KAIS 97 추계학술대회

소프트웨어 안전성 평가를 위한 시스템 결함 분석 기법

백 동 근 · 성 경 배 · 박 만 곤
부경대학교 전자계산학과

요 약

컴퓨터 소프트웨어를 내장한 시스템의 안전성을 평가하기 위한 여러 가지의 결함 분석 기법들이 있다. 이러한 결함 분석 기법들은 전통적으로는 하나의 시스템을 분석하는데 단지 하나의 방법으로만 분석해 왔으나, 시스템의 종류와 특성이 다양해지면서 시스템에 가장 알맞은 분석 기법이 동원되어야 함은 이제 필수적이다.

여기에 착안하여, 시스템 내에서 소프트웨어의 크기가 비교적 작고, 안전성과 관련한 시스템의 반응 시간이 특별히 민감하지 않는 소프트웨어의 안전성을 평가하는 방법으로 결함 트리 분석(FTA)과 소프트웨어 오류경향 및 영향 분석(소프트웨어 FMEA)을 결합한 시스템 결함 분석 방법을 제안하고자 한다.

1. 서론

과거의 인간-기계 시스템에서 안전성을 위하여 인간의 조작으로 통제되던 각종 장치들이 컴퓨터 기술의 발달에 힘입어 안전성 확보를 위한 많은 부분에서 감시·통제 기능을 수행함으로써 인간을 대신하고 있다. 이에 따라 인간은 감시·통제 기능을 하는 소프트웨어 시스템이 정상적으로 동작을 하는지에 관심을 가지게 되었고, 안전성에 핵심을 둔 시스템에서 하드웨어 또는 소프트웨어의 결함으로 재난이 발생하기 시작하자 현재의 시스템이 인간에게 안전을 보장해 줄 것인가에 대한 사람들의 신뢰는 낮아지고, 공학자들은 안전에 핵심을 둔 시스템이 과연 안전한가에 대하여 평가를 시도하게 되었다.

안전성이란 사고나 손실로부터 자유로운 상태라고 정의할 수 있는데, 소프트웨어 시스템의 경우 외부로부터의 잘못된 입력으로부터 시스템을 안전하게 지켜 주고, 재난의 발생을 막기 위한 통제를 수행할 수 있을 것인가에 관심을 갖는 것을 말한다(Nancy G. Leveson, 1995).

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(Nancy G. Leveson, 1986).

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 심도있는 연구가 필요하다고 판단된다. 이는 안전성 확보의 실패는 인간의 생명과도 관계가 있기 때문이다.

시스템 안전성은 1940년대 후반에 관심을 갖기 시작하여 1950년대 후반에 하나의 학문

으로 분리, 정의되었다. 이러한 학문 분리의 근원은 1950~1960년초에 개발되었던 무기 시스템과 관련한 위험들을 제거하는 새로운 접근에서 출발하였다. 대륙간 탄도탄(ICBM)은 형식을 갖추고 시스템의 안전성 프로그램과 관련하여 실험한 최초의 시스템 중의 하나이다. 시스템 안전성은 조작적 영향력, 시간, 비용의 제한 내에서 시스템의 전 일생을 통하여 안전을 책임지기 위하여 과학적, 경영학적, 공학적 원리가 적용되는 한 분야로 발전하고 있다(Nancy G. Leveson, 1986).

시스템의 안전성을 평가하기 위한 결함 분석 기법들로는

(1) 결함 트리 분석(fault tree analysis)

(2) 결함 및 운영가능성 분석(hazard and operability studies)

(3) 오류 경향과 영향 분석(failure modes and effect analysis)

등이 알려져 있으며, 하드웨어, 소프트웨어를 포함한 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 결함분석을 위한 모델 적용에 있어서 상호보완적인 역할을 할 수 있는 두 가지 이상의 분석을 결합(combine)하는 연구가 눈에 띄고 있다.

"제품의 라이프사이클 동안에 결함 분석 기법의 사용 : HAZOP, FMEA 비교" (James Catmur, Morris Chudleigh, Felix Redmill, 1995)에서, HAZOP는 부품 사이의 상호작용을 시험함으로써 결함을 찾는데 반하여, FMEA는 부품 자체에서 가능한 오류를 시험하는 것으로써 HAZOP는 반드시 평가팀에 의하여 수행되지만, FMEA는 대개 개별적으로 수행되기도 한다.

이 두 가지 기법은 종종 혼란한 것이지만, 서로 다른 시스템의 영역 사이의 상호작용이나, 서로 다른 시스템 영역의 현존하는 부품에 초점을 맞춘 것으로 보여진다. 따라서 그것들은 상호보완적인 것이라고 설명하고 있다.

"안전성에 핵심을 둔 시스템에 있어서 내장된 소프트웨어의 안전성 설계 지원을 위한 FMEA와 FTA"(Thomas Maier, 1995)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, 폴트트리의 구축을 통하여 완성될 수 있다. 이 방법론은 전술한 목표들을 넘어서 안전성에 핵심을 둔 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 시스템의 특성과 크기에 따라 가장 적절한 방법으로 시스템이 평가되어야 할 것이다.

따라서 본 연구에서는 시스템 내에서 소프트웨어의 크기가 비교적 작으며, 제어에 있어서 실시간 반응 시간에 아주 특별한 성능을 요구하지 않는 시스템의 소프트웨어 안전성 평가를 위하여 FTA와 소프트웨어 FMEA기법을 결합한 소프트웨어의 결함 분석 방안을 제안하고자 한다.

2. 소프트웨어 안전성의 평가

안전성은 그 정도를 정량적으로 표현하기가 매우 어렵다. 소프트웨어 시스템의 실패는 거의 모두가 항상 복합적인 요인에 의하여 발생된다. 더 나아가서, 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 된다. 엄격한 평가를 준비하는 것은 매우 어렵지만 소프트웨어 시스템의 안전성을 평가하기 위한 측정의 시도는 이루어지고 있다. 안전성은 시스템의 품질이다. 소프트웨어의 안전성을 평가하기 위한 모델은 소프트웨어의 모든 기능에 대한 평가를 고려해야 한다(Nancy G. Leveson, 1986).

일반적으로 재난이 일어날 확률을 M이라 하며,

M = Pr(결함) * Pr(결함으로 인한 재난)

예를 들어 로봇의 움직임을 통제할 제어 기능을 가진 컴퓨터가 있다고 할 때 간단한

모델을 구성해 보면 다음과 같다.

