# 소프트웨어 라이프사이클 동안의 안전성 활동

o
성경배 · 박만곤
부경대학교 전자계산학과

# Safety Activities on The Software Life-cycle

o
Kyung-Bae Sung · Man-Gon Park
Department of computer science, Pu-Kyong National University

- 요 약 -

컴퓨터 시스템들이 인간 생활의 많은 부분에서 사용되어지면서 하드웨어 또는 소프트웨어 시스템의 안전성 문제에 대한 관심이 커지고 있다. 소프트웨어 시스템에 있어서 각각의 소프트웨어 개발 단계가 가지는 특성이 다르므로 안전성 활동들도 그 활동 관점이나 방법이 달라져야 할 것으로 판단된다.

소프트웨어의 안전성 평가의 방법론, 안전성을 분석하기 위한 결함분석 기법들에 대한 고찰을 본 연구의 기반으로 하여 소프트웨어 라이프사이클 동안에 안전성 확보를 위한 최선의 활동의 방안들에 대하여 연구하였다.

## 1. 서론

컴퓨터 시스템의 이용이 보편화될수록 컴퓨터 시스템에 채용된 소프트웨어의 신뢰성과 안전성은 우리들의 생활에 많은 영향을 미치게 된다. 특히 안전성 임계 소프트웨어 시스템에 의해 유지되어지는 환경 내에서 생활하는 사람에 있어서는 시스템의 안전성이 곧 인간의 생명을 좌우할 정도로 중요한 자리를 차지하게 된다.

안전성이란 사고나 손실로부터 자유로운 상태라고 정의할 수 있는데, 소프트웨어의 경우 외부로부터의 잘못된 입력으로부터 시스템을 안전하게 지켜 주고, 재난의 발생을 막기 위한 통제를 수행할 수 있는가에 관심을 갖는다(Nancy G. Leveson, 1995).

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(Nancy G. Leveson, 1986).

이처럼 안전성과 관련하여 많은 컴퓨터 시스템이 사용되고 내장된 소프트웨어의 안전성에 대하여 사람들의 관심이 높아짐으로써, 개발되고 있거나 양도되어지거나 사용 중인 소프트웨어 시스템이 안전성을 확보하고 있다는 증명이 필요하게 되었다. 따라서 소프드웨어 시스템이 가지고 있는 안전성과 관련한 결함을 분석함으로써 소프트웨어의 안전성을 증명하거나, 발견된 결함을 제거하는 기술에 관하여 보다 심도있는 연구가 필요하다고 판단된다.

소프트웨어의 안전성을 위한 활동에는 시스템 사용자 요구사항 분석에서의 오류 경향과 영향 분석, 소프트웨어 요구사항 분석에서 오류 경향과 임계영향 분석, 설계 단계에서는 폴트트리 분석, 소프트웨어시스템의 코드에 있어서 소프트웨어 폴트트리 분석, 유지보수단계에서의 결함 및 운영가능성 분석 등을 들 수 있다.

소프트웨어의 안전성을 평가하기 위하여 몇 가지 결함 분석 기법들이 알려져 있는데 소프트웨어의 안전성은 어느 하나의 기법만으로는 완벽하게 판단할 수가 없다. 뿐만 아니라 몇몇의 안전성 평가를 위한

결함분석 기법들은 실제 프로젝트에 적용되기 어려운 것도 있으므로, 실제적으로 적용 가능한 안전성 평가 방안을 제시할 필요가 있으며, 소프트웨어 개발 전단계에 걸쳐서 결함이 분석되고 오류 수정이 이루어져야 할 것으로 판단된다.

이에 따라 본 연구에서는 소프트웨어 시스템 개발에 있어서 라이프사이클 동안의 안전성 확보를 위한 활동 방안을 제시하고자 한다.

## 2. 안전성 평가

### 2.1 시스템의 안전성 활동

시스템 안전성 활동들은 프로젝트에 대한 최초의 개념발달 단계에서 시작되어서 설계, 제작, 시험, 조작적 사용, 폐기에 이르기까지 계속된다.

