 소스코드, 설계명세서, 요구사항정의서와 같은 판매업체의 life cycle 문서는 확인된 소프트웨어 확립에 있어 사용이 가능하다.

기기제조업자는 판매업자의 OTS소프트웨어의 구조에 있어 설계와 개발방법론 감사를 반드시 고려해야 하고 OTS소프트웨어를 위해 산출된 개발 및 확인문서를 평가하여

야 한다. 그러한 감사(audit)는 기기제조업자나 자격을 가진 제3의 기관에 의해 수행되어질 수 있다. 이 감사는 반드시 판매업체의 공정을 증명하여야 한다. 규정된 환경 내에서 운영하는 것에 익숙하지 않은 몇몇 제조업체는 기기제조업자의 확인요구사항을 제공할 수 있는 문서화된 life cycle 프로세스를 가지고 있지 않을 수도 있다. 다른 판매업체는 감사를 수락하지 않을 수도 있다.

필요한 확인정보가 판매업체로부터 가능하지 않을 경우, 기기제조업자는 소프트웨어가 “사용자 필요성과 사용의도”를 확립하기 위한 충분한 시스템단계 “black box" 검사 수행을 필요로 할 것이다. 많은 응용프로그램에 있어서의 black box 검사 하나로는 충분하지 않다. 기기생산의 위험요소여하로 만약 적합한 다른 가능한 방법이 있다면, 프로세스 내 OTS 소프트웨어의 역할, 판매업체를 감시할 수 있는 능력, 충분한 판매업체 제공 정보, 소프트웨어의 사용 또는 장비는 적합할 수도 있고 그렇지 않을 수도 있다. 기기제조업자는 또한 지속적인 유지보수와 OTS소프트웨어 제품의 제공을 위한 관련사항들을 고려해야 한다.

이를테면 소프트웨어 컴파일러, 링커, 에디터, 운영시스템 같은 몇몇의 OTS 소프트웨어 개발도구를 위하여 기기제조업자에 의한 철저한 블랙박스 검사는 실용적이 아닐 수 도 있다. 그러한 검사(확인작업의 핵심요소)가 없다면 그것은 아마도 소프트웨어 도구를 확인하기에는 불가능할 지도 모른다. 그러나 그것의 적당한 운영은 다른 의미에 의해 만족하게 추론될 수도 있다. OTS 운영 시스템은 독립된 프로그램과 같이 확인되지 않을 필요가 있다.

그러나 응용소프트웨어의 시스템단계 확인검사는 최대 로딩 조건, 파일 운영, 시스템 에러조건의 핸들링, 그리고 메모리 관리를 포함하여 반드시 사용된 모든 운영시스템 서비스를 제공하여야 한다.

제조 및 공정 소프트웨어 Validation에 대한 더 자세한 사항은 부속서 A를 참고 하기 바란다.

부록 A 참고문헌(REFERENCES)

- Design Control Guidance for Medical Device Manufacturers, Center for Devices and Radiological Health, Food and Drug Administration, March 1997.
- Do It by Design, An Introduction to Human Factors in Medical Devices, Center for Devices and Radiological Health, Food and Drug Administration, March 1997.
- Electronic Records; Electronic Signatures Final Rule, 62 Federal Register 13430 (March 20, 1997).
- Glossary of Computerized System and Software Development Terminology, Division of Field Investigations, Office of Regional Operations, Office of Regulatory Affairs, Food and Drug Administration, August 1995.
- Guidance for the Content of Pre-market Submissions for Software Contained in Medical Devices, Office of Device Evaluation, Center for Devices and Radiological Health, Food and Drug Administration, May 1998.
- Guidance for Industry, FDA Reviewers and Compliance on Off-the-Shelf Software Use in Medical Devices, Office of Device Evaluation, Center for Devices and Radiological Health, Food and Drug Administration, September 1999.
- Guideline on General Principles of Process Validation, Center for Drugs and Biologics, & Center For Devices and Radiological Health, Food and Drug Administration, May 1987.
- Medical Devices; Current Good Manufacturing Practice (CGMP) Final Rule; Quality System Regulation, 61 Federal Register 52602 (October 7, 1996).
- Reviewer Guidance for a Pre-Market Notification Submission for Blood Establishment Computer Software, Center for Biologics Evaluation and Research, Food and Drug Administration, January 1997
- Student Manual 1, Course INV545, Computer System Validation, Division of Human Resource Development, Office of Regulatory Affairs, Food and Drug Administration, 1997.
- Technical Report, Software Development Activities, Division of Field Investigations, Office of Regional Operations, Office of Regulatory Affairs, Food and Drug Administration, July 1987.