M=Pr(컴퓨터가 이상 동작을 할 확률)
* Pr(인간의 움직임 영역)
* Pr(사람이 움직일 시간이 없거나 고장 진단에 실패할 확률)

컴퓨터가 연속적인 방어나 감시기능을 가졌을 때, 안전기능에서 첫 번째로 요구되는 잠재적인 결함 조건을 검출하는 다른 예를 보면,

M=f{Pr(위험한 조건을 발생하는 설비),
Pr(컴퓨터가 위험사태 발견 실패),
Pr(안전 기능 시작에 실패),
Pr(위험을 예방하는 안전 기능 실패),
Pr(위험이 재난을 이끌어내는 조건 발생)}

표준화 검사와 시스템의 복잡도에 의한 상세한 요구의 증가로 안전성의 사례가 극적으로 성장하고 있어서 시스템 안전성의 전 장면에서 이루어지는 유지보수를 어렵게 하고 있으며, 낮은 수준의 분석들과 대형 시스템을 위한 증거 지원들은 고수준의 논의에서 볼 때는 명백하게 위험에 빠뜨릴 수 있다. 현재의 안전성 사례는 단순히 선형적인 모양을 하고 있으며, 안전성 사례의 높은 수준의 구조적 논의와 증거를 지원하는 관계를 보여주는 것은 매우 어렵다. 안전성 사례를 증명한다는 것은, 항상 완전한 것도 아니며, 이것의 안전성을 확신한다는 것 또한 어렵다.

고전적인 안전성 평가 기법들은 전형적으로 시스템 안전성의 한 면으로만 접근을 하고 있는데 실제로는 단순한 프로젝트에 있어서도 다수의 기법 사용을 필요로 하고 있다. 안전성을 평가하는 기법들은 독립적인 것이 아니고, 사실은 기법들 사이의 결과 관계가 중요하며, 그것들을 인정할 수 있는 확고하고 완전한 다수의 규칙을 제공한다. 그럼에도 불구하고, 시스템을 위한 개별적인 분석들 사이의 관계는 특별한 기초 위에서 자주 다루어지며, 실제적인 안전성 사례의 분석에 있어서는 분석들 사이에 특별한 의미를 두지 않고 있는 경우가 많다(S.P. Wilson, T.P. Kelly, J.A. McDermid, 1995).

안전성은 시스템의 설계 단계에서 핵심적인 요소가 되어야 한다. 안전성 분석의 초기에서는 종종 요구사항과 설계결정을 이끌어 낸다. 그러나 아직까지 초기분석의 결과를 요구사항과 설계 결정에 연결시키는 것은 매우 어렵다.

3. 시스템 결함(system hazard)

사고를 예방하기 위해서는 사고의 조짐에 대한 무엇인가의 정보가 있어야 하고, 그 조짐은 시스템 운영자에 의하여 통제될 수 있어야 한다. 시스템에 있어서 객체의 실체나 시스템의 선천적인 속성, 잠재적으로 재난의 원인이 되는 것을 위험이라 하며, 내용 그 자체는 위험이 아니다. 오히려 결함은 일련의 조건이 결합된 것이다. 예를 들면 염소는 공기 중에 알맞게 포함되어 있을 때에는 해로운 것이 아니지만, 많은 양이 방출되면 그것은 해로운 것이 되는 것이다. 또다른 예를 들면, 물 그 자체는 위험한 것이 아니지만, 몇 몇 조건의 결합으로 익사, 물에 데임, 교통사고 등을 유발하게 됨을 쉽게 생각할 수 있다. 결함의 정의에 있어서 문제는 잠재적으로 재난을 일으키거나 사고를 이끌어 낼 수 있는 것이 대부분의 상태라는 것이다. 이상과 같은 논의를 통하여 결함을 한마디로 다음과 같이 정의할 수 있을 것이다.

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 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 사례를 생각해 보면,

* 입력 정보 오류
* 입력 정보 분석의 오류
* 출력 제어 신호의 오류

등을 생각할 수 있다. 안전성에 핵심을 둔 소프트웨어는 안전과 관련하여 결함을 지닌채

로 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. 시스템 결함 분석

시스템의 결함 분석은 다음과 같은 목적으로 이루어진다(Nancy G. Leveson, 1995).

* 발달(development): 새로운 시스템의 잠재적인 결함을 인식하고 평가하며, 그것을 제거하거나 통제하기 위한 시험.

* 조작적 관리(operational management): 현존하는 시스템의 안전성의 수준을 개선하기 위하여 능률과 운전의 안전성을 높이기 위한 동기를 증가시키기 위한 시험.

* 검정(certification): 설계된 것이거나 현존하는 시스템의 안전성 수준을 증명하며, 공공기관이나 대중의 승인을 받기 위한 시험을 목적으로 이루어지는 시험.

또한 시스템의 프로세스 단계에 따라 다음 네가지의 분석 방법이 있다(Harold W. Lawson, 1995).

(1) 예비 결함 분석(PHA ; Preliminary hazard analysis): PHA는 정밀 시스템 기능들과 광범위한 시스템 결함의 인식을 위하여 일생의 초기단계에서 사용되어 진다. 인식된 결함들은 종종 평가되고 우선순위가 매겨질 때, 안전성 요구사항과 안전설계의 판단기준이 확인될 것이다.

PHA는 초기의 개념탐구 단계에서 시작되거나 라이프사이클의 최초 단계에서 시작된다.

프로세스는 PHA는 설계가 포함한 것과 변경된 것에 대한 보다 많은 정보에 대한 갱신 등으로 반복된다. 결과는 이후의 분석의 기준으로 서비스되고, 시스템 안전성 요구사항과 설계 명세와 성능의 준비에 사용되어 질 수 있다.

(2) 하부시스템 결함 분석(SSHA ; Sub system hazard analysis): SSHA는 서브시스템이 충분히 자세하게 설계되면 곧바로 시작되고, 설계가 완료되었을 때 갱신된다. 설계의 변경은 시스템의 안전성에 영향을 주었는지 결정을 위하여 평가되고 영향을 받는다. SSHA의 목적은 하부시스템의 설계, 포함된 부품 실패 모델, 위험스런 사람의 입력 에러, 부품과 장비로 구성된 각각의 하부시스템 사이의 기능과 관련하여 도출되는 위험들을 감지하는 데 있다.