시스템 안전성과 안전성에 대한 다른 접근 방법들이 구별되는 하나의 관점은 위험에 대한 초기의 확인과 분류를 강조하는 것이다. 그 결과 수정활동이 최종적인 설계 결정들이 되어지기 전에 그들 위험들을 제거하거나 줄이도록 할 수 있는 것이다.

(1) 시스템 안전성은 완성된 설계에다 안전성을 추가하는 것이 아니라 안전하게 구축하는 것을 강조한다.

(2) 시스템 안전성은 서브시스템들이나 부품들에 대해서 보다는, 전부로서의 시스템을 다룬다.

(3) 시스템 안전성은 단지 결함들보다는 위험에 대한 더 넓은 관점을 취한다.

(4) 시스템 안전성은 과거의 경험과 표준보다는 분석을 강조한다.

(5) 시스템 안전성은 양적인 접근보다는 질적인 것을 강조한다.

(6) 시스템 안전성은 시스템 설계에 있어서 모순과 갈등들의 중요성을 인정한다.

(7) 시스템 안전성은 단지 시스템 공학 이상의 것이다.

### 2.2 소프트웨어 시스템의 안전성

컴퓨터는 많은 시스템들에서 제어기능을 담당한다. 그것은 다른 구성성분들과의 일반적인 인터페이스를 가지는 것 뿐 아니라 흔히 전자 기계적인 구성성분들의 작동상태와 이들 구성성분들 간의 상호작용들을 제어할 책임이 있다. 대부분의 사고들은 그 인터페이스들과 구성성분들 간의 상호작용에서 일어나기 때문에, 소프트웨어가 시스템 안정성에 직접적이고 중요한 역할을 하고, 시스템 안전성 노력들의 통합적인 부분이 되어야함은 틀림이 없다.

소프트웨어는 두 가지 면에서 시스템 안전성에 영향을 줄 수 있다.

(1) 출력되는 값에 의한 표시 상태와 시스템이 위험스런 상태에 이르는데 걸리는 시간

(2) 몇 가지 면에서 제어하거나 응답하기로 되어 있는 하드웨어 고장들을 인식하거나 조정하는 데 실패할 수 있는 것

이러한 소프트웨어 안전성에 관한 두 측면은 효과적인 시스템 안전성 프로그램에서 다루어져야만 될 이슈이다.

소프트웨어에 관련된 사고들은 소프트웨어가 그 사양을 만족시킬 때와 소프트웨어의 운용 신뢰성이 매우 높을 때 일어나고 있다. 이들 경우에는 다음과 같은 것들이 있다.

(1) 소프트웨어가 그 요구사항을 정확히 구현하지만 그 요구가 시스템의 예상으로부터 안전하지 않은 움직임을 나타낼 수 있다.

(2) 요구사항들이 시스템 안전성에 요구되어지는 어떤 특별한 움직임을 나타내지 못한다.

(3) 소프트웨어는 요구사항에 명세화되어진 바 이상의 동작은 의도되지 않고 불안전한 것을 가진다.

### 2.3 소프트웨어 안전성 평가

소프트웨어의 안전성은 그 정도를 정량적으로 표현하기가 매우 어렵다. 소프트웨어 시스템의 실패는 거의 모두가 항상 복합적인 요인에 의하여 발생된다. 또한, 그 가능성은 매우 작아서 소프트웨어 시스템의 안전성을 평가하는 것은 대단히 어렵다. 예를 들면, 어떤 항공통제 모델에서 원인과 원인의 집합, 실패가 일어나는 횟수가 곧 항공통제가 실수를 일으킬 확률이 되는데 그 확률을 측정하여 안전성을 평가하는 것이 쉽지 않다는 뜻이다.