▪ **Other Government References**

- W. Richards Adrion, Martha A. Branstad, John C. Cherniavsky. NBS Special

Publication 500-75, Validation, Verification, and Testing of Computer Software, Center for Programming Science and Technology, Institute for Computer Sciences and Technology, National Bureau of Standards, U.S. Department of Commerce, February 1981.
- Martha A. Branstad, John C Cherniavsky, W. Richards Adrion, NBS Special Publication 500-56, Validation, Verification, and Testing for the Individual Programmer, Center for Programming Science and Technology, Institute for Computer Sciences and Technology, National Bureau of Standards, U.S. Department of Commerce, February 1980.
- J.L. Bryant, N.P. Wilburn, Handbook of Software Quality Assurance Techniques Applicable to the Nuclear Industry, NUREG/CR-4640, U.S. Nuclear Regulatory Commission, 1987.
- H. Hecht, et.al., Verification and Validation Guidelines for High Integrity Systems. NUREG/CR-6293. Prepared for U.S. Nuclear Regulatory Commission, 1995.
- H. Hecht, et.al., Review Guidelines on Software Languages for Use in Nuclear Power Plant Safety Systems, Final Report. NUREG/CR-6463. Prepared for U.S. Nuclear Regulatory Commission, 1996.
- J.D. Lawrence, W.L. Persons, Survey of Industry Methods for Producing Highly Reliable Software, NUREG/CR-6278, U.S. Nuclear Regulatory Commission, 1994.
- J.D. Lawrence, G.G. Preckshot, Design Factors for Safety-Critical Software, NUREG/CR-6294, U.S. Nuclear Regulatory Commission, 1994.
- Patricia B. Powell, Editor. NBS Special Publication 500-98, Planning for Software Validation, Verification, and Testing, Center for Programming Science and Technology, Institute for Computer Sciences and Technology, National Bureau of Standards, U.S. Department of Commerce, November 1982.
- Patricia B. Powell, Editor. NBS Special Publication 500-93, Software Validation, Verification, and Testing Technique and Tool Reference Guide, Center for Programming Science and Technology, Institute for Computer Sciences and Technology, National Bureau of Standards, U.S. Department of Commerce, September 1982.
- Delores R. Wallace, Roger U. Fujii, NIST Special Publication 500-165, Software Verification and Validation: Its Role in Computer Assurance and Its Relationship with Software Project Management Standards, National Computer Systems Laboratory, National Institute of Standards and Technology, U.S. Department of

Commerce, September 1995.
- Delores R. Wallace, Laura M. Ippolito, D. Richard Kuhn, NIST Special Publication 500-204, High Integrity Software, Standards and Guidelines, Computer Systems Laboratory, National Institute of Standards and Technology, U.S. Department of Commerce, September 1992.
- Delores R. Wallace, et.al. NIST Special Publication 500-234, Reference Information for the Software Verification and Validation Process. Computer Systems Laboratory, National Institute of Standards and Technology, U.S. Department of Commerce, March 1996.
- Delores R. Wallace, Editor. NIST Special Publication 500-235, Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric. Computer Systems Laboratory, National Institute of Standards and Technology, U.S. Department of Commerce, August 1996.