소프트웨어 결함 분석은 SSHA의 일종이다.

이 분석은 서브시스템이나 부품, 조작과 관련된 결함 인식 또는 실패 모드, 포함된 성능, 성능의 저하, 기능적 실패, 부주의한 작용 등을 자세히 관찰한다. SSHA는 특별히 부품 또는 소프트웨어 통제 기능의 실패가 시스템의 안전성 전반에 걸쳐 어떤 영향을 주는 지를 결정하기 위한 것이다. 이것은 제거하는 방법을 결정하거나 인식된 재난의 위험을 감소시키기 위하여 필요한 활동들의 확인을 포함한다.

(3) 시스템 결함 분석(SHA ; System hazard analysis): SHA는 설계가 완성되었을 때 시작되어- 최초 설계의 재고 - 설계가 수정되고 그것이 완료될 때까지 계속된다. 역시 설계 변경은 평가를 필요로 한다. 안전성에 핵심적인 인간의 잠재적인 에러를 포함하여, 서브시스템이나 시스템의 조작 전반에 걸친 두 가지의 인터페이스에서 생성되어진 가능성있는 재난의 상세한 연구를 수반한다. 특별히, SHA는 ⓐ 안전성 측면에서 시스템 요구사항의 승인; ⓑ 가능한 독립적인 결함, 독립적이고, 종속적이며, 그리고 동시에 발생하는 재난이나 실패, 조정이나 재난의 원인이 된 안전성 장치의 실패의 결합; ⓒ 하부시스템의 정상적인 조작에서 비롯된 시스템 안전성의 저하 등 모든 서브시스템의 인터페이스를 시험한다.

이것은 SSHA와 같은 방법으로 성취된다. 어쨌든 SSHA 어떤 부품의 동작이나 실패가 시스템에 어떤 영향을 주는지를 시험하는 것임에 반하여, SHA는 어떤 시스템의 동작과 오류 경향이 시스템과 하부시스템의 안전성에 영향을 줄 수 있는지를 결정하는 것이다.

(4) 운영 · 지원 결함 분석(OSHA; Operating and support hazard analysis): OSHA는 시스템이 사용되고 유지보수되는 전 단계에서 재난과 위험의 감소 절차 중에 확인된다. 이것은 특별히 인간-기계사이의 인터페이스에서 발생되는 재난을 시험하는 것이다.

4. 시스템 결함 분석 기법들

4.1. 결함트리 분석 (FTA; fault tree analysis)

FTA는 항공우주, 전자공학 그리고 핵산업 등에 폭넓게 사용되어진다. 그것은 원래 대륙간 탄도탄 발사 통제 시스템을 검토하기 위하여 Bell telephone 연구소의 H.A. Watson에 의해 1961년에 발전되어지고, 보잉사의 기술자들과 수학자들이 더 진전된 방향으로 발전시켰으며 FTA의 첫 번째 제안자들이 되었다.

FTA는 주로 위험요소들을 확인하는 것이 아니라, 위험요소들의 원인을 분석하기 위한 수단이다(Nancy G. Leveson, 1995).

트리구조의 상위 레벨에 있는 사건은 다른 기술들에 의해 예견되어지고 확인되어져야만 한다. FTA는 위험이 많은 사건을 구성할 수 있는 개별적인 오류들의 결합들을 설명하기 위하여 Boolean 논리를 사용한다. 트리의 각 단계는 그 단계의 위쪽에서 보여진 문제의 원인이 될 정도로 필요하고 충분한 더 많은 기초적 사건들을 목록화 한다.

FTA는 top- down 방식의 탐구방법이다. 뒤쪽 방향 혹은 앞쪽 방향의 탐구 기술들은 시간을 넘어서서 사건들의 연대순 배열이다. 그러나 결함트리의 각 단계는 단지 더 자세하게 같은 것을 보여준다. 중간에 있는 사건들은 가상의 사건이다. 그것들은 다만 결합체들이거나 기초적 혹은 주요 사건들의 세트들이다. 그리고 그것들은 대체로 트리의 형식적인 분석 동안에 제거되어 진다.

트리가 조립되자마자 그것은 Boolean 표현으로 쓰여질 수 있고, 원하지 않은 상위 레벨의 사건의 원인이 되는 기초적 사건들의 특별한 결합체들을 충분히 보여주기 위해 단순화되어질 수도 있다. 만약 양적인 분석이 요구되어지고, 가능성이 있다면 상위 레벨 사건의 빈번성은 예측되어 질 수 있다. 결함트리 분석은 다음의 4가지 기초 단계들을 가진다.

(1) 시스템의 정의: 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.

분석자는 기능적 다이어그램, 흐름 다이어그램, 로직 다이어그램, 설계도 등이 필요할 것이다. 특히 물리적인 시스템 경계의 정의도 매우 중요하다.

(2) 결함트리 조립: 시스템을 정의하고 난 뒤, 다음 단계는 결함트리의 조립이다. 간단히 분석자는 먼저 개개의 시스템의 상황과 상위 레벨 사건을 가정해야 하고, 그 다음에 관계들을 설명하기 위해 논리적 기호들을 사용해서 상위 레벨 사건과 관련되어진 원인이 되는 사건들과 그 사이에 있는 논리적 관계들을 기록한다. 트리에 사용되는 기호는 다음 <그림 1>과 같다.

기 호	설 명
	논리 게이트를 통해 사건들의 결합으로 생겨난 사건.
	더 이상의 발달을 요구하지 않는 기본적 오류 사건.
	더 이상 발달되어지지 않은 오류 사건, 왜냐하면 사건은 필연적인 것이 아니거나 필요한 정보는 쓸모있는 것이 아니기 때문일 때.
	정상적으로 발생할 것으로 기대되어질 수 있는 사건.
	게이트의 출력을 생산하는데 고려되어져야만 하는 상태.
	이 동
	**AND** 게이트
	**OR** 게이트
	**INHIBIT** 게이트

<그림 1> 결함트리 기호

(3) 정성분석: 트리가 조립되어진 후에 질적인 분석이 시작될 수 있다. 기본적으로, 그 목적은 트리를 논리적으로 동등한 형태로 상위 레벨 사건의 원인이 될 정도로 충분한 기초 사건들의 특별한 결합들을 보여주는 것으로 감소하는 것이다. 본질적으로 중간에 있는 가상의 사건들은 제거되어지고 단지 상위 레벨 사건과 초기 사건들 사이에 있는 관계들만이 설명되어질 뿐이다. 이것은 cut sets라 불리어지는데 분석의 목표는 가장 작은 cut set를 발견하는 것이다.

