Journal of Korea Multimedia Society Vol. 15, No. 1, January 2012(pp. 115-130)

소프트웨어 안전성 평가를 위한 소프트웨어 고장 유형과 영향 분석에 관한 연구

김명희†, 박만곤††

요 약

오늘날 대다수의 안전필수 시스템들(Safety-Critical Systems)에는 컴퓨터, 전기 및 전자 부품이나 장치들에 소프트웨어를 칩에 내장하거나 제어용 소프트웨어 시스템이 탑재되어 구축되고 있다. 이에 따라, 컴퓨터 소프트웨어를 내장하였거나 탑재한 시스템들의 안전성을 평가하기 위한 여러 가지의 결함 분석 기법들이 제안되어져 오고 있다. 이러한 소프트웨어 결함 분석 기법들은 전통적으로는 하나의 안전필수 시스템을 분석하는데 단지 하나의 방법으로만 분석해 왔으나, 시스템의 종류와 특성이 다양해지면서 그 시스템에 가장 알맞은 결함 분석 기법이 동원되어야 함은 필수적이다. 본 연구에서는, 안전필수 시스템에서 소프트웨어의 크기가 비교적 작고, 안전성과 관련한 시스템 제어 반응 시간이 특별히 민감하지 않는 소프트웨어의 안전성을 평가하는 방법으로 결함트리 분석(FTA)과 소프트웨어 고장 유형 및 영향 분석(FMEA)을 결합한 시스템 결함 분석 방법을 제안하고 유비쿼터스 헬스케어 시스템을 이용하여 사례연구를 수행하고자 한다.

A Study on the Software Fault Modes and Effect Analysis for Software Safety Evaluation

Myong-Hee Kim†, Man-Gon Park††

ABSTRACT

These days, most of safety-critical systems, which are systems those failures or malfunction may result in death or serious injury to people, or loss or severe damage to social systems, or environmental harm, are being built of embedded software or loaded controlling software systems on computers, electrical and electronic components or devices. There are a lot kind of fault analysis methods to evaluate safety of the safety-critical systems equipped computers, electrical and electronic components or devices with software. However, the only assessment method to evaluate software safety of a safety-critical system is not enough to analysis properly on account of the various types and characteristic of software systems by progress of information technology. Therefore, this paper proposes the integrated evaluation method and carries out a case study for the software safety of safety-critical system which embedded or loaded software sizes are small and control response times are not sensitive by use of two security analysis methods which are Fault Tree Analysis (FTA) and Fault Modes and Effect Analysis (FMEA) for ubiquitous healthcare system.

Key words: Fault Tree Analysis(결함트리분석), Fault Modes and Effect Analysis(고장유형과 영향분석), Safety Evaluation(안전성 평가), Safety-Critical System(안전필수 시스템)

※ 교신저자(Corresponding Author) : 박만곤, 주소 : 부산 남구 대연3동 599-1 (608-737), 전화 : 051)629-6240, FAX : 051)628-6155, E-mail : mpark@pknu.ac.kr
접수일 : 2011년 11월 2일, 수정일 : 2011년 12월 2일
완료일 : 2011년 12월 28일

† 정회원, 부경대학교 교육대학원 전자계산교육전공 강의전담교수
(E-mail: mygold@pknu.ac.kr)
†† 종신회원, 부경대학교 IT융합응용공학과 교수
※ 이 논문은 2010학년도 부경대학교의 지원을 받아 수행된 연구임 (PK-2010-104)

1. 서 론

오늘날 컴퓨터, 전기, 전자 부품 또는 장치들이 복합적으로 포함된 안전필수 시스템(Safety-Critical System)은 시스템의 고장(Failure)이 치명적인 재난의 원인이 되어 사람의 생명과 밀접한 관계가 있거나, 심각한 손실 또는 재난을 주거나, 환경 파괴를 가능하게 하는 시스템들을 말한다. 우리 생활 주변에 활용되는 안전필수 시스템의 예를 자세히 살펴보면, 기간시설(응급 서비스 급송 시스템, 전기 발전 및 송전 설비, 화재 감시 및 통제 시스템, 회로차단 설비, 연소제어 설비, 통신 설비 등), 의료 설비(인공심장, 방사선 치료 장비, 체내 삽입형 자동제세동기, 로봇 수술 장비 등), 원자력 시설(원자력 발전 및 제어 설비 등), 주택 시설(승강기, 에스컬레이터, 전기 및 가스 제어 장비, 스프링클러 통제 장치 등), 레크리에이션 시설(유희시설, 파라슈트, 스쿠버 장비, 등산장비 등), 수송(철도 신호 제어 장치, 선박 운행 통제 시설 등), 자동차(에어백 장치, 브레이크 제어장치, 시트벨트 등), 항공시설(항공관제 및 운항 시스템, 무인항공기, 승무원 생명 지원 시스템, 무선 항법장치 등) 및 금융서비스(온라인 거래 시스템, 스마트 지불결제 시스템 등) 등이 있으며 이들 시스템들은 평소 안전성 관리를 수행하는 정형적인 방법과 장애 복구 기능 등 엄격한 규제를 요구하고 있다. 이들 대다수의 안전필수 시스템들은 복잡한 시스템(Complex System)이거나 복합 시스템 (Composite System)이며 비안전 중심 컴포넌트나 비안전 중심 시스템이 안전 중심 시스템의 일부로서 복합 형태로 통합되어 있다. 예를 들면, 네트워크 장비나 무선 센서 장비는 본질적으로 안전필수 시스템이 아니지만 응급 서비스 시스템이나 항공관제 시스템과 통합하게 되면 안전필수 시스템의 일부가 된다고 할 수 있다. 하드웨어(내장 소프트웨어 포함) 및 소프트웨어들로 구성된 시스템이 (1) 위험하거나 안전 필수 하드웨어 및 소프트웨어를 제어하는 경우, (2) 위험 통제의 한 부분으로서 안전필수 하드웨어 및 소프트웨어를 감시하는 경우, (3) 안전관련 의사결정이 이루어지는 정보를 제공하는 경우, (4) 자동 또는 수동으로 수행되는 위험한 운용의 영향을 분석하는 경우, (5) 하드웨어 및 소프트웨어의 위험 제어를 확인해야 하는 경우, (6) 기능수행으로부터 적절하게 안전필수 하드웨어 또는 소프트웨어를 보호해야 하는 경우들을 안전필수 시스템이라고 한다[1-4].

안전성 유지를 위하여 인간의 조작에 의해서 전적으로 통제되던 과거의 인간-기계 시스템(Human-Machine System)에서 각종 장치들이 컴퓨터와 반도체 기술의 발달에 힘입어 안전성 확보를 위한 많은 부분에서 내장된 소프트웨어 또는 탑재된 제어용 소프트웨어 시스템들이 감시 및 통제 기능을 수행함으로써 인간을 대신하고 있다. 이에 따라 감시 및 통제 기능을 수행하는 내장 소프트웨어 또는 제어용 탑재 소프트웨어 시스템이 정상적으로 동작을 하는지에 관심을 가지게 되었고, 안전필수 시스템에서 하드웨어 또는 소프트웨어의 결함으로 재난이 발생하기 시작하자 시스템의 안전을 보장해 주기 위한 노력으로 결함의 분석, 안정성 평가 모델의 개발, 안전성 보증 방법론 등에 대한 연구가 큰 관심이 되었다. 소프트웨어 안전성 작업들은 시스템 안전성 인력과 소프트웨어 품질 보증 인력 또는 독자적인 소프트웨어 안전성 엔지니어에 의해서 수행된다. 소프트웨어 안전성 작업들의 목표는 시스템에 통합되어진 최종 소프트웨어가 안전한가를 보증하는 것이다. 소프트웨어 안전성 작업들의 목표는 소프트웨어 개발 프로젝트 팀들에게 안전성 관련 교육, 소프트웨어 제품의 분석, 검사 입증과 기타 기법들을 통해서 수행된다[5]. 안전성이란 사고나 손실로부터 자유로운 상태라고 정의할 수 있는데, 소프트웨어 시스템의 경우 외부로부터의 잘못된 입력으로부터 시스템을 안전하게 지켜주고, 재난의 발생을 막기 위한 통제를 수행할 수 있을 것인가에 관심을 갖는 것을 말한다. 이와 관련하여 신뢰성과 안전성은 소프트웨어에 있어서는 매우 유사한 개념으로 사용되었으나 최근에는 그 개념이 분리되는 추세에 있다. 신뢰성은 특정한 환경 조건에서 준비된 기능이 특정 시간 내에 바르게 수행될 수 있는지에 대한 가능성을 말하는 것이라면, 안전성은 외부에서의 잘못된 입력으로부터 재난이 일어나지 않도록 할 수 있는 조건과, 계획된 기능이 수행될 것인지 아닌지에 대한 가능성을 말한다. 이처럼 안전성과 관련하여 많은 컴퓨터 시스템이 사용되어지고 내장된 소프트웨어의 안전성에 대하여 사람들의 관심이 높아짐으로써 개발되고 있거나, 양도되어지거나, 사용 중인 소프트웨어 시스템이 안전성을 확보하고 있다는 증명이 필요하게 되었다. 따라서 소프트웨어

시스템이 가지고 있는 안전성과 관련한 결함을 분석함으로써 소프트웨어의 안전성을 증명하거나, 발견된 결함을 제거하는 기술에 관하여 보다 심도 있는 연구가 필요하다고 판단된다. 이는 안전성 확보의 실패는 인간의 생명과도 관계가 있기 때문이다.