▪ International and National Consensus Standards

- ANSI / ANS-10.4-1987, Guidelines for the Verification and Validation of Scientific and Engineering Computer Programs for the Nuclear Industry, American National Standards Institute, 1987.
- ANSI / ASQC Standard D1160-1995, Formal Design Reviews, American Society for Quality Control, 1995.
- ANSI / UL 1998:1998, Standard for Safety for Software in Programmable Components, Underwriters Laboratories, Inc., 1998.
- AS 3563.1-1991, Software Quality Management System, Part 1: Requirements. Published by Standards Australia [Standards Association of Australia], 1 The Crescent, Homebush, NSW 2140.
- AS 3563.2-1991, Software Quality Management System, Part 2: Implementation Guide. Published by Standards Australia [Standards Association of Australia], 1 The Crescent, Homebush, NSW 2140.
- IEC 60601-1-4:1996, Medical electrical equipment, Part 1: General requirements for safety, 4. Collateral Standard: Programmable electrical medical systems. International Electrotechnical Commission, 1996.
- IEC 61506:1997, Industrial process measurement and control – Documentation of application software. International Electrotechnical Commission, 1997.
- IEC 61508:1998, Functional safety of electrical/electronic/programmable electronic

safety-related systems. International Electrotechnical Commission, 1998.
- IEEE Std 1012-1986, Software Verification and Validation Plans, Institute for Electrical and Electronics Engineers, 1986.
- IEEE Standards Collection, Software Engineering, Institute of Electrical and Electronics Engineers, Inc., 1994. ISBN 1-55937-442-X.
- ISO 8402:1994, Quality management and quality assurance – Vocabulary. International Organization for Standardization, 1994.
- ISO 9000-3:1997, Quality management and quality assurance standards - Part 3: Guidelines for the application of ISO 9001:1994 to the development, supply, installation and maintenance of computer software. International Organization for Standardization, 1997.
- ISO 9001:1994, Quality systems – Model for quality assurance in design, development, production, installation, and servicing. International Organization for Standardization, 1994.
- ISO 13485:1996, Quality systems – Medical devices – Particular requirements for the application of ISO 9001. International Organization for Standardization, 1996.
- ISO/IEC 12119:1994, Information technology – Software packages – Quality requirements and testing, Joint Technical Committee ISO/IEC JTC 1, International Organization for Standardization and International Electrotechnical Commission, 1994.
- ISO/IEC 12207:1995, Information technology – Software life cycle processes, Joint Technical Committee ISO/IEC JTC 1, Subcommittee SC 7, International Organization for Standardization and International Electrotechnical Commission, 1995.
- ISO/IEC 14598:1999, Information technology – Software product evaluation, Joint Technical Committee ISO/IEC JTC 1, Subcommittee SC 7, International Organization for Standardization and International Electrotechnical Commission, 1999.
- ISO 14971-1:1998, Medical Devices – Risk Management – Part 1: Application of Risk Analysis. International Organization for Standardization, 1998.
- Software Considerations in Airborne Systems and Equipment Certification. Special Committee 167 of RTCA. RTCA Inc., Washington, D.C. Tel: 202-833-9339. Document No. RTCA/DO-178B, December 1992.

▪ **Production Process Software References**

- The Application of the Principles of GLP to Computerized Systems, Environmental Monograph #116, Organization for Economic Cooperation and Development (OECD), 1995.
- George J. Grigonis, Jr., Edward J. Subak, Jr., and Michael Wyrick, "Validation Key Practices for Computer Systems Used in Regulated Operations," Pharmaceutical Technology, June 1997.
- Guide to Inspection of Computerized Systems in Drug Processing, Reference Materials and Training Aids for Investigators, Division of Drug Quality Compliance, Associate Director for Compliance, Office of Drugs, National Center for Drugs and Biologics, &Division of Field Investigations, Associate Director for Field Support, Executive Director of Regional Operations, Food and Drug Administration, February 1983.
- Daniel P. Olivier, "Validating Process Software", FDA Investigator Course: Medical Device Process Validation, Food and Drug Administration.
- GAMP Guide For Validation of Automated Systems in Pharmaceutical Manufacture, Version V3.0, Volume 1, Part 1: User Guide & Part 2: Supplier Guide & Volume 2: Best Practice for User and Suppliers
- Good Automated Manufacturing Practice (GAMP) Forum, March 1998: Technical Report No. 18, Validation of Computer-Related Systems. PDA Committee on Validation of Computer-Related Systems. PDA Journal of Pharmaceutical Science and Technology, Volume 49, Number 1, January-February 1995 Supplement.
- Validation Compliance Annual 1995, International Validation Forum, Inc.