(4) 정량분석: 논리적 게이트의 출력 가능성은 입력 사건들과 부합하는 가능성과 같다. 결함트리의 양적인 분석은 기초적 사건들의 발생 가능성으로부터 상위 레벨 사건의 발생 가능성을 계산하기 위해서 최소의 cut sets를 사용한다. 상위 레벨 사건의 가능성이 모든 통계학상으로 독립적이라면 모든 cut sets의 가능성의 총계가 될 것이다.

4.2. 결함과 운영 가능성 분석 (HAZOP;hazard & operability studies)

HAZOP는 영국의 제국화학공업(Imperial Chemical Industry)에서 1960년대 초에 개발되어 그 후반기에 런던에 있는 화학공업협회(Chemical Industries Association)에 의하여 개선되고, 공개되었다. 이제는 반 이상의 화학공장들이 모든 새로운 설비에는 HAZOP을 하고 있다. 이 분석법이 제안되었을 때에는 안전에만 초점을 둔 것이 아니고 효율적인 운영에도 초점을 맞추고 있었다. 뿐만 아니라 이것은 고정 설비에도 종종 적용되었는데, 탱크 트럭들을 위한 응용을 설명함으로써 몇 몇의 검출되지 않은 결함이 확인되고 제거하거나 통제된 사례가 있었다.

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

기본적으로 기법은 결함이나 운영상에 나타날 수 있는 문제점 같은 모든 가능한 경로에 대한 창의적인 생각을 유도한다. HAZOP는 조직적으로 수행되고, 설비에 있어서 각각의 프로세스 유니트들이 고려되고, 각 결함들이 변화된다. 설계에 대한 질문들이 전문가들의 작은 팀에서 생성되고, 팀구성원의 상호작용으로 질문들이 창의적으로 제기되고, 색인목록이 즉시 알려진다. HAZOP는 기대되는 조작 설계로부터 이탈될 수 있는 모든 가능성을 확인하기 위하여 제안된 정성적인 분석기법으로써 체크리스트법과 비슷하다.

HAZOP는 새로운 설계 내에 있거나, 이전에 고려하지 못했던 결함을 밝혀 낼 수 있는데 이는 분석 이후에 밝혀지게 된다.

HAZOP는 프로세스 설명, 흐름도, 제어로직 다이어그램, 배관과 기계 다이어그램, 설비 배치도, 시험운영, 유지보수와 비상실차, 안전성과 훈련메뉴얼, 화학적, 물리적, 독극물과 같은 모든 속성들이 재료, 중간 생성물과 제품 등에 사용되어진다. 싯점에 따라 이러한 많은 정보는 유용하다.

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

4.3. 오류경향과 영향 분석(FMEA ; 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.

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. 그리고 나서 각 오류경향의 결과에 관한 가능성과 심각성이 예측되어진다. 부품 오류 비율들은 경험

에 의해 발전되어졌던 일반적인 비율로부터 예견되어지고 종종 알려진다. 정보 센터들은 그러한 정보를 수집하고 대조하고, 제조자들은 그들 자신의 제품을 위해 대체적으로 이 데이터를 가진다. 부품이 작동하고 있는 환경이 통계학에 수집되어지기 위한 하나의 환경과 일치하도록 돌보아져야만 한다.

FMEA는 기능적 다이어그램과 공학적인 도면작업에서 쉽게 확인될지도 모르는 하드웨어 항목들을 위하여 설계가 진행될 때 적당하다. 분석가는 모형도, 기능적 다이어그램, 부품 조립 사이의 상호관계성에 대한 정보를 포함한 상세한 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. 분석을 단일 유니트로 한계 지움으로 인하여, 다양한 혹은 공통된 원인을 가진 실패를 고려하지 않은 채 오류경향과 영향 분석기법이 적용되기에 간단하게 되어지고 시험은 매우 정돈되지만, 만약 시간 연속성과 복잡한 시스템의 요소들 사이에서의 상호관련성이 고려되지 않는다면 제한적으로 사용될 수 밖에 없을지도 모른다(Nancy G. Leveson, 1995).

4.4. 소프트웨어 FMEA (software failure modes and effect analysis)

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 오류 경향 맵의 결과적인 영향들이 위험하다고 정의된 일련의 핵심적인 변수값으로 나타날 때 그 오류는 잠재적으로 하나의 결함의 원인이 되는 것이다(Peter L. Goddard, 1993).

소프트웨어 FMEA를 수행하는 절차는 다음과 같다.

(1) 소프트웨어 위험성 분석

소프트웨어 FMEA는 시스템 결함이 소프트웨어와 호환 가능한 조건으로 변환될 수 있어야 하는데 이는 소프트웨어 오류 요인들의 영향이 시스템 결함으로 평가될 수 있도록 하기 위해서 이다. 이것이 소프트웨어 FMEA의 첫번째 단계이다.

각각 제어 프로세스에 정의된 시스템 결함들은 소프트웨어 변수들의 세트로 번역되어지고 변수값과 연합하게 되는데, 시스템 수준에서 결함과 직접 관련된 변수의 집합들로 변환할 수 있다.

(2) 입력변수 오류 경향

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 경향

분석가들은 루틴의 논리적 기능을 기초로 하여 분석된 루틴의 소프트웨어 로직 오류 경향을 성장시킨다. 예를 들면, 계산루틴의 오류 경향은 계산된 값이 크거나, 작은 결과로 나타나게 될 것이다.

(4) 출력변수 오류의 영향

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.

