신뢰성응용연구
제16권, 제3호. pp.216-223, 2016

기능 안전 관점에서의 의료기기 소프트웨어 신뢰성 평가 방법에 관한 연구

김성민† · 고병각 · 도경훈 · 김혜진 · 함중걸

한국산업기술시험원

Study on Reliability Assessment for the Medical Device Software from the Viewpoint of Functional Safety

Sung Min Kim† · Byeonggak Ko · Gyeong-Hun Do · Hye Jin Kim · Jung-Keol Ham

Korea Testing Laboratory

Purpose: This paper suggests the procedure to enhance the reliability of the software of the medical device that is to cure, treat, diagnose, and prevent a disease or an abnormal health conditions.
Methods: After test requirements are classified by the software requirements specification for safety and backgrounds, reliability assessment methods are suggested.
Results: Verification and validation for function and safety can be performed whether the medical device software are implemented as intended.
Conclusion: Procedure on the static analysis, unit test, integration test, and system test are provided for the medical device software.

Keywords: Medical Device Software, Reliability Assessment, Functional Safety, Procedure

1. 서 론

1.1 배경

전자기기와 컴퓨터와 같은 기술의 발전은 실생활을 더 편안하고 안락하게 해주고 있다. 그러나 컴퓨터를 이용한 시스템에 대한 의존도가 높아지면서 시스템이 제 기능을 하지 못하거나 정상 동작을 하지 않았을 경우 심각한 불편을 초래하게 된다[1].

특히 의료와 관련된 시스템이 제 기능을 하지 못하거나 오작동을 한다면 그 결과는 실로 치명적일 것이다 시스템의 고장이 치명적 재난의 원인이 되어 사람의 생명을 위협하거나 심각한 손실을 입힐 수 있는 시스템을 안전필수 시스템(Safety-Critical System)이라고 하며, 의료기기가 그 한 예이다. 이러한 의료기기 소프트웨어의 중요성은 날로 높아져 가고 있다. 이는 기존 사람이 하던 감시 및 통제 기능이 소프트웨어에서 대신 수행되기 때문이다. 따라서 이러한 의료기기 소프트웨어 안전의 검증에 대한 논의가 어느 때 보다 시급한 실정이다[2].

지금까지 소프트웨어 오류로 발생된 대표적인 사고 사례로는 다음과 같다. 1985년부터 1987년까지 캐

† 교신저자 smkim@ktl.re.kr
2016년 8월 16일 접수; 2016년 9월 20일 수정본 접수; 2016년 9월 23일 게재 확정.

나다 AECL에서 만든 종양제거 방사선 치료기 'Therac25'가 대표적인 사례이다. 당시 소프트웨어 오류로 발생한 6건의 사고 때문에 3명이 사망하고, 3명은 심각한 방사능 후유증에 시달렸다[3].

이와 같이 안전 소프트웨어의 역할이 커지면서 정상적으로 소프트웨어가 동작하는 가, 더 나아가 안전한가를 보증할 수 있는 방안들의 제시가 요구되고 있다. 기존의 의료분야 소프트웨어 연구의 경우, 신뢰성 평가 방법인 FMEA 기법을 적용한 연구 논문이 있으나, 품질 측면에서의 신뢰성 평가 방법을 제시하였을 뿐, 기능안전성 평가 범위는 포함하고 있지 않다[4]. 또한 IEC 62304 기반으로 소프트웨어 평가 품질 평가 방법에 관한 연구 논문은 시험 요구 사항을 제시하고IEC 61508-3과 비교하여 프레임 워크 모델을 제시하였으나, 구체적인 시험 방법 및 기준은 제시하지 않았다[5]. 본 논문에서는 의료기기용 소프트웨어의 안전등급별 시험 요구 사항을 분류하였고, 그 배경을 조사하여 신뢰성 검증 및 향상을 위한 신뢰성 평가 기준 및 구체적인 방법에 관한 절차를 기능 안전 관점에서 제시하였다.