시스템의 안전성을 평가하기 위한 결함 분석 기법들로는 결함 위험 분석(Fault Hazard Analysis; FHA), 결함 트리 분석(Fault Tree Analysis; FTA), 오류 경향과 영향 분석(Failure Modes and Effect Analysis; FMEA), 오류경향, 효과 및 임계치 분석(Failure Modes, Effect and Criticality Analysis; FMECA), 결함 및 운영가능성 연구(Hazard and Operability Studies; HAZOP), 소프트웨어 공통 원인 고장 분석(Software Common Cause Failure Analysis; SCCFA) 등이 알려져 있으며, 하드웨어와 소프트웨어를 포함한 안전필수 시스템의 안전성에 대한 연구가 이루어져 왔다. 고전적인 안전성 평가 기법들은 전형적으로 시스템 안전성의 한 면으로만 접근을 하고 있는 데 반하여 최근에는 어느 하나의 시스템을 평가하기 위한 방법으로 결함 분석을 위한 모델 적용에 있어서 상호보완적인 역할을 할 수 있는 두 가지 이상의 분석을 결합하는 연구가 눈에 띄고 있다. James Catmur, Morris Chudleigh 및 Felix Redmill [6]의 연구 “제품의 라이프사이클 동안에 결함 분석기법의 사용: HAZOP와 FMEA 비교”에서 HAZOP는 부품 사이의 상호작용을 시험함으로써 결함을 찾는데 반하여, FMEA는 부품 자체에서 가능한 오류를 시험하는 것으로써, HAZOP는 반드시 평가팀에 의하여 수행되지만 FMEA는 대개 개별적으로 수행되고 있다. 이 두 가지 기법은 종종 혼란을 줄 수도 있지만, 서로 다른 시스템의 영역 사이의 상호작용이나 서로 다른 시스템 영역의 현존하는 소프트웨어 탑재 부품에 초점을 맞춘 것이기 때문에 이들은 상호보완적이라고 할 수가 있다. Thomas Maier[2]의 연구인 “안전성에 핵심을 둔 시스템에 있어서 내장된 소프트웨어의 안전성 설계 지원을 위한 FMEA와 FTA”에서는 FMEA를 통하여 아주 복잡한 제어 소프트웨어 내부의 가능한 모든 오류를 조직적으로 확인하고, 각각의 소프트웨어 요구사항의 모든 가능한 방법의 설정에 의하여 이러한 실패는 전환될 수 있고, 결함 트리의 구축을 통하여 완성될 수 있다. 이 방법론은 전술한 목표들을 넘어서 안전필수 시스템의 설계자들을 위하여 안전성과 관련한 피드백을 준비할 수 있게 해 준다고 설명하고 있다. 그러나 이러한 결함 분석 기법들이 모든 시스템에서 그대로 적용될 수 있는 것은 아니며, 안전성과 관련한 부품이나 시스템의 특성과 크기에 따라 가장 적절한 방법으로 시스템의 안정성이 평가되어야 할 것이다.

따라서 본 연구에서는 내장된 또는 탑재된 소프트웨어 시스템의 크기가 비교적 작으며, 시스템 제어에 있어서 실시간 반응 시간에 민감하지 않는 안전중심 시스템들(Safety-Critical Systems)의 소프트웨어 안전성 평가를 위하여 결함 트리 분석(FTA)과 소프트웨어 고장유형과 영향분석(FMEA)기법을 결합한 소프트웨어의 결함 분석 방안을 제안하고 유비쿼터스 헬스케어 시스템을 이용하여 사례연구를 수행하고자 한다.

2. 안전필수 시스템의 결함 분석 기법들

안전필수 시스템에 있어서 재난의 원인이 되는 객체의 실체나 시스템의 선천적인 속성을 잠재적으로 위험(Hazard)이라 하며, 내용 그 자체는 위험이 아니다. 오히려 결함(Fault)은 일련의 위험 조건이 결합된 것이다. 예를 들면, 물 그 자체는 위험한 것이 아니지만, 몇 몇 조건의 결합으로 익사 사고, 열탕화상, 물 오염중독, 홍수 피해 등을 유발하게 됨을 쉽게 생각할 수 있다. 따라서 주어진 환경 내에서 다른 조건들과 함께 사고를 일으키거나 손실을 초래할 수 있는 시스템 또는 객체의 상태나 일련의 조건들을 결함(Fault)이라 한다. 하드웨어(내장 소프트웨어 포함) 및 소프트웨어들로 구성된 시스템의 위험 원인은 하드웨어, 소프트웨어 및 운영인력(Operators)에서 발생하며 그 에 따른 통제도 하드웨어, 소프트웨어 및 오퍼레이터를 중심으로 수행된다[1]. 다음 표 1은 안전필수 시스템의 위험원인과 통제 행위에 대한 예이다.

우리가 사용하고 있는 대부분의 안전 필수 시스템은 사람이 감시하고 통제하는 사회-기술기반의 인간-기계 시스템(Human-Machine System)이다. 중대한 시스템은 고장(Failure)으로 인한 비용이 많이 들기 때문에 예상치 못한 상황을 사전에 관리하고 복구할 수 있는 관리자, 개발자 및 엔지니어들이 필요하다. 시스템의 성공적인 안전성 관리 프로세스는 다음 그림 1과 같은 관리자, 개발자 및 엔지니어들에

표 1. 안전필수 시스템의 위험원인과 통제 행위의 사례

원인(Cause)	통제(Control)	사 례(Examples)
하드웨어	하드웨어	압력 제거 밸브와 함께하는 압력 용기
하드웨어	소프트웨어	결함 검출과 안전하게 작동하는 기능
하드웨어	오퍼레이터	오퍼레이터가 고장 유니트를 제거하기 위해서 스위치를 연다.
소프트웨어	하드웨어	소프트웨어 제어 시스템을 동작시키지 않게 하기 위해서 안전 스위치를 센서를 통해서 직접 조정
소프트웨어	소프트웨어	하나는 다른 것을 점검하고 만약 고장이 검출된다면 조정하는 두 개의 독립 소프트웨어 프로세서들
소프트웨어	오퍼레이터	오퍼레이터가 디스플레이 화면에서 제어모수 위반사항을 눈으로 보고 작업 프로세스를 중단시킴
오퍼레이터	하드웨어	두 번의 오퍼레이터의 오류를 용인하는 점화 회로에 있는 세 개의 직렬 전기 스위치들
오퍼레이터	소프트웨어	오퍼레이터에 의해서 시작된 위험한 명령어를 소프트웨어가 확인하는 것
오퍼레이터	오퍼레이터	두 명의 승무원이 하나는 명령을 내리고 하나는 그 명령을 모니터링 하는 것

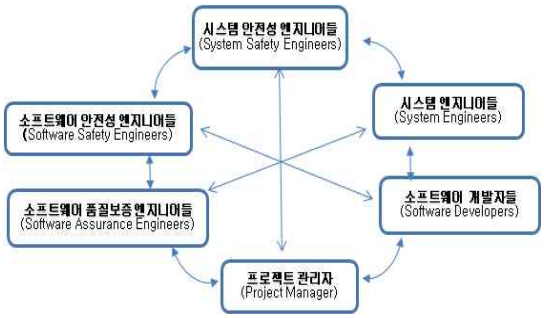

그림 1. 안전성 프로세스 관련 관리자, 개발자 및 엔지니어들

의해서 유기적으로 수행된다. 결함(Fault)은 고장(Failure)의 원인이 되며 고장이 발생하는 경우는 하드웨어의 결함 (설계상의 실수, 부품의 오류, 하드웨어의 마모 등), 소프트웨어의 결함 (분석 및 설계시의 오류, 코딩의 버그, 구현의 미비 등)과 관리자의 실수로 대부분 발생한다.

소프트웨어 안전성의 관점에서 볼 때 소프트웨어가 동작하는 시스템 내에서 입력된 정보를 분석하여 하드웨어의 동작에 이르기까지의 과정을 제어하기까지의 프로세스를 소프트웨어가 제공한다고 볼 때, 소프트웨어의 결함은 곧 재난의 발생이라고까지 확대할 수 있을 것이다. 안전성에 핵심을 둔 시스템에서 소프트웨어가 가질 수 있는 결함의 사례로는 입력 정보 오류, 입력 정보 분석의 오류 및 출력 제어 신호의 오류 등이 있다. 안전성에 핵심을 둔 소프트웨어는 안전과 관련하여 결함을 지닌 채로 가동되어서는 안 되며, 안전성에 대한 확신이 있을 때 시스템의 핵심적인 한 부분으로써 사용이 가능할 것이다.