고전적인 안전성 평가 기법들은 전형적으로 안전성의 한 면으로만 접근을 하고 있는데 실제로는 단순한 프로젝트에 있어서도 다수의 기법 사용을 필요로 하고 있다. 안전성을 평가하는 기법들은 독립적인 것이 아니고, 기법들 사이의 결과 관계가 중요하며, 그것들을 인정할 수 있는 확고하고 완전한 다수의 규칙을 제공한다.(S.P. Wilson, T.P. Kelly, J.A. McDermid , 1995).

안전성은 시스템의 설계 단계에서 핵심적인 요소

가 되어야 한다. 안전성 분석의 초기에서는 종종 요구사항과 설계결정을 이끌어 내지만, 아직까지 초기 분석의 결과를 요구사항과 설계 결정에 연결시키는 것은 매우 어렵다. 그러나 초기단계에서의 안전성을 위한 노력은 소프트웨어 개발에 있어서 비용과 시간의 절감을 가져다 줄 가능성이 높아진다.

## 3. 결함 분석

시스템의 환경 내에서 다른 조건들과 함께 사고나 손실을 일으킬 수 있는 시스템이나 객체의 상태나 일련의 조건들을 결함이라 한다(Nancy G. Leveson, 1995).

소프트웨어 안전성의 관점에서 볼 때 소프트웨어가 동작하는 시스템 내에서 입력된 정보를 분석하여 하드웨어의 동작에 이르기까지의 프로세스를 소프트웨어가 제공한다고 볼 때, 소프트웨어의 결함은 곧 재난의 발생이라고까지 확대할 수 있을 것이다. 안전성 임계 소프트웨어는 안전과 관련하여 결함을 지닌 채로 가동되어서는 안되며, 안전성에 대한 확신이 있을 때 시스템의 핵심적인 한 부분으로써 사용이 가능할 것이다. 알려져 있는 소프트웨어의 결함 분석 모델들이 다음 장에 소개되어 있다.

### 3.1 FMEA(오류경향과 영향 분석; Failure Modes and Effect Analysis)

FMEA는 신뢰성 공학자들이 설비의 신뢰성을 예견할 수 있도록 하기 위해서, 그들에 의해서 발달되어졌다. 그것은 결함요소와 위험보다는 성공적인 기능을 강조하는 신뢰성 분석의 형태이다. 이것의 목적은 제품이 한정된 시간내에서 실패가 없도록 조작하게 하거나, 아니면 제품이 오류들 사이에서 확실한 시간의 길이를 조작하게 하는 종합적인 가능성을 설립하는 것이다.(Nancy G. Leveson, 1995).

FMEA는 기능적 다이어그램과 공학적인 도면작업에서 쉽게 확인될지도 모르는 하드웨어 항목들을 위하여 설계가 진행될 때 적당하다. 시스템 분석가는 모형도, 기능적 다이어그램, 부품 조립 사이의 상호관계성에 대한 정보가 포함된 상세 설계가 필요하다.

FMEA는 단일의 유니트 혹은 단일의 실패를 분석함으로써 하나 하나의 완전 무결한 항목들을 늘리는데 효과적이다. 각각의 실패는 그것이 만들어지는 그 이후의 결과들을 제외하고 시스템 내에서 다른 실패들과 관련이 없는 채로 독립적인 방법으로 다루어진다.

FMEA는 어떤 설계가 알맞게 진행되고 있을 때 적당한데, 그 때는 공학적인 설계와 기능적인 다이어그램 위에서 하드웨어 항목들이 쉽게 구별되어질 수 있는 때이다. FMEA는 개별적인 항목의 통합성을 높이기 위해, 단일 유니트들이나 단일 오류들을 분석하는 데 효과적이다.

모든 중요한 오류경향들은 미리 알려져야한다. 그래서 FMEA는 적고 잘 알려진 오류경향을 가진 표준 부분에 가장 적당하다.

### 3.2 FMECA(오류경향·영향 및 심각성 분석; Fail-ure Modes, Effects, and Criticality Analysis)

FMECA는 기본적으로는 오류의 심각성에 대한 더 상세한 분석을 가진 FMEA이다. 두 단계(보통 칼럼)가 FMEA에서 추가된다.