2. 이 론

2.1 기능안전 표준 체계

안전이란 위험이 생기거나 사고가 날 염려로부터의 자유를 뜻하며, 이 정의에 따라 안전의 개념은 1차적 안전, 기능안전, 간접안전으로 분리된다[6].

1차적 안전은 화재, 감전과 같은 하드웨어에 의한 직접적 사고로부터의 안전을 정의한 것이고, 기능안전은 리스크 평가 측정결과에 따라서 설계과정을 통해 위험이 제거되는 장비의 안전을 말하며, 간접 안전은 데이터베이스 정보 에러와 같이 잘못된 정보 제공으로 발생할 수 있는 원인으로부터의 안전을 정의한다[7].

따라서 위험 제거 또는 낮출 수 있는 위험 영역인 기능안전에 대해 많은 연구가 이루어지고 있으며 관련하여 다양한 분야의 표준들이 존재하고 있다[8].

ISO/IEC Guide 51은 제품 표준에 안전 관련 규정을 도입하기 위한 기본적인 가이드라인이다. 위 가이드라인의 A 표준은 광범위한 제품, 프로세스 또는 서비스에 대해 적용되는 일반적인 안전 측면에 관한 기본 개념과 원칙, 요구사항을 포함하고 있고, B 표준은 몇 개 또는 한 무리의 유사한 제품, 프로세스 또는 서비스에 적용할 수 있는 안전 측면을 포함하는 표준으로 IEC 61508 표준 등이 이에 해당한다. C 표준은 특정 또는 한 무리의 제품, 프로세스 또는 서비스의 안전 측면을 포함하는 표준으로 철도 안전 표준인 EN 50129, 의료기기 안전 표준인 IEC 60601, 자동차 기능안전 표준인 ISO 26262등이 이에 해당한다[9].

2.2 의료기기 소프트웨어 관련 표준 및 평가 방안

의료기기 소프트웨어 관련 표준은 다음과 같다. IEC 62304에서는 의료기기의 개발 및 유지보수의 개발 생명주기를 정의하고 각 단계별로 수행해야 하는 활동과 지표 및 산출물을 정의하고 있다. IEC 60601 표준에서는 의료기기에 대한 안전 요구사항을 도출해 안전성을 분석하고 있으며, ISO 14971은 의료기기 개발 시 필요한 위험분석 방법을 규정하고 소프트웨어 개발 단계에 이를 적용하고 있다.

<Fig. 1>은 PEMS(프로그래밍이 가능한 전기 의료기기, Programmable Electrical System)의 개발 과정과 현상을 설명하기 위한V 모델로서, 소프트웨어 요구사항의 설정에서부터 소프트웨어 단위(유닛)이 소프트웨어 시스템에 통합할 때가지 적용되는 흐름을 도해한 것이다. 이 흐름은 위험관리 활동(프로세스)이 수반되어야 함을 의미한다. 소프트웨어의 위해요인 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 밸리데이션의 범위, 시험방법 및 그 수준(정도)은 달라질 것이다[10].

의료기기 소프트웨어 제품 신뢰성 평가를 위한 시험방법은 IEC 62304를 기반으로 정적시험, 단위시험, 통합시험 및 시스템 시험으로 구성할 수 있다. IEC 62304에서 정의된 의료기기 소프트웨어 안전등급별 시험요구사항은<Table 1>과 같다.

제조업체는 소프트웨어 시스템에서 발생하며, 환자, 작업자 또는 기타 사람에 대한 위해의 영향에 따라 각 소프트웨어 시스템에 소프트웨어 안전등급을 할당해야 한다. 소프트웨어 안전등급은 다음과 같은 심각도에 기초해 할당된다[10].