2.1 결함 위험 분석(Fault Hazard Analysis, FHA)

결함 위험 분석(Fault Hazard Analysis, FHA)은 정성 분석으로만 사용하거나, 원하는 경우 정량으로 확장시킬 수 있는 분석의 연역적 방법입니다. FHA는 서브시스템에 대한 자세한 조사가 성분 위험 경향, 이들 위험들의 원인 및 서브시스템과 그 운영에 주는 결과적인 영향들을 결정하기 위해서 필요합니다. 이 같은 형태의 분석은 FMEA와 FMECA라고 부르는 신뢰성 분석의 한 형태입니다. FMEA / FMECA 및 FHA 사이의 주된 차이는 깊이의 문제입니다. FMEA이나 FMECA가 모든 고장들과 그들의 효과들을 조사하는 것에 대하여 FHA는 안전성에 관련된 효과들만 고려하고 있다. FHA는 반드시 고장의 “치명적인(Catastrophic)”면과 “허용범위 밖의 경향들 (Out-of-Tolerance Modes)” 양쪽 모두 고려해야 한다[6].

FHA를 수행하기 위해서는 어떤 시스템의 특성들로서 (1) 장비의 임무, (2) 운용 제한사항들, (3) 성공과 고장의 경계들, (4) 현실적인 고장 경향들과 그들의 측도, (5) 발생 확률 등에 대해서 알고 이해함이 필수적이다. FHA의 수행 순서는 다음 표 2와 같다.

그렇지만 FHA는 다음과 같은 심각한 제한사항을 갖는다.

(1) 어떤 서브시스템은 재난으로 나타나지 않는

표 2. 결함 위험 분석 (FHA)의 수행순서

단계	수행 내용	준비물 또는 산출물
1	효과적으로 취급할 수 있도록 시스템을 기능적으로 모듈들 (서브시스템들)로 나눈다.	
2	시스템과 서브시스템들을 기능 다이어그램, 약식도면, 스케치 그림들 사이의 상호관계 및 성분 부속 조립품들 사이의 상호관계를 결정하기 위해서 검토를 수행한다.	시스템 기능 블록 다이어그램(Functional Block Diagram, FBD)
3	시스템의 성분까지 분석 작업을 수행하기 위해서 각 성분의 특별한 기능을 포함하는 완전한 성분 리스트를 준비한다. 분석 작업이 기능적 또는 분할된 수준까지 수행되어지는 경우에는 이 성분 리스트를 최종 분석 수준까지 준비한다.	성분 리스트(List of Components)
4	시스템에 영향을 주는 운영 또는 환경적인 중점사항들이 시스템이나 성분들에 역효과 여부를 검토 한다.	
5	성분들에 발생하고 영향을 주는 중요한 고장 메커니즘들을 기능 다이어그램과 엔지니어링 도면으로부터 결정한다. 그러고 나서 서브시스템 고장들의 효과들도 고려한다.	중요한 고장 메커니즘들(Significant Failure Mechanism)
6	서브시스템의 다양한 고장 가능 메커니즘들을 선도할 개별 성분들의 고장 유형들을 확인한다.	개별 성분들의 고장 유형들 (Failure Modes of Individual Components)
7	하나의 성분 또는 조립품에 영향을 주는 모든 조건들이 고장이나 손상의 확률을 증가시키는 사건들로서 조작, 압력, 개인행동들에 대한 특별한 기간의 여부에 관한 표시들을 리스트 한다.	특별한 기간에 고장이나 손상을 증가시키는 조작, 압력, 개인행동 여부에 관한 조건 표시 리스트
8	위험 범주가 반드시 지정되어야 한다.	위험 범주(Risk Category)
9	위험을 제거하거나 제어할 수 있는 예방 조치(Preventive Measures)들이 리스트 되어야 한다.	위험 제어 예방 조치 리스트
10	초기 확률 값들을 넣는다. 이 값들은 최고의 판단 값이고 디자인 과정이 진행되는 동안에 수정된다. 그리고 그 확률이 평가되어질 특별한 고장 유형의 나타내고 있는가에 관심을 기울여야 한다. 단 하나의 고장률이 종종 한 종류 성분의 고장 유형 전체를 커버하도록 제공된다.	시스템과 각 성분들의 고장률(Failure Rates)
11	마지막으로 예비 임계치 분석을 수행한다.	예비 임계치 분석 결과서

고장들을 가질 것이다. 시스템 안전 프로그램(System Safety Program; SSP)에서 이 모든 고장들을 추적하면 비용이 크게 발생하고 비효율적이게 된다. 이 같은 경우의 접근에는 신뢰성 프로그램(Reliability Program)에 의해서 수행되는 FMEA를 함께 결합하는 것이 비용을 절약할 수 있다.

(2) 이러한 접근 방식은 소프트웨어 고장들을 낮게 잡고 보통 하드웨어 고장에 집중하게 된다. 그리고 종종 부적당한 배려가 인적 요인들에 의해서 주어집니다.

(3) 보통 환경적인 조건이 고려되어 지지만, 이 조건들의 발생확률은 거의 고려하지 않는다. 이 결과는 비실제적인 사건을 취급하는데 적용하게 된다.

(4) FHA에서 큰 함정중의 하나는 수학적인 분석에 있어서 정밀도를 초과하는 문제이다. 엄청 자주 안전성 분석가는 부정확한 데이터를 이용하여 정확한 수치를 얻기 위해 애쓴다. 이는 위험들을 제거하기 보다는 분석에 있어서 수치적 정밀도를 향상시키는데 너무 많은 시간을 소비하게 된다.

2.2 결함 트리 분석(Fault Tree Analysis, FTA)

FTA는 유명하고 생산적인 위험 식별 도구이다. FTA는 위험들을 평가하고 통제하기 위한 표준화된 훈련방식을 제공한다. FTA 프로세스는 안전성에서 경영 이슈들에 이르기까지 넓고 다양한 문제들을 해결하는데 종종 사용되어진다. 이 도구는 전문적인 안전성 및 신뢰성 공동체에 의해서 위험과 고장들을 예방하고 분해하는데 모두 사용되어 진다. 안전 운영에 대한 가장 까다로운 한 시스템에서 영역들을 식별하기 위해서 정성적인 방법으로도 정량적인 방법으로도 모두 사용된다. 어느 쪽의 접근이든지 효과적이

다. 결과물은 고장과 위험을 나타내는 논리적 도식 표현의 결함 트리 다이어그램이다. 결함 트리에 사용되는 기호는 다음 표 3과 같다.

안전필수 시스템을 위한 FTA는 주로 위험 요소들을 확인하는 것이 아니라 위험 요소들의 원인을 분석하기 위한 수단이다. 트리 구조의 상위 레벨에 있는 사건은 다른 기술들에 의해 예견되어지고 확인되어져야만 한다. FTA는 위험이 많은 사건을 구성할 수 있는 개별적인 오류들의 결합들을 설명하기 위하여 부울(Boolean) 논리를 사용한다. 트리의 각 단계는 그 단계의 위쪽에서 보인 문제의 원인이 될 정도로 필요하고 충분한 더 많은 기초적 사건들을 목록화해야 한다. FTA는 탑-다운 방식의 탐색 방법이다. 뒤쪽 방향 혹은 앞쪽 방향의 탐색 기술들은 시간을 넘어서서 사건들의 연대순 배열이다. 그리고 결함 트리의 각 단계는 낮은 단계로 갈수록 더 자세하게 위험 요소들을 보여준다. 중간에 있는 사건들은 가상의 사건이다. 그것들은 다만 결합체들이거나 기초적 혹은 주요 사건들의 집합들이다[7-11]. 그리고 그것들은 대체로 트리의 형식적 분석 동안에 제거되어 진다. FTA의 수행 순서는 다음 표 4와 같다.

2.3 공통 원인 고장 분석(Common Cause Failure Analysis, CCFA)

CCFA는 FTA의 한 확장 도구로서 잠재적으로 상호 종속이 될 성분 고장들이 원인되는 “연결 요인들(Coupling Factors)”을 식별하는데 사용된다. FTA로부터 최소 커트 세트(Minimal Cut Sets)의 주 사건들은 고장들이 환경, 위치, 2차 원인, 인적 오류, 또는 품질 관리에 관계되는 어떤 공통 원인에 연결되어 있는가에 대한 여부를 결정하기 위해서 행렬들의 개발을 통해서 조사된다. 하나의 커트 세트는 발생 원인이 시스템을 고장 나게 하는 성분 고장들의 집합과 같은 기본 사건들의 모임이다. 최소 커트 세트(Minimal Cut Set)는 모든 중복 결함 경로들을 제거하면서 FTA를 축소시킨다. CCFA는 FTA 사건들과 그들의 원인들 사이에 상호 종속관계를 더 잘 이해하게 해준다. CCFA는 실제 중복성(Redundancy)을 중심으로 안전성 시스템을 분석하며 성분들, 물리적 레이아웃, 오퍼레이터, 그리고 검사자의 데이터가 가용하면서 상세한 FTA 도면을 완성하고 난 후 시스템 고장들 속으로 부가적인 통찰을 제공할 수 있다[12,13]. CCFA의 수행 순서는 표 5와 같다.