▪ **General Software Quality References**

- Boris Beizer, Black Box Testing, Techniques for Functional Testing of Software and Systems, John Wiley &Sons, 1995. ISBN 0-471-12094-4.
- Boris Beizer, Software System Testing and Quality Assurance, International Thomson Computer Press, 1996. ISBN 1-85032-821-8.
- Boris Beizer, Software Testing Techniques, Second Edition, Van Nostrand Reinhold, 1990. ISBN 0-442-20672-0.
- Richard Bender, Writing Testable Requirements, Version 1.0, Bender &Associates, Inc., Larkspur, CA 94777, 1996.
- Frederick P. Brooks, Jr., The Mythical Man-Month, Essays on Software

Engineering, Addison-Wesley Longman, Anniversary Edition, 1995. ISBN 0-201-83595-9.
- Silvana Castano, et.al., Database Security, ACM Press, Addison-Wesley Publishing Company, 1995. ISBN 0-201-59375-0.
- Computerized Data Systems for Nonclinical Safety Assessment, Current Concepts and Quality Assurance, Drug Information Association, Maple Glen, PA, September 1988.
- M. S. Deutsch, Software Verification and Validation, Realistic Project Approaches, Prentice Hall, 1982.
- Robert H. Dunn and Richard S. Ullman, TQM for Computer Software, Second Edition, McGraw-Hill, Inc., 1994. ISBN 0-07-018314-7.
- Elfriede Dustin, Jeff Rashka, and John Paul, Automated Software Testing – Introduction, Management and Performance, Addison Wesley Longman, Inc., 1999. ISBN 0-201-43287-0.
- Robert G. Ebenau and Susan H. Strauss, Software Inspection Process, McGraw-Hill, 1994. ISBN 0-07-062166-7.
- Richard E. Fairley, Software Engineering Concepts, McGraw-Hill Publishing Company, 1985. ISBN 0-07-019902-7.
- Michael A. Friedman and Jeffrey M. Voas, Software Assessment - Reliability, Safety, Testability, Wiley-Interscience, John Wiley &Sons Inc., 1995. ISBN 0-471-01009-X.
- Tom Gilb, Dorothy Graham, Software Inspection, Addison-Wesley Publishing Company, 1993. ISBN 0-201-63181-4.
- Robert B. Grady, Practical Software Metrics for Project Management and Process Improvement, PTR Prentice-Hall Inc., 1992. ISBN 0-13-720384-5.
- Les Hatton, Safer C: Developing Software for High-integrity and Safety-critical Systems, McGraw-Hill Book Company, 1994. ISBN 0-07-707640-0.
- Janis V. Halvorsen, A Software Requirements Specification Document Model for the Medical Device Industry, Proceedings IEEE SOUTHEASTCON '93, Banking on Technology, April 4th -7th, 1993, Charlotte, North Carolina.
- Debra S. Herrmann, Software Safety and Reliability: Techniques, Approaches and Standards of Key Industrial Sectors, IEEE Computer Society, 1999. ISBN 0-7695-0299-7.
- Bill Hetzel, The Complete Guide to Software Testing, Second Edition, A