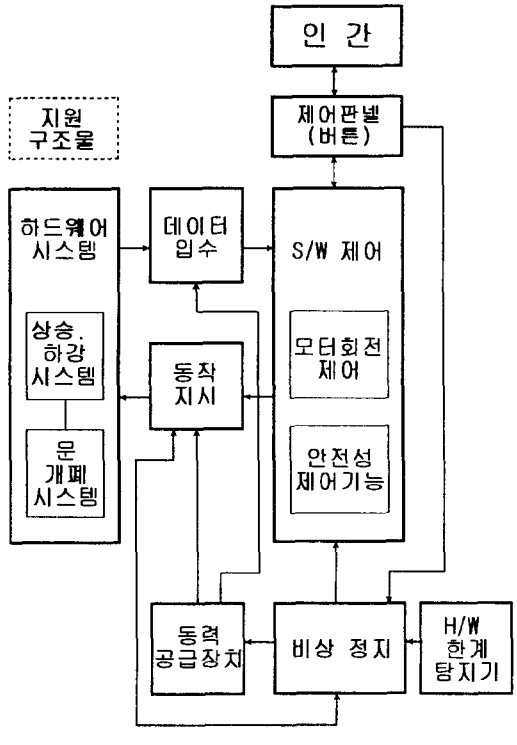
5. FTA와 소프트웨어 FMEA를 결합한 시스템 결함 분석

안전성에 핵심을 둔 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 엘리베이터 시스템에서 사용된 소프트웨어의 안전성을 평가하는 모형으로써 엘리베이터를 가상으로 설계한 다음 그 소프트웨어의 결함을 분석하는 방법을 보인 것이다.

5.1. 시스템의 정의

어느 시스템이든 안전성 분석을 위하여 시스템 영역이 정의되어야 하는데 본 연구에서는 최상위 수준에서의 엘리베이터 시스템을 블록 다이어그램으로 다음 <그림 2>와 같이 표현하였다.



<그림 2> 엘리베이터 시스템 블록 다이어그램

5.2. 기능별 제어 분석

엘리베이터 시스템의 소프트웨어에 포함되어야 할 안전성 기능 요소들을 열거하면 다음과 같다.

* 검출 기능
  - 버튼 입력 검출
  - 위치 제어 에러 검출
  - 가·감속 에러 검출
  - 최대속도 초과 에러 검출
  - 문닫기 동작에서 장애 요소 검출
* 움직임
  - 돌발사태에서의 멈춤 기능
  - 일상적인 조심스러운 멈춤
  - 문 열기
  - 문 닫기

안전성과 관련하여 엘리베이터 시스템의 소프트웨어 기능은 다음과 같은 요소들이 증명되어야 한다.

* 소프트웨어와 관련된 정보의 입수
* 모터의 제어(회전속도, 가속, 감속)
* 소프트웨어와 관련된 제어 판넬

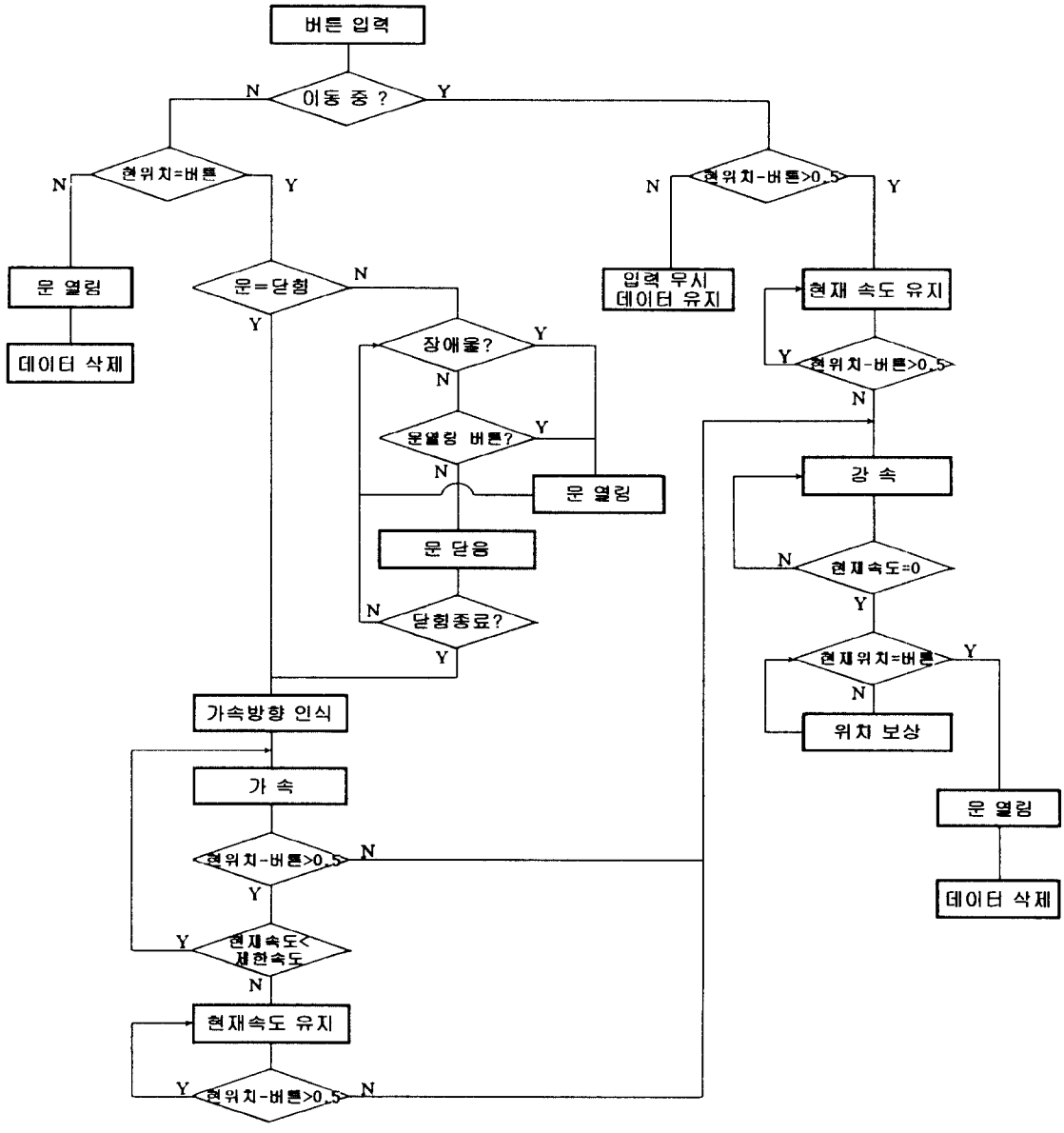
이상과 같이 시스템에서 요구되는 기능들이 알려지고, 소프트웨어 설계가 완료되는 시점에서 FTA는 시작될 수 있을 것이다.