2.4 고장유형 및 영향 분석(Failure Modes and Effect Analysis, FMEA)

결함 유형과 영향 분석(Failure Modes and Effect Analysis, FMEA) 기법은 신뢰성/안전성 공학자들이 설비의 신뢰성을 예견할 수 있도록 하기 위해서 개발되어졌으며 부품 그 자체에 고장 발생 원인이 개입되는 것을 피하기 위한 기법이다. FMEA는 설계, 공정, 품질보증 등 각 부문에 산재한 안전성 문제

표 3. 결함 트리의 기호

기 호	내 용
	사상(Event): 논리 게이트를 통해 사건들의 결합으로 생겨난 사건을 나타내는 기호
	기본 사상(Basic Event): 더 이상 발달을 요구하지 않는 기본적인 오류 사건을 나타내는 기호
	전개 안 된 사상(Undeveloped Event): 정보 부족에 의해 분석되지 않거나 또는 분석 필요가 없는 생략 현상을 나타내는 기호
	동일한 Fault Tree속에서 내용이 같은 다른 부분과의 사이에 전이를 나타내는 기호. 삼각형의 상부에 선이 나오는 경우는 다른 부분에서의 전입을 의미하고 측면에서 나오는 경우는 다른 부분으로의 전출을 의미
	정상적으로 발생할 것으로 기대되어질 수 있는 사건을 나타내는 기호
	게이트의 출력을 생산하는데 고려되어져야만 하는 상태를 나타내는 기호
	AND 게이트: 하위의 사건을 모두 만족하는 경우에 사용하는 논리 게이트
	OR 게이트: 하위의 사건 중 하나라도 만족하면 사용하는 논리 게이트

표 4. 결함 트리 분석 (FTA)의 수행순서

단계	수행 내용	준비물/산출물
1	시스템을 정의한다. 이것은 FTA작업의 가장 어려운 부분이다. 그것은 상위 레벨 사건을 결정하고, 초기 환경들이 존재하는 사건들 그리고 허가할 수 없는 사건들을 요구한다. 상위 레벨 사건들의 선택은 아주 중요하다. 왜냐하면 시스템에 있는 위험요소들의 평가는 만약 결함트리가 모든 중대한 상위 레벨 사건들을 위해 그려지지 않았다면 포괄적이지 못할 것이기 때문이다. 시스템의 완전한 이해와 정의, 그리고 그것의 상호관계가 있는 것이 FTA에 있는 모든 단계들을 위해 필요하다. 안전성 분석가는 기능적 다이어그램, 흐름 다이어그램, 로직 다이어그램, 설계도 등이 필요할 것이다. 특히 물리적인 시스템 경계의 정의도 매우 중요하다.	시스템 기능도 시스템 설계도
2	결함트리를 조립한다. 시스템을 정의하고 난 뒤, 다음 단계는 결함트리의 조립이다. 간단히 분석자는 먼저 개개의 시스템의 상황과 상위 레벨 사건을 가정해야 하고, 그 다음에 관계들을 설명하기 위해 <표 3>의 기호들을 사용해서 상위 레벨 사건과 관련되어진 원인이 되는 사건들과 그 사이에 있는 논리적 관계들을 기록한다.	결함 트리 다이어그램
3	결함 트리 다이어그래밍에서 톱-다운 프로세스를 모든 가지(Branch)의 뿌리 원인이 확인될 때까지 또는 더 이상 분해(Decomposition)가 필요 없을 때까지 계속한다.	
4	트리의 각 가지에 최저 레벨의 사건까지 고장 확률을 지정한다. 이 고장 확률 값은 예측, 분할 또는 과거 경험 데이터를 통해서 찾을 수가 있다.	
5	부울 논리를 사용하여 트리의 부울 방정식을 설정한다. 그리고 원하지 않는 최고 레벨의 사건에 대한 확률을 산정한다.	
6	시스템 레벨의 요구사항과 비교한다. 만약 요구사항과 일치하지 않으면 수정 작업을 수행한다. 수정 작업은 재설계에서 정밀 분석까지 진행될 수 있다.	
7	정성 분석을 수행한다. 트리가 조립되어진 후에 질적인 분석이 시작될 수 있다. 기본적으로, 정성 분석의 목적은 결함 트리들을 논리적으로 동등한 형태로 상위 레벨 사건의 원인이 될 정도로 충분한 기초 사건들의 특별한 결합들을 보여주는 것이다. 본질적으로 중간에 있는 가상의 사건들은 제거되어지고 단지 상위 레벨 사건과 초기 사건들 사이에 있는 관계들만이 설명되어질 뿐이다. 이것은 커트 세트(Cut Sets)라 불리어지는데 분석의 목표는 최소 커트 세트 (Minimal Cut Sets)를 발견하는 것이다.	
8	정량 분석을 수행한다. 논리적 게이트의 출력 가능성은 입력 사건들과 부합하는 가능성과 같다. 결함트리의 양적인 분석은 기초적 사건들의 발생 가능성으로부터 상위 레벨 사건의 발생 가능성을 계산하기 위해서 최소 커트 세트를 사용한다. 통계적으로 독립이라면 상위 레벨 사건의 가능성은 모든 커트 세트의 가능성의 합계가 될 것이다.	

표 5. 공통 원인 고장 분석 (CCFA)의 수행 순서

단계	수행 내용	준비물/산출물
1	중요 트리 그룹(Critical Tree Groups)을 설정한다. 중요 성분들이나 기능들의 해석의 범위를 제한하기 위해서 FTA나 FMECA, SCA를 사용하여 수행한다. FTA는 중요 기능들을 FMECA는 중요 성분들을 SCA(Sneak Circuit Analysis)는 감추어진 상호 관계를 확인할 수 있다.	중요트리그룹이 설정된 FTA 도면
2	단계 1에서 공통 성분들을 확인한다. 예를 들면, 이들은 아마도 공통 전력원을 함께 쓰는 중복 의료 처리 프로세서들이거나 공통 혈압 펌프에서 공급되는 중복 혈압 시스템일 것이다. 아니면 다른 시스템에 물리적으로 인접한 완전 중복 혈압 라인일 것이다.	공통성분이 확인된 FTA 도면
3	결손, 액체 누출, 하자가 있는 운영 절차 등과 같은 확신할 수 있는 고장 유형을 식별한다.	식별된 고장 유형들
4	식별된 고장 유형을 통해서 공통 원인을 확인한다. 이것은 관련된 시스템/하드웨어의 이해, “배운 교훈”의 사용 및 이전 데이터를 필요로 한다.	식별된 공통 원인들
5	시정 조치의 식별을 포함하여 분석 결과를 요약한다.	공통 원인 고장 분석 결과 요약

점을 정량적으로 관리하기 위한 기법이며, 점차 복잡화 되는 안전성 문제 발생 형태를 제품 개발 초기단계에서 사전 제거하기 위한 목적으로 활용된다[14-19].

FMEA에서의 첫 번째 단계는 모든 부품들과 그들의 오류경향들을 확인하고 리스트하는 것이다. 모든 가능한 작동모드들을 고려한 채로 각 오류경향을 위해, 다른 모든 시스템 구성요소들에 놓인 결과들은 종합적인 시스템에 놓인 결과와 함께 결정되어진다. 그러고 나서 각 오류경향의 결과에 관한 가능성과 심각성이 예측되어진다. 부품 오류 비율들은 경험에 의해 발전되어졌던 일반적인 비율로부터 예견되어지고 종종 알려진다. 정보 센터들은 그러한 정보를 수집하고 대조하고, 제조자들은 그들 자신의 제품을 위해 대체적으로 이 데이터를 가진다. FMEA는 기능적 다이어그램과 공학적인 도면작업에서 쉽게 확인될 수 있는 하드웨어 항목들을 위하여 설계가 진행될 때가 적당하다. 안전성 엔지니어들은 모형도, 기능적 다이어그램, 부품 조립 사이의 상호관계성에 대한 정보를 포함한 상세한 설계가 필요하다. FMEA는 단일 유니트 혹은 단일 고장을 분석함으로써 하나의 완전무결한 항목들을 늘리는데 효과적이다. 각각의 고장은 그것이 만들어지는 그 이후의 결과들을 제외하고 시스템 내에서 다른 고장들과 관련이 없는 채로 독립적인 방법으로 다루어진다. 분석을 단일 유니트로 한계 지음으로 인하여, 다양한 혹은 공통된 원인을 가진 고장을 고려하지 않은 채 오류 경향과 영향 분석 기법이 간단하게 적용되고, 시험은 매우 정돈되지만 만약 시간 연속성과 복잡한 시스템의 요소들 사이에서의 상호관련성이 고려되지 않는다면 제한적으로 사용될 수밖에 없을지도 모른다[20-22].