(1) 이미 존재하거나 제안된 제어의 방법이 결정되어지고,

(2) 연구는 고장의 수정 기회 혹은 그 이상의 제어가 필요한지 아닌지에 대한 지시의 부과 등과 같은 제어 절차들을 고려하여 수정되어진다.

| Item | Failure Modes | Cause of Failure | Possible Effects | Prob. | Level | Possible Action to Reduce Failure Rate or Effects |
|---|---|---|---|---|---|---|
| Motor case | Rupture | Poor workmanship<br>Defective materials<br>Damage during transportation<br>Damage during handling<br>Overpressurization | Destruction of missile | 0.0006 | Critical | • Close control of manufacturing processes to ensure that workmanship meets prescribed standards.<br>• Rigid quality control of basic materials to eliminate defectives.<br>• Inspection and pressure testing of completed cases.<br>• Provision of suitable packaging to protect motor during transportation. |

위의 표는 FMECA의 예시이다.

심각성 순위는 일반적으로 가능성이나 빈도로 표현되어지는바, 이를테면 임계모드에서 수행된 100만번의 오퍼레이션 동안 기대되어진 특정 유형의 오류의 수이다.

### 3.3 SFTA(소프트웨어 폴트 트리 분석 ; Software Fault Tree Analysis)

SFTA는 시스템의 폴트트리에 의해 결정되어진 임계적인 제어 결함으로부터 소프트웨어 입력에 대

한 프로그램 코드나 설계를 통해 반대 방향으로 행해진다. 그 접근은 전형적인 검증 방법이 사용된 백워드 증명과 비슷하다. 그러나 보다 많은 한정된 목표를 가지고 있다.

즉, SFTA는 그것이 안전한 상태 말고는 부정확에 관한 것이 아무것도 증명되지 않는다 하더라도 프로그램이 특별히 불안한 상태에 이르는 것을 용납하지 않을 것이라는 것을 검증해 보려고 한다.

대부분의 리얼타임 내장 시스템들은 두가지의 목표를 가지고 있다. (1)임무와 기능의 성취와 (2)프로세스에 있어서 손해를 유발하지 않기

SFTA는 두 번째 목표를 지향하고 있다.

SFTA에서는 소프트웨어가 불안전한 제어활동을 만들어 낸다고 가정을 하고, 그 가정이 모순에 이르기 때문에 이것이 일어날 수 없다는 것을 보여주는 방법으로 분석이 이루어진다. 소프트웨어와 파생된 제어되는 시스템 내에서나 논리적인 모순을 포함하지 않은 그 환경주 변을 통해, 위험은 접근 가능한 것이고, 이것은 그 시스템 설계에서 고려되어져야 할 필요가 있다.

### 3.4 HAZOP(결함과 운영 가능성 분석 ; hazard and operability studies)

HAZOP는 영국의 ICI에서 1960년대 초에 개발되어 그 후반기에 런던에 있는 화학공업협회에 의하여 개선되고, 공개된 이래 반 이상의 화학 공장들이 모든 새로운 설비에는 HAZOP을 하고 있다. 이 분석법이 제안되었을 때에는 안전에만 초점을 둔 것이 아니고 효율적인 운영에도 초점을 맞추고 있었다. 뿐만 아니라 이것은 고정설비에도 종종 적용되었다.

HAZOP는 앞으로 진행해야 할 때 진행이 되지 않거나, 후진하는 것처럼 계획된 조작이나 설계로부터 이탈된 원인과 같은 사고의 시스템 이론에 기초를 두고 있다(Nancy G. Leveson, 1995).

HAZOP는 프로세스 설명, 흐름도, 제어 로직 다이어그램, 배관과 기계 다이어그램, 설비 배치도, 시험운영, 유지보수와 비상절차, 안전성과 훈련메뉴얼, 화학적, 물리적, 독극물과 같은 모든 속성들이 재료, 중간 생성물과 제품 등에 사용되어진다. 싯점에 따라 이러한 많은 정보들이 유용하게 적용된다.