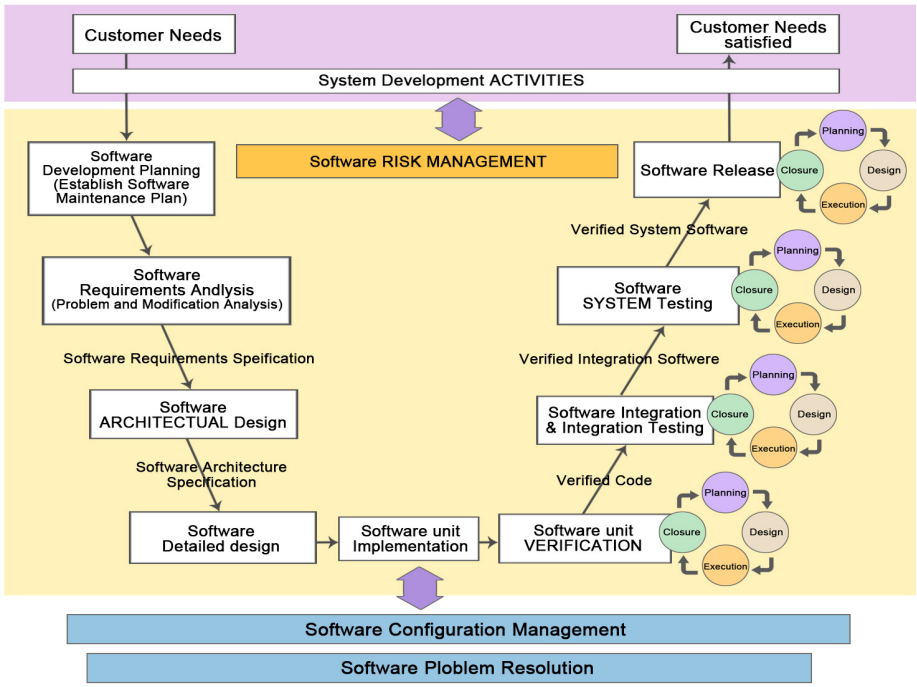

Fig. 1 Software development life cycle(V-model)

Table 1 Classification of test requirements by the software requirements specification for safety

IEC 62304	Requirements	Test type	Software safety classification		
			A	B	C
5.5.3	The manufacturer shall establish acceptance criteria for software units prior to integration into lager software items as appropriate, and ensure that software units meet acceptance criteria. Examples of acceptance criteria are: - Does the software code implement requirements including risk control measures? - Is the software code free from contradiction with the interfaces documented in the detailed design of the software unit? - Does the software code conform to programming procedure or coding standards?	Unit verification Unit verification Static Analysis		○	○
5.6.3	The manufacturer shall test the integrated software items in accordance with the integration plan and document the results.	Integration testing		○	○
5.6.4	For software integration testing, the manufacturer shall address whether the integrated software item performs as intended.	Integration testing		○	○
5.7.1	The manufacturer shall establish and perform a set of tests, expressed as input stimuli, expected outcomes, pass/fail criteria and procedure, for conducting software system testing, such that all software requirements are covered.	Software System testing		○	○

(1) 등급 A: 부상이나 건강 피해가 없다.
(2) 등급 B: 중상이 가능하지 않다.
(3) 등급 C: 사망이나 중상이 가능하다.

3. 본 론

3.1 의료기기 소프트웨어 정적시험

3.1.1 소프트웨어 정적시험의 목적 및 배경

소프트웨어 정적시험은 소프트웨어 코드의 오류 또는 오류 유발요소를 제거함으로써 소프트웨어 안전성을 향상시키는데 그 목적이 있다. 바람직한 코드 특성을 일관성 있게 달성하려면, 선호하는 코딩 스타일을 지정하기 위해 코딩표준이 사용되어야 한다. 코딩 표준에는 이해성, 언어 사용 규칙이나 제약조건 및 복잡성 관리에 대한 요구사항이 포함된다. 각 단위에 대한 코드를 검증해 지정한 코딩 표준을 준수하는지 확인해야 한다[11].

3.1.2 소프트웨어 정적시험의 기준

정적시험의 목적을 달성하기 위해 의료기기 소프트웨어 신뢰성평가 절차로 코딩 규칙 적합성 시험과 코드 복잡도 분석 시험을 채택하였다.

(1) 코딩 규칙 적합성 시험

의료기기 소프트웨어 신뢰성 평가 절차는critical system 분야에 사용되는 MISRA C:2004의 모든 의무 적용 규칙을 소프트웨어 코드에 적용할 것을 요구하였다.

