중소기업정보기술융합학회 논문지 제2권 제1호 pp. 11-18, 2012

안전 필수 시스템을 위한 안전성 분석 기법

김의섭, 윤상현, 유준범*
건국대학교 정보통신대학

A Survey on Safety Analysis Techniques for Safety-Critical Systems

Eui-Sub Kim, Sanghyun Yoon, Junbeom Yoo
College of Information and Communication, Konkuk University

요 약 소프트웨어의 규모가 커지고 복잡해지면서 소프트웨어의 기능적 실패(Functional Failure)를 만들어 내는 위험(Hazard) 요소들을 분석하기가 어려워지고 있다. 안전 필수 시스템(원자력 발전소, 항공관제 시스템, 철도 운영 시스템)에서 이런 기능적 실패는 곧 큰 사고 (인명피해, 환경오염) 로 이어지게 된다. 따라서 이러한 기능적 실패를 방지하고 소프트웨어의 안전성을 높이기 위해서는 소프트웨어 안전성 분석이 필요하다. 하지만 몇 가지 이유 (시간과 노력, 안전성 분석 기법의 다양한 지식 부족, 기관이나 회사의 관습적인 방법 사용) 로 적절하지 못한 안전성 분석 기법을 선택하게 되는 경우가 있다. 따라서 본 논문에서는 기존 안전성 분석 기법과 최신 안전성 분석 기법, 통합 모델 몇 가지를 소개 하겠다. 이것을 통해 전문가는 여러 종류의 안전성 기법을 간략하게 확인 할 수 있을 것이고, 시스템에 맞는 안전성 분석 기법을 선택하는데 도움을 받을 수 있을 것이다.

키워드 : 안전성 분석 기법, 안전 필수 시스템

Abstract As scale of software has been expanded and complicated, it is difficult to detect hazards which induce functional failure of software. Functional failure of safety-critical system (nuclear power plant, air traffic control systems, railway operating system) could result in a disaster (personal injury, environmental pollution). Therefore, it is necessary to conduct a safety analysis for preventing functional failure and increasing safety of the software. However, there are some reasons (time and effort problem, low knowledge of various safety analysis techniques, selecting conventional technique in company, organization) which disturb selecting an apposite one. This paper presents some traditional safety analysis techniques, recently presented techniques and combined models. We expect that it helps stakeholders to choice adequate one for target system.

Key Words : Safety analysis technique, Safety-Critical System

1. 서론

소프트웨어의 규모가 커지고, 복잡해지면서 소프트웨어의 기능적 실패(Failure)를 만들어 내는 위험(Hazard) 요소들을 분석하기가 어려워지고 있다. 소프트웨어가 일상생활뿐만 아니라, 안전성이 강조되는 분야(원자력 발전소, 항공관제 시스템, 철도 운영 시스템)에도 쓰이게 되면서, 이런 기능적 실패는 곧 큰 사고(accident)로 이어지

*교신저자(jbyoo@konkuk.ac.kr)
접수일(2012년 4월 2일), 심사완료일(2012년 6월 17일)

게 된다. 사고는 작게는 일상의 불편함이나 혼란과 같은 작은 피해를 가져오지만, 안전 필수 시스템(safety-critical system) 일 경우에는 인명 피해나 환경 오염처럼 재난과도 같은 결과를 가져 오게 된다[1]. 그렇기 때문에 소프트웨어의 기능적 실패를 제거하고, 안전성과 신뢰성을 높이기 위해 안전성 분석이 필요해 지고 있다. 또한, 시스템과 관련된 여러 규제 기관들도 시스템의 시행허가를 위해 안전성 분석을 요구하고 있다.

소프트웨어가 발전한 만큼 다양한 안전성 분석 기법들이 연구되어 왔다. 안전성 분석 기법의 적용은 많은 시간과 노력이 필요하고[1], 안전성 분석 기법들이 각각 대상으로 하는 개발 단계가 다르기 때문에 전문가(Stakeholder)는 이런 다양한 안전성 분석 기법 중에서 시스템에 맞는 적절한 안전성 기법을 신중히 선택해야 한다[2]. 그러나 안전성 분석 기법에 대해 다양하게 모르거나, 기관이나 회사에서 사용하는 방법을 관습적으로 계속 사용하기 때문에 적절하지 않은 안전성 분석 기법을 사용하는 경우가 많다.