그런데 설계상의 결함이 확인된다면 이것은 주요한 변경을 너무 늦게 만드는 일이 종종 있다. 그러므로 결함은 대체로 설계 변경에 의하여 제거되기보다는 보호장치의 추가에 의하여 통제된다.

## 4. 라이프사이클 동안의 안전성 활동

안전성 임계 소프트웨어의 안전성을 평가하기 위하여는 소프트웨어가 개발되는 라이프사이클 동안에, 각각의 개발 단계에서 소프트웨어의 특성에 따라 가장 적절한 방법으로 결함이 분석되고, 처치되어야 한다.

다음에 소프트웨어 라이프사이클 동안의 안전성을 평가하기 위한 결함 분석 방안을 제시해 본다.

### 4.1 시스템의 정의

소프트웨어 시스템의 계획 단계에는 소프트웨어를 포함하는 전체 시스템이 갖추어야할 기능과 성능을 파악하고 이를 처리하기 위한 소프트웨어의 기능을 정의한다.

시스템의 범위를 결정하는 것은 모든 시스템에 있어서 매우 중요한 일이다. 시스템의 범위는 소프트웨어 시스템의 크기와 접근의 한계를 결정하는 것으로서 소프트웨어 시스템의 안전성 확보를 위한 위한 활동을 결정하게 된다.

소프트웨어의 안전성이 핵심적인 시스템의 범위를 결정할 때에는 다음의 사항을 고려해야 한다.

소프트웨어를 포함한 시스템의 영역을 결정함에 있어서 시스템과 관련하여 입출력 데이터를 시스템의 내부 활동으로 결정하느냐, 외부 사상으로 결정하느냐는 시스템의 안전성의 평가 척도에 변화가 생긴다.

마찬가지로 소프트웨어의 안전성 평가를 위하여 시스템의 범위를 결정하는 것은 시스템의 안전성 확보를 위한 활동에 영향을 줄 수 있게 되므로, 외부 사상으로 보이기 쉬운 데이터 또는 프로세스일지라도 안전성과 밀접한 관련이 있다고 판단되면 해당하는 데이터 및 프로세스는 시스템의 영역에 반드시 포함시켜야 할 것이다.

### 4.2 요구사항 분석

사용자 요구사항 분석에서의 안전성 확보를 위한 활동은 대단히 중요한 활동이다. 여기에서 가능성 있는 많은 결함들이 발견되고 교정되어야 하기 때문이다. 이 단계에서 안전성 확보의 실패는 시간적, 물질적 손해를 유발시킬 확률이 가장 높기 때문이다.

FMEA는 사용자 요구사항을 반영에서 발생할 수

있는 오류모드를 발견하고 그 오류모드가 사용자 요구사항에 미치는 영향을 예측한다. 이것으로 최상위 기능에서의 오류모드를 분석하고 시스템 수준에서의 영향을 분석한다.

이러한 분석들이 알려지면 그 오류모드들이 시스템에 미치는 심각성을 판단하여 등급을 매길 수 있게 된다. 이러한 절차들은 반드시 정형화되어 있는 것은 아니고 대체로 이러한 절차를 따른다는 것일 뿐이다. 따라서 소프트웨어의 특성에 따라 약간의 방법은 달라질 수 있다. 다음으로 안전성 심각성 등급은 10단계로 나누거나 5단계로 나눌 수 있는데 5단계로 나누었을 경우 다음과 같이 정의할 수 있다.

| 등급 | 시스템에 미치는 영향 |
|---|---|
| 5 | 심각한 손상으로 인한 기계의 비극적인 종결(사람에게 피해를 주는 것) |
| 4 | 심각한 위험, 기계의 손상이나 파괴 등 복구할 수 없는 것 |
| 3 | 작동이 끝나지 않는 것 |
| 2 | 부분적으로만 작동되는 것 |
| 1 | 작동은 완료되었으나 일정에 없었던 유지보수가 필요한 것 |

소프트웨어 개발에 있어서 사용자 요구 기능은 필수적으로 포함되어야 할 것인데 요구사항이 잘 반영되고, 그 기능이 안전하게 동작되어야함은 두 말할 필요가 없을 것이다. 따라서 사용자 요구분석과 심각성 등급에 따라 핵심적인 오류 경향에 대한 결함분석이 이루어져야 할 것이다. 즉, FMECA에 의한 사용자 요구분석을 중심으로, 안전성 심각성 등급을 중심으로 안전성을 확보하기 위한 활동이 가장 적절하다 하겠다.