엘리베이터 제어 시스템에서 정상 동작을 위한 최상위 수준에서의 시스템 기능을 다음의 <그림 3>과 같이 분석하였다. 이 모형이 계속되는 분석에 적용되어 질 것이다.

5.3. FTA에 의한 결함 분석

FTA는 위험한 사건이나 시스템 상태를 이끌어 낼 수 있는 사건들의 순차적이고 동시 결합인 그래프적인 모형을 만들어 준다. 잘 개발된 결함트리를 사용해서 다른 시스템 요소들에서 발생하는 일련의 상호 연관된 결함의 결과를 관측하는 것이 가능하게 해주는 분석 방법이다.

시스템이 정의되고 설계가 완료된 단계에서 FTA를 통하여 결함을 분석하였다



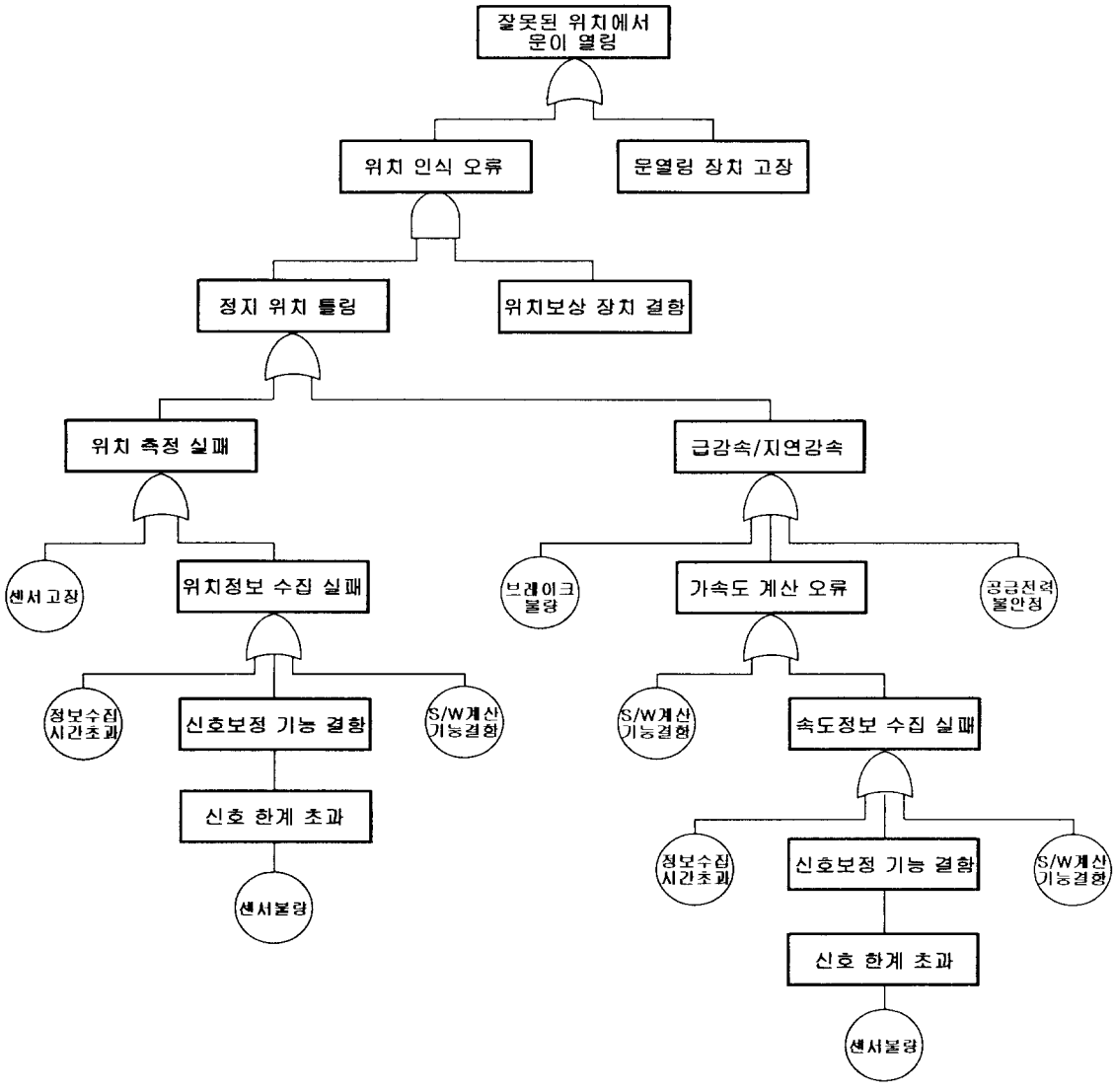
<그림 3> 엘리베이터 시스템의 기능 모형

다음의 예시는 엘리베이터의 움직임을 제어하는 시스템의 안전성을 평가하기 위하여 시스템의 결함과 관련된 요구사항을 분석한 것이다. 잘못된 위치에서 문이 열리는 결함에 대한 시스템 최상위 수준에서의 결함트리는 다음의 <그림 4>와 같다.

이상과 같이 결함트리가 조립되어 안전성에 핵심을 둔 시스템에 요구되는 기능, 시스템 결함의 원인이 파악됨으로써, 설계된 시스템의 정성적, 정량적 분석이 시작될 수 있다.

기초적인 결함 사건이 일어날 가능성은 시뮬레이션을 통하여 확인될 수 있고, FTA는 양적인 분석을 위하여 발달되어졌지만 일반적으로는 질적인 분석으로 사용되어 진다 (Nancy G. Leveson, 1995).

분석자는 결함트리를 조립하는 동안에 매우 상세하게 시스템에 대한 이해를 요구하기 때문에 이 단계에서 대부분의 결함들이 발견되어지고 수정이 될 수 있다.



<그림 4> 잘못된 위치에서 문이 열리는 결함에 대한 최상위 수준의 결함트리

5.4. 소프트웨어 FMEA에 의한 소프트웨어 결함 분석

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 로직으로 발전시킨다.

소프트웨어 FMEA는 각각의 소프트웨어 루틴들에 있어서 입력변수와 소프트웨어 로직의 오류경향이 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.

다음은 엘리베이터 시스템 중에서 잘못된 위치에서 문이 열리는 결함에 대하여 소프트웨어 FMEA 기법으로 분석한 것이다.