안전필수 시스템에서 설계의 불완전이나 잠재적인 결점을 찾아내기 위해 구성요소의 고장 경향과 그 상위 아이템에 대한 영향을 해석하는 기법인 FMEA에서 특히 고장 영향의 치명성(Severity or Criticality)에 대한 정도를 중요시 할 때에는 결함 유형, 영향 및 치명성 분석 (Failure Modes, Effects and Criticality Analysis, FMECA)라고 한다. 이 분석 기법은 완성된 부품이나 시스템의 안전성을 검토하기 위한 것이 아니라, 앞으로 개발하려고 하는 부품이나 시스템의 설계를 개선하여 시스템의 안전성을 높이는 데에 활용하는 것이다[23]. 초기에는 FMECA를 FMEA로 불렀다. FMECA의 “C”는 다양한 고장 효과들의 치명성(Criticality or Severity)을 나타낸다. 오늘날 FMEA가 FMECA로 종종 동일하게 사용된다. FMEA와 FMECA 사이의 구분이 모호해지고 있다. FMECA의 수행 순서는 표 6과 같다.

3. FTA와 소프트웨어 FMEA를 결합한 결함 분석 사례

안전필수 시스템을 제어하는 소프트웨어의 안전성을 평가함에 있어서 시스템이 정의되고 설계가 완성되었을 때 FTA를 통하여 소프트웨어의 결함을 분석하고, 소프트웨어 FMEA를 통하여 코드된 소프트웨어의 결함을 분석함으로써 소프트웨어의 안전성을 평가하고자 한다. 다음의 사례는 의료 서비스 시스템에서 사용된 소프트웨어의 안전성을 평가하는 모형으로써 안전필수 유비쿼터스 헬스케어 시스템을 가상으로 설계한 다음 그 소프트웨어의 안전성 평가를 위한 고장 유형, 영향 및 치명성을 분석하는 방법을 보인 것이다[24-31].

3.1 소프트웨어 고장 유형과 영향 분석 (Software FMEA)

소프트웨어 FMEA는 정의된 시스템 결함과 결함의 원인이 될 수 있는 소프트웨어의 상태 사이의 맵을 확립하는 소프트웨어 결함 분석에 기초하고 있다. 그러면 분석자들은 오류 경향을 입력변수와 각각의 소프트웨어 루틴을 위한 소프트웨어 로직에까지 발전시킨다. 소프트웨어 FMEA는 각각의 소프트웨어 루틴들에 있어서 입력변수와 소프트웨어 로직의 오류 경향이 소프트웨어 루틴들의 출력변수들에 미치는 오류의 영향을 매핑하기 위하여 분석이 수행된 다음, 소프트웨어 결함분석에 의하여 핵심적인 소프트웨어 변수의 상태가 설립되는 식으로 수행된다. 만약 어떤 오류경향 맵의 결과적인 영향들이 위험하다고 정의된 일련의 핵심적인 변수값으로 나타날 때 그 오류는 잠재적으로 하나의 결함의 원인이 되는 것이다[32-34]. 소프트웨어 FMEA를 수행하는 절차는 다음 표 7과 같다.

표 6. 결함 유형, 영향 및 치명성 분석(FMECA)의 수행 순서

단계	수행 내용	상세 내역
1	기능들을 식별해 낸다.	안전필수 시스템을 준비하고 기능을 확인한다.
2	잠재적인 고장 유형을 식별해 낸다. (일부 또는 프로세스가 고장 날 수 있는 모든 방식들)	부셔졌거나, 느슨하게 되어 있거나, 기형이 되었거나, 누출하였거나, 산화되었거나, 간과되어진 고장 유형들.
3	다른 시스템, 부속품 또는 사람에 의한 어떠한 잠재적 고장 효과라도 식별해 낸다.	시끄럽고, 불안정하고, 효력이 없고, 손상을 입고, 상해를 입고, 죽을 수 있는 잠재적 고장효과.
4	고장 효과의 치명성(Severity)에 1에서 10까지 등급을 매긴다.	Severity=1은 아무런 위험이 없음 Severity=10이면 재앙적임(Catastrophic) Severity=7~9이면 심각함(Critical) Severity=4~6이면 중함(Major, Marginal) Severity=1~3이면 심각하지 않음(Minor, Negligible)
5	고장에 대한 잠재적 원인과 메커니즘을 평가한다.	부정확한 재료, 부적당한 보수유지, 피로, 마모 등과 같은 모든 잠재적인 원인과 고장 메커니즘의 리스트를 작성한다.
6	고장 원인과 메커니즘의 발생 가능성(Occurrence)의 등급을 1에서 10까지 매긴다.	Occurrence=0은 발생 가능성이 전혀 없고, Occurrence=1~2이면 발생 가능성이 거의 없고 (Very Unlikely), Occurrence=3~4이면 발생 가능성이 조금 있고 (Remote) Occurrence=5~6이면 발생 가능성이 이따금씩 있고(Occasional) Occurrence=7~8이면 발생 가능성이 높고(Probable), Occurrence=9~10이면 발생 가능성이 아주 높음(Frequent).
7	현재의 설계 제어사항의 리스트를 작성한다.	설계의 적당함을 보증하고 고장 발생을 예방하거나 줄이기 위해서 예방 및 검출 활동의 리스트를 작성한다.
8	이들 설계 제어들을 이용하여 하나의 고장을 검출할 수 있는 능력을 1-10까지 등급을 매긴다.	고장 검출 능력 (Detection)=1은 거의 확실하고 고장 검출 능력 (Detection)=10은 절대적으로 불확실하다.
9	각각의 부품이나 단계의 위험 우선순위 (RPN; Risk Priority Number)를 계산한다.	RPN = 고장효과의 치명성(Severity) * 발생 가능성 (Occurrence) * 고장 검출 능력 (Detection).
10	추천할 개선 행동들을 설계한다.	고장효과의 치명성(Severity), 발생 가능성 (Occurrence), 및 고장 검출 능력 (Detection)의 값들을 줄일 수 있는 추가적인 행동들을 설계한다. 고장 효과의 치명성(Severity)=9 또는 10이면 특별한 주의가 요구된다.
11	설계된 개선사항들을 구현하기 위해서 의무사항과 목표 완료 일자를 지정한다.	
12	수행된 개선행동들과 RPN값으로 그 효과를 모니터링 한다.	

표 7. 소프트웨어 고장유형과 영향분석(Software FMEA) 수행 순서

단계	수행내용	상세 내역
1	소프트웨어의 위험 분석을 수행한다.	소프트웨어 FMEA는 시스템 결함이 소프트웨어와 호환 가능한 조건으로 변환될 수 있어야 하는데 이는 소프트웨어 오류 요인들의 영향이 시스템 결함으로 평가될 수 있도록 하기 위해서이다. 이것이 소프트웨어 FMEA의 첫 번째 단계이다. 각각 제어 프로세스에 정의된 시스템 결함들은 소프트웨어 변수들의 세트로 번역되어지고 변수 값과 연합하게 되는데, 시스템 수준에서 결함과 직접 관련된 변수의 집합들로 변환할 수 있다.
2	입력변수의 오류 경향을 분석한다.	입력변수 오류 경향 분석은 모든 입력변수, 변수형, 그리고 분석되어지는 각각의 루틴의 소프트웨어 로직을 위하여 일련의 오류 경향들을 성장시켜야 한다. 이러한 오류 경향들은 논리적으로 완전해야 하고, 모든 가능한 실패들이 평가되어야 한다.
3	소프트웨어 로직 오류 경향을 계산한다.	안전성 엔지니어들은 루틴의 논리적 기능을 기초로 하여 분석된 루틴의 소프트웨어 로직 오류 경향을 개선시킨다. 예를 들면, 계산 루틴의 오류 경향은 계산된 값이 크거나, 작은 결과로 나타나게 될 것이다.
4	소프트웨어 모듈들의 출력변수들에 대한 오류의 영향을 분석한다.	각각의 소프트웨어 모듈들의 출력 오류 영향은 분석되어진 루틴의 정의된 출력이나, 루틴에 의하여 접근된 전역변수 등의 변수형태에 기초를 두고 있다. 소프트웨어의 에러나 하드웨어의 고장이 있었다면, 소프트웨어 루틴에 의해 접근된 전역변수의 값은 변경되었을 것이다. 이러한 출력변수가 받는 오류의 영향을 기초로 하여 입력변수가 미치는 오류의 경향에까지 발전시킬 수 있음에 따라, 입력변수의 오류 경향과 출력변수의 영향에 대한 매트릭스를 구성함으로써 최종적으로 소프트웨어의 결함을 분석한다.