### 4.3 안전성을 위한 설계 및 분석

시스템 안전성 지침들은 다음과 같은 유형으로 적용되어짐으로써 위험성이 감소되어짐을 알려준다.

(1) 결함의 제거 : 설계들은 결함을 제거함으로써 본질적으로 안전하게 된다.

(2) 결함 감소 : 결함의 발생이 감소된다.

(3) 결함 통제 : 만약 어떤 위험이 일어난다면 그것이 하나의 사고를 일으킬 가능성은 줄어든다.

(4) 손해의 최소화 : 사고로 인한 손실은 종종 시스템 범위 밖에서 일어난 사고 때문에 시스템 설계만으로 제거되지 않을 수 있다.

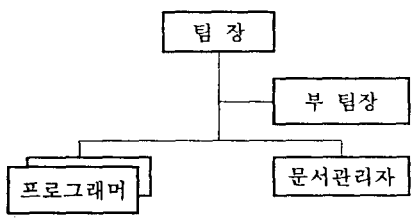
(5) 설계 기법들과 우선 순위 유형들 : 안전설계를 제품내에 포함시키는 아이디어는 새로운 것이 아니다. 시스템 안전성과 신뢰성 공학의 발달이 도움이 되어 고장으로부터 더 표준적인 공학 신뢰성 대신에 사고를 예방하는 것이 강조되어지고 있다.

시스템 안전성 설계 목표의 기본은 인식된 위험들을 제거하는 것이거나 혹은 그것이 불가능하다면 허용할 수 있는 수준까지 관련된 위험을 줄이는 것이다.

### 4.4 소프트웨어의 구현

시스템의 구현이란 시스템 설계 명세서에 따라 프로그래밍언어로 코딩하고 테스트하여 소프트웨어를 현실화시키는 작업을 말한다. 이 단계는 설계가 완료된 시점이므로 프로그래밍팀의 조직이 소프트웨어의 구현에 많은 영향을 미칠 수 있다. 따라서 시스템 분석가는 경험많고 소프트웨어 분석과 설계에 참여하였던 프로그래머를 팀장으로 하여 프로그래밍팀을 구성하여야 한다.

IBM에서는 팀프로그래머방식으로 IPT (improved program technique)를 채택하고 있다.(박재연, 1997)




위와 같은 조직으로 프로그래밍 팀을 구성하는 것은 선행사례에 의한 팀조직을 따름으로서 안전성을 확보한 소프트웨어의 개발에 도움이 되게 하고자 하는 것이다.

이 단계에서 소프트웨어의 안전성 확보는 기술적인 문제보다는 인적 요소들의 영향이 크다고 말할 수 있다. 대체로 민주적이면서 강력한 리더쉽이 있는 팀장의 역량과 원만한 인간관계, 그리고 구성원들의 협력체제 등이 필요하다. 특히 대규모 소프트웨어일수록, 안전성 임계 소프트웨어일수록 프로그래밍팀의 조직은 프로젝트 수행에 많은 영향을 줄 수 있다.

### 4.5 소프트웨어 안전성의 검증

프로그램의 코딩이 완료되면 각각의 소프트웨어 모듈을 통합하여 시험하는 통합시험(integration), 전체 시스템이 요구사항을 만족하는 가를 시험하는 시스템 시험(system test)을 거치게 된다. 이 과정에서 소프트웨어 루틴들에 대하여 각 기능들을 검증함으로써 코드된 소프트웨어의 안전성을 평가할 수 있게 된다. 이 때 소프트웨어의 규모에 따라 소프트웨어의 결함을 분석하는 기법은 달라질 수 있다.