5.4.1. 소프트웨어 결함 분석 (software hazard analysis)

소프트웨어에 존재하고 있는 결함들이 알려지기 위하여 안전성과 관련이 있는 변수들을 먼저 확인한다. 각각 제어 프로세스에 정의된 시스템 결함들은 소프트웨어 변수들의 세트로 전환되고 변수값으로 연합하게 된다.

소프트웨어 변수들을 위한 시스템 결함의 결함트리 매핑을 끝내기 위한 상세 설계 수준은 시스템에 의존된다. 최적의 상세 수준은 서로 독립적인 일련의 변수들을 최소 크기로 매핑하는 것을 준비하는 것과 소프트웨어 변수 조건내에서 결함이 발생하는 조건을 충분히 묘사하는 것이다.

결함트리 분석에서 소프트웨어의 변수 요소를 포함하여 표 모양으로 재구성된 결함트리 분석의 결과가 다음 <표 1>에 표현되어 있다. 이 표는 누구든지 시스템 결함과 소프트웨어 실패 영향에 대하여 쉽게 확인할 수 있도록 해준다.

<표 1> 소프트웨어 결함과 관련된 변수

결함 및 원인 \ 안전과 관련된 변수들		pstn	pr_ps	p_dat	pr_pd	p_calc	speed	pr_sp	s_dat	pr_sd	s_calc	refr	acel	p_flg	s_flg	open
위치 정보 수집 실패	시간초과											√				
	신호보정 오류	√	√	√	√									√		
	위치 계산 오류	√		√	√	√										√
속도 정보 수집 실패	시간 초과											√				
	신호보정 오류						√	√	√	√					√	
	속도 계산 오류						√		√	√	√					
가속도 계산 실패	가속도 계산 오류						√	√			√		√			

5.4.2. 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. <표 2>에 보여지는 소프트웨어 변수의 오류 경향들은 변수형에 있어서 잘못된 값을 가지게 되는 경우이다.

시스템의 반응을 위한 정보의 획득 단계에서 이러한 입력변수들이 가질 수 있는 값의 한계를 초과했는지를 확인함으로써 시스템의 결함을 빠르게 발견하게 되고, 그 원인도 찾기 쉽게 될 수 있을 것이다. 오류 경향들이 변수의 성공적인 상태와 결합할 때 모든 가능한 변수들의 상태가 논리적으로 완전한 문장이 될 수 있다.

<표 2> 소프트웨어 입력 변수 오류 경향

변수형	변수명	오류 경향	영 향
실수형	p_dat	신호 없음, 신호한계초과	신호 보정 기능 요구
	s_dat	신호 없음, 신호한계초과	신호 보정 기능요구
	refr	제한 시간 초과	위치, 속도 정보 수집 실패

5.4.3. 소프트웨어 로직 오류 경향

소프트웨어의 로직이 바르게 표현되어 있을 때 정확하게 계산된 값이 가능해지고, 기능적으로 완전한 로직 세트가 가능해진다.

이것은 소프트웨어 로직의 오류 경향에서 분석될 수 있으며, 역시 입력변수와 출력변수의 맵을 통하여 확인될 수 있다.

<표 3> 소프트웨어 로직 관련 출력변수

소프트웨어 로직	변수형	출력 변수	영 향
위치계산	실수형	pstn	잘못된 위치에 정지
속도계산		speed	과속, 초저속 운행
가속도 계산		acel	급가속, 급감속
위치 입력신호 한계 검사	논리형	p_flg	잘못된 정보 전달, 위치 보상 루틴 미작동
속도 입력신호 한계 검사		s_flg	잘못된 정보 전달, 속도제한 루틴 미작동
정지위치 계산		open	잘못된 위치에서 문 열림

5.4.4. 오류가 출력변수에 주는 영향

각각의 오류의 유형들이 출력변수에 미치는 영향과 결합하여 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 경우의 두 가지가 있을 수 있다.

소프트웨어 FMEA를 통한 매트릭스를 구성한 표는 다음 <표 4>와 같다.

이상과 같은 자료를 기초로 하여 소프트웨어의 매트릭스를 구성하고 소프트웨어의 변수값의 변화를 추적하면 소프트웨어의 결함이 발생한 위치와 결함의 유형, 그리고 실패로 인한 영향의 종류 또는 정도를 파악할 수 있게 된다.

이 때 소프트웨어 FMEA의 결과를 FTA의 트리에서 결함이 발생한 기능 루틴을 확인할 수 있고, 그 결함이 다른 기능에 주는 영향과 파급 범위를 확인할 수 있을 것이다.

소프트웨어에 결함이 발생했을 때 변수값의 추적으로 결함 발생의 위치를 확인하고, 결함트리에서 원인을 발견할 수 있을 것이다.

5.4.5. FTA와 소프트웨어 FMEA 결합의 장점

주요 시스템의 결함 분석을 위한 FTA는 소프트웨어의 안전성 기능을 유도하고, 안전성에 핵심이 될 수 있는 하드웨어-소프트웨어 상호작용 부분의 조직적인 접근방식이다. 안전성에 핵심을 둔 하드웨어-소프트웨어 상호작용은 결함트리를 통하여 모든 소프트웨어가 하드웨어 부품에 미칠 때까지 성장함으로써 결정될 수 있다(Thomas Maier, 1995). FTA가 가장 효력이 있기 위해서는 완전하게 되어진 시스템 설계와 모든 작동하는 모드들 내에서 시스템과 그것의 동작에 대한 완전한 이해를 요구한다. 또한 연동장치가 요구되어지는 곳에서는 일찍 설계에 사용되어질 수 있고 그것은 효력이 있는 것이 될 수 있다(Nancy G. Leveson, 1995).

소프트웨어 FMEA는 소프트웨어의 입력변수의 오류 경향 분석, 소프트웨어 로직 분석, 출력변수에서 나타나는 영향에 대한 매트릭스를 구성함으로써, 소프트웨어 설계가 가진 결함을 정확하고 유효하게 확인시킬 수 있다.