3.2 안전필수 시스템의 소프트웨어 설계 프로세스에서의 안전성 관리 활동

안전필수 시스템의 소프트웨어 개발을 위한 설계 단계는 요구사항들을 개발 단계에서 프로그램 코드로 바꾸기 위한 일련의 구성을 제공한다. 많은 요구사항들이 중복된 방법으로 구현되어져 잇을 수 있다. 디자인 활동은 이들을 한 접근 방법으로 선택하게 한다. 그리고 이 단계에서 결함 분석 기법은 완성된 부품이나 시스템의 안전성을 검토하기 위한 것이 아니라, 앞으로 개발하려고 하는 부품이나 시스템의 설계를 개선하여 시스템의 안전성을 높이는 데에 기여할 것이다. 다음 표 8은 안전성 필수 시스템을 설계하기 위한 프로세스를 나타낸다.

3.3 안전 필수 유비쿼터스 헬스케어 시스템의 기능 블록 다이어그램

어느 시스템이든 안전성 분석을 위하여 시스템 영역이 정의되어야 하는데 본 연구에서는 최상위 수준에서 유비쿼터스 헬스케어 시스템의 기능 블록 다이어그램 (Functional Block Diagram, FBD)을 다음 그림 2와 같이 표현하였다[10].

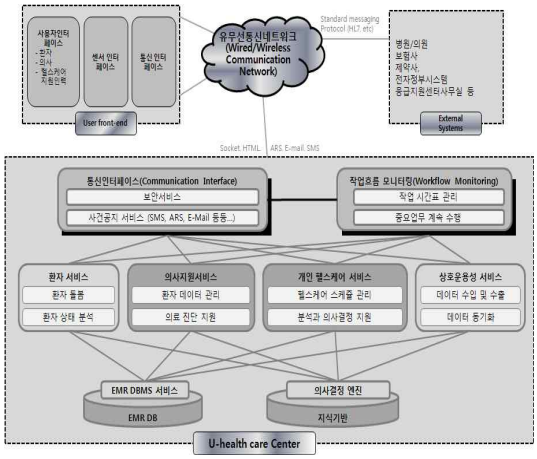

그림 2. 유비쿼터스 헬스케어 시스템의 기능 블록 다이어그램 (FBD)

유비쿼터스 헬스케어 서비스를 제공하기 위해서는 우선 서비스를 제공하는 서버가 네트워크상에 존재하고, 서비스의 수혜자 및 제공자가 유비쿼터스 단말기를 이용하여 서비스를 위한 데이터를 제공하거나 전달받게 된다. 또한 통합적인 헬스케어 서비스의 제공을 위해 관련된 외부기관들과의 자료 교환이 필요하게 된다[24,31].

표 8. 안전성 필수 시스템을 설계하기 위한 프로세스

단계	수행내용	상세 내역
1	설계 특성과 방법을 확인한다.	설계 프로세스는 소프트웨어 요구사항들을 구현하기 위해서 소프트웨어를 통해서 사용되어질 안전성 특성들로서 금지, 함정, 연동 안전장치, 부당한 주장 등을 포함하는 설계 특성 (객체/클래스 선택, 데이터 은닉, 구조적 분산 등)과 방법들을 식별한다.
2	모든 소프트웨어 요구사항들을 소프트웨어 설계로 할당한다.	설계 특성이 한 개 이상의 요구사항을 포함하고 있더라도 개개의 요구사항이 설계의 한 부분으로 구현되어 진다.
3	안전성 필수 컴퓨터 소프트웨어 성분들을 확인한다.	하나의 안전성 필수 요구사항을 구현하는 어떤 소프트웨어의 성분도 “안전성 필수”라고 표지한다.
4	실현 타당성과 기능성을 보증하기 위해서 설계 분석을 수행한다.	설계가 요구사항과 일치하는가를 검정하기 위해서 분석 작업이 가급적 일찍 수행되어져야 한다.
5	설계의 안전성 분석을 수행한다.	안전성 분석은 잠재적 위험을 식별한다. 소프트웨어 FEMA와 같은 분석 기법이 사용된다. 개개의 안전성 필수 성분은 위험 원인 여부를 검정하기 위해서 분석되어진다. 모든 성분들은 반드시 비안전성 필수 성분이 안전성 필수 성분들에게 영향을 줄 수 없다는 것을 검토 받아야 한다. 안전성 필수 성분들에 영향을 주는 데이터, 순서 매김, 타이밍 제한사항들, 기타 수단들이 관과 되어서는 안 된다.
6	소프트웨어 통합 테스트 계획을 개발하고 검토하며, 시스템 인수 테스트 계획을 갱신한다.	소프트웨어 통합 테스트 계획과 시스템 및 인수 테스트 계획을 개발하는 일은 안전성 특성들이 잘 테스트되어 졌는가를 알게 하는 중요한 절차이다.

3.3.1 User Front-end

사용자측에서는 환자용 클라이언트, 의사용 클라이언트, 개인 건강관리 클라이언트 응용모듈을 통해서, 사용자로부터 또는 사용자에 대한 정보를 수집하여 서버 측의 해당 서비스에 정보를 전달하고, 결과 또는 서비스로부터의 메시지 또는 관리 명령을 전달받게 된다.

3.3.2 u-Healthcare Center

실제 유비쿼터스 헬스케어 서비스의 중심이 되는 서버로서 제공되는 서비스는 크게 환자에 대한 서비스(patient service), 의사에 대한 서비스(physician assistant service), 개인 건강관리 서비스(personal healthcare service), 외부시스템과의 정보공유 및 교환에 관련된 서비스로서 상호운용성(interoperability)로 구분된다. 환자관리 서비스에서는 환자를 지속적으로 추적 관리하는 기능과 환자의 질환 상태를 분석하는 기능을 포함한다. 의사지원 서비스에서는 해당 의사가 주치의인 환자들에 대한 정보를 관리하는 기능과 진료지원 정보를 효과적으로 제공하는 기능을 포함하고, 개인 건강관리 서비스에서는 개인 건강관리 관련 스케쥴 관리 기능과 건강상태의 모니터링을 통한 분석 및 상태 판단 기능을 포함한다. 상호공동작업 서비스에서는 외부의 시스템과의 자료 공유와 교환을 위한 기능을 포함한다.

3.3.3 External Systems

외부의 헬스케어와 관련된 기관들(병원, 보험회사, 제약회사, 전자정부 시스템, 응급센터 등)과의 신속하고 원할한 정보 교환 및 공유를 위해, 의료정보 교환 표준인 HL7(Health Level Seven)을 비롯한 PACS, OCS, EMR 시스템과의 연동을 위한 서비스를 제공한다.

3.4 유비쿼터스 헬스케어 시스템의 기능에 따른 고장 유형들

고장 유형과 영향 분석을 위해서 다음으로는 우선 적당한 워크시트가 결정되어야한다. 많은 경우에는 고객들이 보수유지 관리 시스템에 적합 시키기 위해서 이 워크시트를 요구하게 될 것이다. 아래의 표 9는 고장 유형과 영향분석을 위한 워크시트의 한 표본이다[16].

표 9의 워크시트를 사용하여 유비쿼터스 헬스케어 시스템에서 발생할 수 있는 주요 고장들과 그 유형들을 기능에 따른 서브시스템들인 User Front-end, u-Healthcare Center 및 External Systems 별로 표 10에 정리하였다.

3.5 FTA에 의한 고장의 표현

유비쿼터스 헬스케어 시스템의 고장유형의 분석 결과를 가지고 결함 트리 분석(FTA)기법에 의한 “응급환자의 사망사고 발생”이라는 최상위 위험수준에 대해서 결함 트리로 표현하면 그림 3과 같다.