MISRA C:2004는 총 121개의 의무 적용(mandatory requirements) 규칙과 20개의 권고 적용(advisory requirements)로 구성되어 있으며, 각 규칙은 정적 분석(static analysis)를 통해 기계적으로 검출 가능하거나 또는 기계적 검출이 불가능한 성격을 갖는다.

기본적으로 MISRA C:2004의 규칙은 다음을 만족하기 위한 규정들의 집합으로 이해될 수 있다.

A: 소프트웨어 코드 개발과정에서 발생할 수 있는 기본적인 실수의 방지
B: 코드 개발자에게 혼동을 야기할 가능성이 있는 코딩 스타일의 배제
C: 컴파일러 및 대상 시스템 (target hardware)에 대한 의존성의 배제

Table 2 Criteria of the cyclomatic complexity for a source code

Complexity metrics	Criteria	Software safety classification
Cyclomatic Complexity	Less than 20	Classes B and C

D: 런타임(runtime)시 오류를 야기할 수 있는 잠재적 위험성을 갖는 코드 구현의 배제
E: 컴파일러의 오인식 방지

위의 5가지 사항 중, 코드 개발 과정에서 규칙의 적용으로 인해 투입되어야 하는 노력(effort)이 제한적이고 도구에 의한 검출이 용이하며, 현장에서 기본 규칙(common sense)으로 인식되는 규칙인 A, B, C, E 항목의 경우, 소프트웨어 안전등급과는 무관하게 적용되는 것이 바람직하다. 그 외에 코드 개발에 상당한 제약으로 작용할 수 있는 D 규칙의 경우는 제외하기로 하였으며, 의료기기 소프트웨어 신뢰성평가 절차는 의무적용 규칙 중 안전등급 B는 90개, 안전등급 C는 107개 적용을 정의하였다.

(2) 소스코드 복잡도 시험

소프트웨어 코드의 복잡도의 증가와 코드의 오류 발생 확률 간에 일정 수준 이상의 상관성이 존재한다는 것은 널리 알려진 사실이며, 복잡도가 높은 코드의 경우 동적 시험(dynamic test)을 위한 시험케이스(test case)의 개발 및 시험 수행이 어렵다는 문제점이 있다. 따라서 <Table 2>의 대표적인 복잡도 지표를 통한 소프트웨어 코드의 복잡도 관리 수행은 필수적이며, 모든 함수에 대한 복잡도 분석 수행이 필요하다. Cyclomatic complexity는 1개 함수 내부의 분기문 밀도이다[12].

3.2 의료기기 소프트웨어 단위시험

3.2.1 소프트웨어 단위시험의 목적 및 배경

소프트웨어 단위시험이란, 소프트웨어를 구성하는 최소 기능단위를 검증하기 위한 시험이다. 각 단위에 대한 코드가 정확하지 못할 경우 의료기기 소프트웨어는 의도된 대로 기능을 수행하지 않기 때문에 코드를 검증하여야 한다[10]. C 언어의 경우 1개의 C 함수가 최소단위로 간주될 수 있다. 단위시험의 목적은 다음과 같다[11].

(1) 소프트웨어 유닛에 할당된 소프트웨어 요구사항(software requirements)과 해당 코드의 동작간의 일치성 검증
(2) 소프트웨어 유닛 사양(software detailed design)과 해당 코드의 동작간의 일치성 검증 등
(3) 소프트웨어 유닛 사양에 정의된 함수원형(function prototype)의 구성요소(즉, return value의 data type, argument data type)와 실제 코드간의 일치성 확인
(4) 입력 argument의 유효성 검사 코드의 정확한 동작 수행 여부 등
(5) 소프트웨어 유닛의 동작 과정에서 발생 할 수 있는 소프트웨어 및 하드웨어적 결함, 소프트웨어 코드의 의도치 않은 변경(code mutation), 하드웨어 레지스터 값의 오류 등이 발생시 오동작이 규정된 대응 메커니즘에 따라 제어되거나 또는 그 영향이 제한적임을 확인
(6) 해당 함수에 적용되는 소프트웨어 요구사항 및 소프트웨어 유닛 사양에 정의되거나 또는 예측된 자원 사용량을 기준으로 실제 코드의 자원 사용량을 확인