이에 따라 FTA는 적어도 설계가 완료된 시점에서 시작되어 시스템에 대한 결함이 분석되고, 결함트리는 시스템의 한부분을 차지하는 소프트웨어에 대하여 하드웨어-소프트웨어 상호작용부분에서 존재하는 결함을 확인해 준다. 소프트웨어 FMEA에서 생성된 매트릭스를 이용하여 변수 값의 변화를 감시함으로써 소프트웨어의 결함이 시스템에 주는 영향 분석 및 결함의 위치를 확인하게 되고, 결함트리에서 결함의 원인을 찾음으로써 시스템을 안전하게 하는데 도움이 되게 할 수 있을 것이다.

뿐만 아니라 완전한 분석의 성취를 위하여 각 장면에서 가장 적절한 기법을 선택하는 현명함이 요구된다. 어느 안전 시스템이든 발달 동안에 결함 분석이 반복되어 질 때만이 성취될 수 있으며, 따라서 소프트웨어의 전일생을 통하여 결함 분석이 이루어져야 한다고 판단된다.

<표 4> 소프트웨어 FMEA 매트릭스

오류 성향 \ 출력변수		pstn	pr_ps	pr_pd	p_calc	speed	pr_sp	pr_sd	s_calc	acel	p_flg	s_flg	open
p_dat	신호 보정 오류	pr_ps와 차가 너무 크거나 작음	.	p_dat와 차가 너무 크거나 작음	.	.	.	.	.	.	= 'F'	.	.
	위치 계산 오류	.	.	p_dat와 차가 너무 크거나 작음	pr_ps와 차가 너무 크거자 작음	.	.	.	.	.	.	.	.
s_dat	신호 보정 오류	.	.	.	.	pr_sp와 차가 너무 크거나 작음	.	s_dat와 차가 너무 크거나 작음	.	.	.	= 'F'	.
	속도 계산 오류	.	.	.	.	.	.	s_dat와 차가 너무 크거나 작음	speed값과 차가 너무 크거나 작음	.	.	.	.
	가속도 계산 오류	.	.	.	.	.	.	.	.	실제 가속도와 차가 너쿠 크거나 작음	.	.	.
refr	정보 수집 시간 초과	refr > (소프트웨어에서 기대하는 제한 시간)											
프로세스 로직	위치신호가 범위를 벗어났으나 신호보정 루틴 미실행	p_calc값과 다름	p_calc와 차가 너무 크거자 작음	.	.	.	.	.	.	.	= 'T'	.	.
	속도신호가 범위를 벗어났으나 신호보정 루틴 미실행	.	.	.	.	.	s_calc와 차가 너무 크거자 작음	.	.	.	.	= 'T'	.
	정지위치가 잘못되었으나 위치보상 장치 미작동	p_calc값과 다름	.	.	.	.	.	.	.	.	.	.	= 'T'
	위치 계산 루틴 오류	.	.	.	실제 위치 계산값과 다름	.	.	.	.	.	.	.	.
	속도 계산 루틴 오류	.	.	.	.	.	.	.	실제 속도 계산값과 다름	.	.	.	.
	가속도 계산 루틴 오류	.	.	.	.	.	.	.	.	실제 가속도 계산값과 다름	.	.	.

6. 결론 및 향후 연구 과제

소프트웨어의 안전성을 평가하기 위하여 몇 가지 결함 분석 기법들이 알려져 있다. 소프트웨어의 안전성은 어느 하나의 기법만으로는 완벽하게 판단할 수가 없다. 소프트웨어의 안전성을 보다 엄격하게 평가할 수 있는 방안의 하나로 상호보완적인 관계에 있는 결함 분석 기법을 이용하여 안전성을 분석하는 방법으로 FTA와 소프트웨어 FMEA 분석 기법을 결합한 방법을 제안하였다.

이러한 상호보완적인 결함 분석이 필요한 것은 현존하는 어느 분석 기법이든 그 기법이 모든 소프트웨어에 공통으로 최고의 적용은 될 수 없기 때문이다.

본 연구에서 예시한 엘리베이터의 움직임을 제어하는 소프트웨어의 안전성을 평가함에 있어서, FTA에서 얻은 결함의 원인과 소프트웨어 FMEA를 결합할 때 소프트웨어 전체의 결함의 위치와 원인을 분석할 수 있을 것이다. 이 제안이 소프트웨어의 안전성 평가에 최선

의 방법이라고는 할 수 없을지 모르지만 시스템의 특성과 크기에 따라 가장 알맞은 방법이 동원되어야 한다는 것은 확실하다 할 것이다.

소프트웨어의 안전성 평가를 위한 결함 분석 기법과 관련하여 앞으로 관심을 가져야 겠다고 판단되는 과제로는

(1) 소프트웨어의 유형에 따른 효율적인 분석 기법의 적용 방안,

(2) 소프트웨어 안전성 평가에서 보다 효율적인 결함 분석 기법들의 결합에 대한 연구,

(3) 시스템의 안전성과 관련한 소프트웨어의 결함 분석 기법에 대한 연구도 필요하리라 생각된다.

참 고 문 헌

[1] Harold W. Lawson, "An Assessment Methodology for Safety Critical Computer Based Systems", Safety and Reliability of Software Based Systems, Twelfth Annual CSR Workshop, 1995

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

[3] Janusz Gorski, Bartosz Nowicki, "Object -Oriented Approach to Safety Analysis", Safety and Reliability of Software Based Systems, Twelfth Annual CSR Workshop, 1995

[4] Mark D. Hansen, Ronald L. Watts, "Software System Safety and Reliability", Proceedings Annual Reliability and Main tainability Symposium, 1988

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

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

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

[8] Peter L. Goddard, R. Davis, "Automated FMEA Techniques", Final Technical Report, RADC-TR-84-244, AD-154161, 1984

[9] Peter L. Goddard, "A Combines Analysis Approach To Assessing Requirements For Safety Critical Real-Time Control Systems", Proceeding Annual Reliability & Maintainability Symposium, 1996

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

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

[12] S.P. Wilson, T.P. Kelly, J.A. McDermid, "Safety Case Development: Current Practice, Future Prospects", Safety and Reliability of Software Based Systems, Twelfth Annual CSR Workshop, 1995

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

[14] Stephen S. Cha, Nancy G. Leveson, Timothy J. Shimeall, "Safety Verification in Murphy Using Fault Tree Analysis", Proceeding on the 10th International Conference on Software Engineering, 1988

[15] Stephen S. Cha, "Management Aspect of Software Safety", Proceeding on Inter national conference on Computer Assur ance, 1993

[16] 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