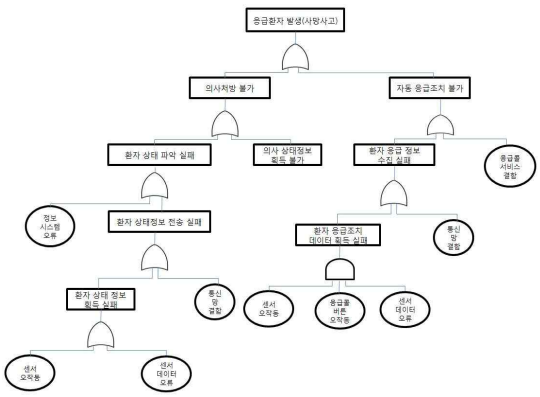

그림 3. 최상위 위험수준 “응급환자의 사망사고 발생”에 대한 결함 트리 표현

표 9. 고장 유형과 영향분석을 위한 워크시트

시스템 명						담당자				페이지	
참조 도면번호						작성일자				/	
유니트에 대한 기술			고장에 대한 기술			고장 효과		고장률 등급 (1-10)	치명성 등급 (1-10)	위험감소 측도	비고
참조 번호	기능	운영 유형	고장 유형	고장원인/ 메커니즘	고장 검출	서브시스템에서	시스템 기능에서				

표 10. 유비쿼터스 헬스케어 시스템에서의 고장유형들

서브시스템	기 능	고 장 들	고장 유형
User Front-end	센서 데이터 획득	센서 오작동	Catastrophic
		센서 데이터 오류	Critical
	통신	통신망 두절	Catastrophic
		노이즈로 인한 혼선	Critical
	프라이버시 및 보안	개인정보 노출	Critical
	사용자 인터페이스	데이터 확인 불가	Major
		데이터 입력 불가	Critical
		데이터 송신 불가	Critical
	사용자식별 (RFID)	사용자 식별 불가	Minor
	위치추적 (GPS)	위치추적 불가	Major
u-Healthcare Center	사용자 관리	사용자 관리(등록, 삭제, 수정) 불가	Major
		접근 권한 및 인증 오류	Major
	통신 인터페이스	클라이언트 접근 불가	Critical
		서버 서비스로부터 외부에 메시지 전달불가	Critical
	통지	서비스 구성원에게 메시지 전달 불가	Critical
	의사결정지원	지식베이스 DB 오류	Major
		전문가시스템 엔진 고장	Major
	데이터베이스 관리	데이터베이스 관리 불가	Major
		데이터베이스 검색 불가	Major
	데이터 교환	서버와 서비스간 데이터 교환 불가	Major
		서버와 외부 시스템과 교환 및 공유 불가	Major
	프라이버시 및 보안	정보보호기능 상실	Major
		프라이버시 정책관리 오류	Major
	스케쥴 관리	건강관리 스케쥴 컴포넌트 오류	Major
	상황인지	클라이언트의 현재 상황 판단 불가	Critical
	온톨로지 및 지식베이스	서비스 환경 및 영역에 대한 온톨로지와 지식을 축적하여 관리	Major
	데이터 입출력	데이터 입력오류	Critical
		데이터 출력오류	Critical
	패턴인식	입력신호 및 데이터로부터 정보 추출 불가	Critical
External Systems	사용자 관리	사용자 관리(등록, 삭제, 수정) 불가	Major
		접근 권한 및 인증 오류	Major
	통신 인터페이스	클라이언트 접근 불가	Critical
		서버 서비스로부터 외부에 메시지 전달불가	Critical
	통지	서비스 구성원에게 메시지 전달 불가	Critical
	데이터베이스 관리	데이터베이스 관리 불가	Major
		데이터베이스 검색 불가	Major
	데이터 교환	서버와 서비스간 데이터 교환 불가	Major
		서버와 외부 시스템과 교환 및 공유 불가	Major
	데이터 입출력	데이터 입력오류	Critical
		데이터 출력오류	Critical

3.6 U-healthcare 시스템의 고장에 대한 리스크 메트릭스

고장 유형에 관한 리스크는 고장유형의 빈도수와 고장 유형의 잠재적인 최종 효과인 치명성과의 함수로 나타나다. 고장들이 출력변수에 미치는 영향의 수준과 빈도수가결합하여 매트릭스를 구성한다. 소프트웨어 결함 분석에 있어서 입력데이터의 오류에 따른 오류 경향과 출력변수에 미치는 영향은 입력 데이터의 허용 범위 초과가 원인이 되어 출력변수에 영향을 주게 되고 소프트웨어 결함에 이르는 경우가 있으며, 소프트웨어 로직이 가진 오류가 원인이 되어 소프트웨어 결함에 이르는 경우가 있다. 안전 필수 시스템인 U-healthcare 시스템의 고장에 대한 FMEA 기법을 사용한 리스크 매트릭스를 구성한 표는 다음

표 11. 안전 필수 시스템의 고장에 대한 리스크 매트릭스

발생 가능성 (Occurrence) / 치명성(Severity)	거의 없음 (Very Unlikely)	조금 있음 (Remote)	이따금씩 있음 (Occasional)	높음 (Probable)	아주 높음 (Frequent)
재앙적임 (Catastrophic)	(Serious) Acceptable	(High) Not Acceptable	(High) Not Acceptable	(High) Not Acceptable	(High) Not Acceptable
심각함 (Critical)	(Low) Acceptable	(Serious) Acceptable	(Serious) Acceptable	(High) Not Acceptable	(High) Not Acceptable
중함 (Major, Marginal)	(Low) Acceptable	(Low) Acceptable	(Serious) Acceptable	(Serious) Acceptable	(High) Not Acceptable
심각하지 않음 (Minor, Negligible)	(Low) Acceptable	(Low) Acceptable	(Low) Acceptable	(Serious) Acceptable	(Serious) Acceptable

표 11과 같다.

리스크를 줄이기 위해서는 리스크 메트릭스 항목에서 리스크가 낮고(Low) 수용 가능한(Acceptable) 경우에는 ALARP (As Low As Reasonably Practicable, 합리적으로 실행 가능한 만큼 낮은 수준) 행동 절차가 요구되어지고, 리스크가 심각하고(Serious) 수용 가능한(Acceptable) 경우에는 ALARP 원칙을 사용하고 추가 조사를 고려해야한다. 또한 리스크가 높고(High) 수용 불가한(Not Acceptable) 경우에는 리스크를 감소시킬 조치가 강구되어야 한다.

3.7 시스템의 고장 위험에 대한 우선순위(RPN)의 계산과 시정조치

안전 필수 시스템이나 그 서브시스템들의 고장에 대한 위험 우선순위(Risk Priority Number, RPN)를 계산하여 RPN이 높은 고장부터 시정 조치 (Corrective Action)를 수행한다. RPN의 계산식은 RPN = Severity × Occurrence × Detection이며, Severity는 고장 유형의 치명성의 등급(1～10), Occurrence는 고장 유형의 발생 가능성의 등급(1～10), Detection은 시스템이 사용자에게 양도하기 이전에 고장이 검출될 수 있는 가능성의 등급(1～10)을 말한다. 리스크에 대한 시정 조치(Corrective Action)는 설계 변경, 계획된 안전성 특성들, 안전성 장치들, 경고 장치들, 안전성 관련 절차나 훈련에 의해서 리스크는 줄어들 수 있다. 그리고 시정 조치이후에 RPN을 다시 계산하여 안전성 개선사항을 확인할 수 있다. 다음 표 12는 리스크가 감소되었음을 확인할 수 있는 수정 계산된 RPN 값이다.

표 12. 수정 계산된 RPN 계산 값

RPN 모수 / 계산값	Severity	Occurrence	Detection	RPN
초기	8	7	5	280
수정 후	8	5	4	160
절감비율				43%

4. 결 론

하드웨어(내장 소프트웨어 포함) 및 소프트웨어들로 구성되어서 오퍼레이터들에 의해서 운영되는 대다수의 안전필수 시스템들은 복잡한 시스템(Complex System)이거나 복합 시스템(Composite System)이며 비안전 중심 컴포넌트나 시스템이 안전 중심 시스템의 일부로서 복합 형태로 통합되어 있다. 이들 시스템들은 주로 위험하거나 안전 필수 하드웨어 및 소프트웨어를 제어하는 경우, 위험 통제의 한 부분으로서 안전필수 하드웨어 및 소프트웨어를 감시하는 경우, 안전관련 의사결정이 이루어지는 정보를 제공하는 경우, 자동 또는 수동으로 수행되는 위험한 운용의 영향을 분석하는 경우, 하드웨어 및 소프트웨어의 위험 제어를 확인해야 하는 경우, 그리고 기능수행으로부터 적절하게 안전필수 하드웨어 또는 소프트웨어를 보호해야 하는 경우들에서 사용되어진다. 한편으로 안전성이 필수인 시스템에서 하드웨어와 소프트웨어 그리고 운용인력들에 의한 시스템의 안전성을 평가하기 위하여 많은 결함 분석 기법과 고장의 영향을 분석하는 기법들이 개발되어져 왔다. 안전 필수 시스템에서 소프트웨어의 안전성은 어

느 하나의 기법만으로는 완벽하게 판단할 수가 없다.

본 연구에서는 안전 필수 시스템에서 소프트웨어의 안전성을 보다 엄격하게 평가할 수 있는 방안의 하나로 상호보완적인 관계에 있는 결함 분석 기법을 이용하여 안전성을 분석하는 방법으로 FTA와 소프트웨어 FMEA 분석기법을 결합한 방법을 제안하였다. 이러한 상호보완적인 결함 분석이 필요한 이유는 현존하는 어느 결함 분석 기법이든 그 기법이 모든 형태의 안전필수 시스템의 소프트웨어에 공통으로 적용될 수 없기 때문이다. 그리고 인간의 삶의 질을 한 단계 높여줄 유비쿼터스 헬스케어 시스템은 인간의 생명과 직결되는 만큼 기술의 효용성 보다 그 안정성이 더욱 중요시 되어야 한다. 특히, 유비쿼터스 헬스케어에 시스템에서 인간을 대신하여 감지되고 전송・처리되어 지는 의료정보에 대한 안정성이 반드시 보장되어 한다. 따라서 본 연구에서는 안전 필수 시스템의 안정성 평가를 위해서 FTA 및 FMEA를 통합한 고장유형과 영향분석을 수행하는 사례로 유비쿼터스 헬스케어 시스템을 사용하였다.