3.2.2 소프트웨어 단위시험의 기준

소프트웨어 단위 시험의 수행 목표를 달성하기 위해 충분한 수준의 시험이 수행되었는지 확인이 필요하다. 소스코드에 기초한 평가는 코드 실행 전에 오류를 파악할 수 있는 매우 효과적인 방법으로white-box 시험으로도 알려져 있다. 이는 소스코드, 구체적인 설계 규격과 다른 개발 문서로부터 획득된 지식에 기초하는 시험사례를 명확히 한다. 구조적인 시험은 프로그램이 실행될 경우 한 번도 수행되지 않는dead 코드를 명확히 할 수 있다. 구조적인 시험은 단위 시험으로 우선 수행되어지지만 소프트웨어 시험의 다른 수준까지 확대될 수는 없다[10]. 의료기기 신뢰성평가 절차는<Table 3>에 정의된 각 구조 커버리지의 목표값 달성을 제시하였다. 구문 커버리지는 실행된 소프트웨어 구문의 비율을 뜻하며, 분기 커버리지는 제어흐름이 가지는 분기 중 실행된 비율을 뜻한다[13].

3.3 의료기기 소프트웨어 통합시험

3.3.1 소프트웨어 통합시험의 목적 및 배경

소프트웨어 통합 시험이란, 소프트웨어를 구성하는 최소 기능단위를 단계적으로 통합하여 최종 소프트웨어를 만드는 과정에서 수행되는 시험을 의미하며, 다음 시험목적을 가진다[11].

(1) 소프트웨어 구조(software architecture)를 구성하는 소프트웨어 컴포넌트 별 기능 검증
(2) 소프트웨어 구조를 구성하는 각 소프트웨어 컴포넌트 내부의 소프트웨어 유닛들 간의 함수원형(argument들의 데이터 타입 포함) 및 리턴 값의 데이터 타입의 일치성 등
(3) 소프트웨어 컴포넌트 내부의 소프트웨어 유닛들 간의 제어 흐름(control flow) 및 데이터 흐름(data flow)의 일치성
(4) 소프트웨어가 시스템에 제공하는 각 기능별 수행 순서(sequence diagram)의 검증
(5) 특정 소프트웨어 컴포넌트 또는 소프트웨어 기능을 구현하는 소프트웨어 컴포넌트의 집합이 특정 하드웨어와 인터페이스되는 경우, 해당 하드웨어 인터페이스와의 정합성
(6) 소프트웨어 수준에서 구현된 안전기능의 정상 동작 여부
(7) 소프트웨어 컴포넌트 또는 컴포넌트의 집합이 특정 기능을 수행하는 과정에서 사용하는 하드웨어 자원의 양이 규정되거나 예측된 범위 내에서 사용됨을 확인

3.3.2 소프트웨어 통합시험의 기준

통합시험을 위해서 소프트웨어 단위를 집합 소프

Table 3 Criteria of the structural coverage for the unit testing

No.	Criteria of the structural coverage for the unit testing	Software safety classification			Acceptance criteria
		A	B	C	
1	Statement coverage		O	O	100%
2	Branch coverage		O	O	100%

Talbe 4 Criteria of the structural coverage for the integration testing

No.	Criteria of the structural coverage for the integration testing	Software safety classification			Acceptance criteria
		A	B	C	
1	Function coverage		O	O	100%
2	Call coverage		O	O	100%

트웨어 항목으로 통합하기 위한 계획이 설정되어야 한다. 통합계획에 따라 소프트웨어 통합에 관한 사항을 검증하고 기록하는데, 이 검증은 항목이 의도한 바와 같이 기능을 수행하는지에 대한 검증이 아닌, 계획에 따라 항목이 통합되었는지 검증하는 것이다. 소프트웨어 통합 시험은 소프트웨어 항목의 내부와 외부 연계성에서의 데이터 전달과 관리에 집중한다[10]. 소프트웨어 통합 시험의 수행 목표를 달성하기 위해 충분한 수준의 시험이 수행되었는지 확인이 필요하며, 의료기기 신뢰성평가 절차는<Table 4>에 정의된 각 구조 커버리지의 목표 값 달성을 제시하였다. 함수 커버리지는 설계된 총 함수 대비 실행된 소프트웨어 함수의 비율이며, 호출 커버리지는 설계된 총 함수호출 대비 실행된 소프트웨어 함수호출의 비율이다.