실제 프로젝트에서 적용하여 큰 성과를 얻었던 SFTA는 코드된 소프트웨어 부품들의 기능을 철저하게 분석함으로써 통신위성이 파괴될 수 있었던 오류를 검출한 사례가 있으며, HAZOP는 주로 설비 부분에서 설비가 안전하게 운영될 수 있는지를 평가하기 위하여 사용되었다. SFTA는 규모가 작거나 부품들이 표준화되어 있는 소프트웨어에 적용하기 알맞으며, 소프트웨어의 크기, 특성에 따라 HAZOP 등이 적용되어질 수도 있다.

## 5. 결론 및 향후 연구 과제

소프트웨어의 위험을 제거하기 위하여 소프트웨어의 개발 초기에 기울인 노력은 시스템의 개발 비용을 절감하는 데 기여할 수 있으며, 설계·구현 단계에서의 효율적인 활동은 안전성이 보다 높은 소프트웨어의 개발을 가능성을 더욱 높여 줄 것이고, 안전성과 관련한 최적의 개발 모델이 제시된다면, 사용자에게는 프로그램이 장래에 빠질지도 모르는 위험을 피할 수 있는 가능성이 높아지게 된다.

소프트웨어 생명 주기 모형에서 결함분석 기법이 직접 적용되기 어려운 단계에서는 각 단계의 특성에 따라 안전성의 확보를 위한 의도적인 활동이 이루어져야 한다.

이처럼 소프트웨어 시스템 구축에 있어서 소프트웨어의 안전성을 위한 활동은 소프트웨어 개발의 전 단계에 걸쳐서 수행되어야 하며, 가능하면 소프트웨어 개발의 초기 단계에서 안전성을 위협하는 요소가 발견되고 처치되어야 한다.

소프트웨어의 안전성 평가와 관련하여 앞으로 관심을 가져야 겠다고 판단되는 과제로는

(1) 소프트웨어의 유형 또는 크기에 따른 효율적인 분석 기법의 적용 방안,

(2) 소프트웨어 부품의 안전성 평가를 이용한 소프트웨어시스템의 안전성 평가 등이다.

* 참고문헌 *


[1] James Catmur, Morris Chudleigh, Felix Redmill, "Use of Hazard Analysis Techniques During the Product Life Cycle: HAZOP and FMEA Compared", Safety and Reliability of Software Based Systems, Twelfth Annual CSR Workshop, 1995.

[2] John G. Burch, "Systems Analysis, Design, and Implementation", boyed & fraser publishing company, 1992.

[3] Nancy G.Leveson, Peter R.Harvey, "Analyzing Software Safety", IEEE Tran sactions on Software Engineering, Vol. SE-9. No.5, 1983.

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

[5] Nancy G.Leveson, "Software Safety: Why, What, and How", ACM Computing Surveys, Vol.18, No.2, 1986.

[6] Peter L.Goddard, "Validating The Safety Of Embedded Real-Time Control Systems Using FMEA", Proceedings Annual Reliablity and Maintainability Symposium, 1993.

[7] Rogerio De Lemos, Amer Saeed, Tom Anderson, "Analyzing Safety Requirements for Process Control Systems", IEEE Software, 1995.

[8] Roger S.Pressman, "Software Engineering", McGraw-Hill Companies, 1997.

[9] Samuel J.Keene, "Assuring Software Safety", Proceedings Annual Reliability and Maintainability Symposium, 1992.

[10] Thomas Maier, "FMEA and FTA to Support Safety Design of Embedded Software in Safety-Critical Systems", Safety and Reliability of Software Based Systems, Twelfth Annual CSR Workshop, 1995.

[11] 박재연, "구조적 시스템 분석과 설계", 정익사, 1997.

[12] 왕창종,윤경섭,이세훈, "최신기법의 시스템 분석 및 설계", 정익사, 1997.