Wiley-QED Publication, John Wiley &Sons, Inc., 1988. ISBN 0-471-56567-9.
- Watts S. Humphrey, A Discipline for Software Engineering. Addison-Wesley Longman, 1995. ISBN 0-201-54610-8.
- Watts S. Humphrey, Managing the Software Process, Addison-Wesley Publishing Company, 1989. ISBN 0-201-18095-2.
- Capers Jones, Software Quality, Analysis and Guidelines for Success, International Thomson Computer Press, 1997. ISBN 1-85032-867-6.
- J.M. Juran, Frank M. Gryna, Quality Planning and Analysis, Third Edition, , McGraw-Hill, 1993. ISBN 0-07-033183-9.
- Stephen H. Kan, Metrics and Models in Software Quality Engineering, Addison-Wesley Publishing Company, 1995. ISBN 0-201-63339-6.
- Cem Kaner, Jack Falk, Hung Quoc Nguyen, Testing Computer Software, Second Edition, Vsn Nostrand Reinhold, 1993. ISBN 0-442-01361-2.
- Craig Kaplan, Ralph Clark, Victor Tang, Secrets of Software Quality, 40 Innovations from IBM, McGraw-Hill, 1995. ISBN 0-07-911795-3.
- Edward Kit, Software Testing in the Real World, Addison-Wesley Longman, 1995. ISBN 0-201-87756-2.
- Alan Kusinitz, "Software Validation", Current Issues in Medical Device Quality Systems, Association for the Advancement of Medical Instrumentation, 1997. ISBN 1-57020-075-0.
- Nancy G. Leveson, Safeware, System Safety and Computers, Addison-Wesley Publishing Company, 1995. ISBN 0-201-11972-2.
- Michael R. Lyu, Editor, Handbook of Software Reliability Engineering, IEEE Computer Society Press, McGraw-Hill, 1996. ISBN 0-07-039400-8.
- Steven R. Mallory, Software Development and Quality Assurance for the Healthcare Manufacturing Industries, Interpharm Press,Inc., 1994. ISBN 0-935184-58-9.
- Brian Marick, The Craft of Software Testing, Prentice Hall PTR, 1995. ISBN 0-13-177411-5.
- Steve McConnell, Rapid Development, Microsoft Press, 1996. ISBN 1-55615-900-5.
- Glenford J. Myers, The Art of Software Testing, John Wiley &Sons, 1979. ISBN 0-471-04328-1.
- Peter G. Neumann, Computer Related Risks, ACM Press/Addison-Wesley Publishing Co., 1995. ISBN 0-201-55805-X.

- Daniel Olivier, Conducting Software Audits, Auditing Software for Conformance to FDA Requirements, Computer Application Specialists, San Diego, CA, 1994.
- William Perry, Effective Methods for Software Testing, John Wiley &Sons, Inc. 1995. ISBN 0-471-06097-6.
- William E. Perry, Randall W. Rice, Surviving the Top Ten Challenges of Software Testing, Dorset House Publishing, 1997. ISBN 0-932633-38-2.
- Roger S. Pressman, Software Engineering, A Practitioner's Approach, Third Edition, McGraw-Hill Inc., 1992. ISBN 0-07-050814-3.
- Roger S. Pressman, A Manager's Guide to Software Engineering, McGraw-Hill Inc., 1993 ISBN 0-07-050820-8.
- A. P. Sage, J. D. Palmer, Software Systems Engineering, John Wiley &Sons, 1990.
- Joc Sanders, Eugene Curran, Software Quality, Addison-Wesley Publishing Co., 1994. ISBN 0-201-63198-9.
- Ken Shumate, Marilyn Keller, Software Specification and Design, A Disciplined Approach for Real-Time Systems, John Wiley &Sons, 1992. ISBN 0-471-53296-7.
- Dennis D. Smith, Designing Maintainable Software, Springer-Verlag, 1999. ISBN 0-387-98783-5.
- Ian Sommerville, Software Engineering, Third Edition, Addison Wesley Publishing Co., 1989. ISBN 0-201-17568-1.
- Karl E. Wiegers, Creating a Software Engineering Culture, Dorset House Publishing, 1996. ISBN 0-932633-33-1.
- Karl E. Wiegers, Software Inspection, Improving Quality with Software Inspections, Software Development, April 1995, pages 55-64.
- Karl E. Wiegers, Software Requirements, Microsoft Press, 1999. ISBN 0-7356-0631-5.