3.4 의료기기 소프트웨어 시스템 시험

3.4.1 소프트웨어 시스템 시험의 목적 및 배경

소프트웨어 시스템 시험이란, 모든 소프트웨어 요구사항이 최종 소프트웨어에 의도한 대로 구현이 되었는지 확인하기 위한 시험을 의미하며, 소프트웨어 요구사항 대비 구현된 소프트웨어의 기능, 성능 등의 일치성, 적합성을 확인하기 위한 목적을 가진다. 소프트웨어 시스템 시험을 통해 명세된 기능성이 존재함을 증명할 수 있으며, 이 시험은 프로그램의 기능과 성능을 소프트웨어 요구사항에 대해 빌드된 프로그램으로 검증한다. 소프트웨어 시스템 시험은 소프트웨어가 운영될 실제 환경 하에서 수행되는 것을 권고한다[11].

3.4.2 소프트웨어 시스템 시험의 기준

소프트웨어 시스템이 요구사항을 충족하는지 확인하기 위하여 소프트웨어 시스템 시험 수행을 위한 계획 및 일련의 시험 절차가 수립되고 수행되어야 한다. 시스템 시험은 규격화된 모든 기능과 소프트웨어가 신뢰할 수 있음을 입증하는 것이다. 소프트웨어 시스템시험은 통합 소프트웨어를 시험하며 의도된 운영 환경에서 모의실험, 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[10]. 시스템 시험은 요구사항 커버리지(Requirement coverage)를 활용하여 확인하며, 안전등급 B, C등급에 대하여 100%를 만족해야 한다. 시험동안 fail은 발생하면 안 된다. 요구사항 커버리지는 소프트웨어 요구사항 명세서(software requirement specification)에 정의된 요구사항 대비 실행된 소프트웨어의 비율이다. 의료기기 신뢰성평가 절차는 <Table 5>에 정의된 요구사항 커버리지의 달성 기준값을 제시하였다.

소프트웨어 시스템 시험 중 사소한 변경이라도, 변경이 발생할 경우, 문제해결에 있어서 변경의 유효성을 확인하기 위하여 시험을 반복하거나, 수정한 시험을 수행하는 등의 추가 시험을 수행한다. 소프트웨어 시스템 시험 과정에서 발견된 오류는 소프트웨어의 릴리즈 전에 등재, 분류, 검토 및 해결하여야 한다. 시험 결과의 후속 검토, 독립적 평가 및 시험을 반복할 수 있도록 시험 결과를 문서화하여야 한다. 문서화된 시험 결과에는 모든 시스템 구성요소가 시험과정에

Table 5 Test criteria for the software system testing

No.	Test criteria for software system testing	Software safety classification			Acceptance criteria
		A	B	C	
1	Requirement coverage		O	O	100%

서 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[10].

4. 실 험

4.1 실험 대상

본 논문에서 제시한 의료기기 소프트웨어 신뢰성 평가 절차 정적시험 기준에 따라서 다음 제품을 대상으로 실험을 실시하였다.

(1) 전동식 실린더형 주입펌프용 소프트웨어
(2) 안전등급: C
(3) 운영체제: Microsoft Windows 7 Home Premium K
(4) 기본환경: 호스트 환경 기반
(5) 시험기준: MISRA-C: 2004 coding rules 적합성 시험, Code metric 측정
(6) 시험도구: CodeScroll Code Inspector 3.5

4.2 실험 결과

실험 결과 복잡도는 기준이하로 만족하였으나, 부적합으로 검출된 항목이 있었으며, 해당 항목에 대하여 개선이 필요함을 보고서로 작성하였다. 상세 실험 결과는 다음과 같다.