참 고 문 헌

[1] Harold W. Lawson, “An Assessment Methodology for Safety Critical Computer Based Systems,” *Proceedings of CSR* 12th *Annual Workshop on Safety and Reliability of Software Based Systems*, pp. 183-200, 1995.

[2] Maier T., “FMEA and FTA to Support Safety Design of Embedded Software in Safety-Critical Systems,” *Proceedings of CSR* 12th *Annual Workshop on Safety and Reliability of Software Based Systems*, pp. 351-367, 1997.

[3] Peter L. Goddard, “A Combines Analysis Approach To Assessing Requirements for Safety Critical Real-Time Control Systems,” *Proceeding Annual Reliability & Maintainability Symposium*, pp. 227-230, 1993.

[4] Swarup, M. Ben and P. Seetha Ramaiah, “A Software Safety Model for Safety Critical Application,” *Proceedings of International Forum on Computer Science-Technology and Applications*, pp. 21-32, 2009.

[5] Nancy G. Leveson, *SAFEWARE: System Safety and Computers*, Addison-Wesley Publishing Company Inc, 1995.

[6] James Catmur, Morris Chudleigh, and Felix Redmill, “Use of Hazard Analysis Techniques During the Product Life Cycle: HAZOP and FMEA Compared,” *Proceedings of CSR* 12th *Annual Workshop on Safety and Reliability of Software Based Systems*, pp. 368-377, 1995.

[7] Knight, John C. and Luis G. Nakano, *Software Test Techniques for System Fault-Tree Analysis*, Press of University of York, UK, 1997.

[8] Stephen S. Cha, Nancy G. Leveson, and Timothy J. Shimeall, “Safety Verification in MURPHY using Fault Tree Analysis,” *Proceeding on the 10th International Conference on Software Engineering*, pp. 377-386, 1988.

[9] Yang H., Wang H.X., Han R.F., and Li Juan, “Application of Fault Tree in Software Safety Analysis,” *Proceedings of International Forum on Computer Science-Technology and Applications*, pp. 207-208, 2009.

[10] Younju Oh, Junbeom Yoo, Sungdeok Cha, and Han Seong Son, “Software Safety Analysis of Function Block Diagram using Fault Trees,” *Reliability Engineering and System Safety*, Vol.88(3), pp. 215-228, 2005.

[11] Hye-Jung Jung, “The Analysis of Software Fault and Application Method of Weight using the Testing Data,” *Journal of Korea Multimedia Society*, Vol.14, No.6, pp. 766-774, 2011.

[12] Rogerio De Lemos, Amer Saeed, and Tom Anderson, “Analyzing Safety Requirements for Process Control Systems,” *IEEE Software*, Vol.12, No.3, pp. 42-53, 1995.

[13] Wilson, S.P., T.P. Kelly, and J.A. McDermid, “Safety Case Development: Current Practice, Future Prospects,” *Proceedings of CSR* 12th *Annual Workshop on Safety and Reliability of Software Based Systems*, pp. 351-367, 1997.

[14] MIL-STD 1629, “Procedures for Performing a Failure Mode and Effect Analysis,” 1980.

[15] IEC 60812, “Procedures for Failure Mode and Effect Analysis (FMEA),” 2006.

[16] BS 5760-5, “Guide to Failure Modes, Effects and Criticality Analysis (FMEA and FMECA),” 1991.

[17] SAE ARP 5580, “Recommended Failure Modes and Effects Analysis (FMEA) Practices for Non-Automobile Applications,” 2001.

[18] SAE J1739, “Potential Failure Mode and Effects Analysis in Design (Design FMEA) and Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes Process FMEA) and Effects Analysis for Machinery (Machinery FMEA),” 2002.

[19] SEMATECH, “Failure Modes and Effects Analysis (FMEA): A Guide for Continuous Improvement for the Semiconductor Equipment Industry,” 1992.

[20] N. Snooke and C. Price, “Model-driven automated software FMEA,” *Proceedings of Reliability and Maintainability Symposium*, pp. 1-6, 2011.

[21] Rodrigo de Queiroz Souza and Alberto José Álvares, “FMEA and FTA Analysis for Application of the Reliability Centered Maintenance Methodology: Case Study on Hydraulic Turbines,” *ABCM Symposium Series in Mechatronic*, Vol.3, pp. 803-812, 2008.

[22] Lian-yu Zheng, Kwai-sang Chin, and Li Wei, “Knowledge-Enriched Process FMEA Model for Process Planning,” *The Asian Journal on Quality*, Vol.3, No.1, pp. 12-27, 2002.

[23] Peter L. Goddard, “Validating The Safety of Embedded Real-Time Control Systems using FMEA,” *Proceedings Annual Reliability and Maintainability Symposium*, pp.227-230, 1993.

[24] Yang, W.S., S.H. Lee, G.M. Lee, W.J. Kim, and S.J. Yoon, “Design of Ubiquitous Healthcare Service Development Framework for Establishment of Ubiquitous Hospitals,” *Proceedings of KFIS Autumn Conference*, Vol.16, No.2, pp. 57-60, 2006.

[25] Alberico, D., J. Bozarth, M. Brown, J. Gill, S. Mattern, and A. McKinlay, *Software System Safety Handbook*, Joint Services Software Safety Committee of the Joint Services System Safety Panel and the Electronic Industries Association, G-48 Committee, USA, 1999.

[26] Mark D. Hansen and Ronald L. Watts, “Software System Safety and Reliability,” *Proceedings Annual Reliability and Maintainability Symposium*, pp. 214-217, 1988.

[27] Nancy G. Leveson and Peter R. Harvey, “Analyzing Software Safety,” *IEEE Transactions on Software Engineering*, Vol.SE-9, No.5, pp. 569-579, 1983.

[28] Nancy G. Leveson, “Software Safety: Why, What, and How,” *ACM Computing Surveys*, Vol.18, No.2, pp. 125-163, 1986.

[29] Samuel J. Keene, “Assuring Software Safety,” *Proceedings Annual Reliability and Maintainability Symposium*, pp. 274-279, 1992.

[30] Stephen S. Cha, “Management Aspect of Software Safety,” *Proceeding on International Conference on Computer Assurance*, pp. 35-40, 1993.

[31] u-Policy Information Development Team, “Ubiquitous Society, New Hopes and Challenges,” *National Information Society Association*, pp. 225-240, 2006.

[32] Zhang Hong and Liu Binbin, “Integrated Analysis of Software FMEA and FTA,” *Proceedings of International Conference on Information Technology and Computer Science*, pp. 184-187, 2009.

[33] Tariq Mahmood and Man-Gon Park, “Software Performance Assessment Using Goal-Question-Metrics Approach,” *Journal of Korea Multimedia Society*, Vol.11, No.6, pp. 891-902, 2008.

[34] Peter L. Goddard and R. Davis, “Automated FMEA Techniques,” *Final Technical Report*, RADC-TR-84-244, AD-154161, 1984.

김 명 희

동서대학교 정보통신공학과 (공학사)
부경대학교 대학원 전자계산학과 (이학석사)
부경대학교 대학원 정보시스템학과 (공학박사)

2004년～2007년 정부간 국제기구 CPSC (콜롬보플랜기술교육대학) Assistant Faculty, 정보기술 및 통신학처장
2011년～현재 부경대학교 교육대학원 전자계산교육전공 강의전담교수
관심분야 : 소프트웨어 공학 및 재공학, 멀티미디어정보처리기술, 네트워크성능평가, e-Learning and u-Learning

박 만 곤

경북대학교 수학교육(이학사)
경북대학교 수학교육(교육학석사)
경북대학교 전산통계학(이학박사)
Philippine Women's University (국제행정학석사)
University of Rizal System, Philippines (명예 기술학박사)

Dept. of Electrical & Computer Engineering, University of Kansas (Post Doc.)
1981년～현재 부경대학교 IT융합응용공학과 교수
2008년～현재 한국멀티미디어학회(KMMS) 회장 및 명예회장
2002년～2007년 정부간 국제기구 CPSC (콜롬보플랜기술교육대학) 총재 (Director General and CEO)
2004년～2007년 Asia Pacific Accreditation and Certification Commission 아태지역 인증 및 검증위원회 위원장
2005년～2007년 유네스코 (UNESCO-UNEVOC) 자문위원, 아시아개발은행 자문관
관심분야 : 소프트웨어신뢰성공학, 비즈니스 프로세스 재공학 (BPR) , 소프트웨어 공학 및 재공학, 멀티미디어정보처리기술, 정보시스템성능평가, ICT-based HRD System