(1) 실험 대상 파일 갯수: 6개
(2) 파일 유형: Implementation File
(3) 실험 코드의 Line 갯수: 3,967개
(4) 총 위배 규칙 건수: 41개
(5) 총 위배 검출 건수: 2,403개
(6) 함수 복잡도: 3

5. 결론 및 향후 연구과제

의료기기 소프트웨어의 개발 프로세스는 IEC 62304에 정의되어 있다. IEC 62304 중에서 소프트웨어 시험의 개발 프로세스는 정적시험, 단위시험, 통합시험, 시스템 시험으로 구분된다. 본 논문에서는 개발 프로세스 상에서 의료기기 소프트웨어 신뢰성 향상과 검증을 위한 평가 방법에 관한 절차를 제시하였으며, 아래와 같이 안전등급 B, C인 소프트웨어에 대하여 각 시험의 기준을 설정하였다.

(1) 정적시험: MISRA-C:2004 코딩 규칙 중, 안전등급 B는 90개, C는 107개 만족할 것. 함수 복잡도는 Cyclomatic Complexity 20 이하.
(2) 단위시험: Statement coverage/Branch coverage 각각 100% 만족하는 시험 수행.
(3) 통합시험: Function coverage/Call coverage 각각 100 %만족하는 시험 수행.
(4) 시스템 시험: Requirement coverage 100% 만족하는 시험 수행.

본 논문에서는 기능안전 관점에서 의료기기 소프트웨어 개발 프로세스 상에서 신뢰성 평가를 위한 기준과 절차를 제시하였으나, 의료기기 소프트웨어 특성상 하드웨어 시스템에 탑재되어 있는 Embeded 성격이 있으므로, 향후 하드웨어와의 통합을 평가하는 방법과 실험에 관한 연구가 필요할 것으로 판단된다.

References

[1] Korea Communications Agency (2013). “Technology Trend of the safety standards for Hardware and Software of the IT convergence industry”. Issues and Outlook of the Broadcasting Communication Technology, No. 19
[2] Jang, S. Y. (2015). “Discussion on the application method of standard for functional safety of road vehicles(ISO 26262) from the viewpoint of software testing for the safety software”. Journal of Korean Institute of Information Scientists and Engineers, Vol. 33, No. 7, pp. 27-32.
[3] Dailymedi (2015). “Urgency of improvement for the quality control of the medical device software”.
[4] Park, S. O. and Yang, H. S. (2007). “Reliability Evaluation Method of Software for Electronic Medical Devices”. Journal of the Korea Academia-Industrial coopertion Society, Vol. 88, No. 4, pp. 758-767.
[5] Huhn, M. and Zechner, A. (2010). “Arguing for Software Quality in an IEC 62304 Compliant Development Process”. Leveraging Applications of Formal Methods,

Verification, and Validation, 4th, Springer, Vol. 6416, pp. 296-311.

[6] Storey, N. (1996). “Safety-critical Computer Systems”. Addison-Wesley.

[7] MTL Instruments Group plc (2002). “An introduction to Functional Safety and IEC 61508”.

[8] Smith, D. J. and Simpson, K. (2004). “Functional Safety”. Elsevier Butterworth-Heinemann.

[9] Kang, H. D. (2012). “Comparison on the standards for functional safety of software from the viewpoint of software testing”. Journal of Korean Institute of Information Scientists and Engineers, Vol. 30, No. 2, pp. 62-71.

[10] KFDA. (2007). “Guidelines for Software Validation of Medical devices”. Korea Food & Drug Administration.

[11] IEC 62304. (2006). “Medical device software - Software life cycle processes”. International Electro-technical Commission.

[12] Watson, H. and McCabe, T. J. (1996). “Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric”. National Institute of Standards and Technology.

[13] ISO/IEC/IEEE 29114-4. (2015). “Software and systems engineering-software testing-Part 4: Test techniques”. International Organization For Standardization.