그림 1과 그림 2는 적절하지 않은 기법의 사용을 보여준다. 그림 1, 2에서 원 A와 B는 안전성 분석 기법을 나타내고, 원의 크기는 안전성 분석 기법이 다룰 수 있는 범위와 그에 해당하는 비용을 나타낸다. 그림 1은 불필요한 고 비용의 기법 사용을 나타내고 있고, 그림 2는 소프트웨어에 대해 안전성을 충분히 만족하지 못한 기법의 사용을 나타내고 있다. 그림 2의 경우는 소프트웨어에서 특히 안전성이 더 필요한 부분을 포함하고 있는 기법 A를 사용하는 것이 B를 사용하는 것 보다 옳다는 것을 보여준다. 하지만 많은 전문가들은 위에 나열된 이유 때문에 적절한 안전성 기법을 사용하지 못한다.

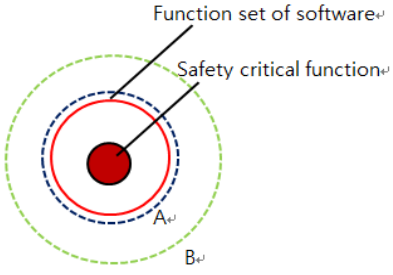

그림 1. 고 비용의 안전성 분석 기법 사용
Fig 1. Use of High-Cost Safety Analysis

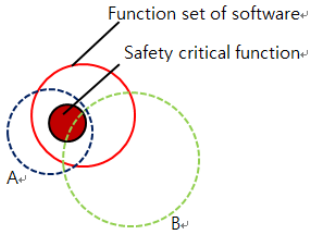

그림 2. 모든 기능에 대해 안전성을 만족하지 못한 기법 사용
Fig 2. Use of Unsatisfiable Safety Analysis

본 논문에서는 과거 안전성 분석 기법과 최신 안전성 분석 기법들의 특징을 설명한다. 여러 종류의 안전성 기법을 간략하게 확인할 수 있을 것이고, 이것을 통해 전문가는 시스템에 맞는 안전성 분석 기법 선택에 도움을 받을 수 있을 것이다.

용어 정의[3]
Error: 설계(design)의 결함 또는 바랐거나 의도했던 상태로 부터 벗어남
Hazard: 시스템 환경에서 다른 상태들과 연관되어 필연적으로 사고를 발생시키게 되는 시스템의 상태나 상태들의 모임
Failure: 특정 환경 상태에서 특정 시간동안 시스템 또는 컴포넌트(component)가 수행하기를 희망했던 기능의 불이행, 불능
Accident: 특정 수준의 손실이라는 결과를 내는 바랐거나 계획하지 않았던 이벤트

2. 안전성 분석 기법

2.1 FHA

Functional Hazard Assessment(FHA)는 failure 에 유발하는 기능(function)을 찾아내는 기법이다 [4, 5]. 개발 초기 또는 시스템을 정의하는 단계에서 적용 할 수 있으며, 하향식(Top-down)으로 분석을 반복한다. 브레인스토밍(Brainstorming)을 통해 기능과 관련된 위험을 정의하고, 위험이 미칠 영향, 영향의 심각성을 정의한다. 이런 과정을 통해 사용자는 안전성 목표(Safety objective)를 얻을 수 있다.

○ Approach: 안전성 목표를 통해 시스템이 safe하기 위해 필요한 것들을 정의 할 수 있고, 이것을 기반으로 사용자는 safety한 개발을 진행할 수 있다.

○ Output: Function, Failure Condition(Hazard Description), Phase, Effects of failure Condition on aircraft/Crew), Classification, Reference to Supporting Model, Verification 등의 속성을 포함하는 테이블이 생성된다.

2.2 PHA

PHA(Preliminary Hazard Analysis)는 요구사항 분석의 늦은 단계나, 설계 과정의 이른 단계에서 사용하기 좋다. 이 기법도 브레인스토밍방법을 사용한다. 따라서 시스템을 잘 이해하고 있는 전문가나, 안전성 분석 기법 경험자가 사용하기에 적합하다. PHA과정에서는 안전성 분석을 위한 체크리스트를 사용한다.

○ 체크 리스트: Life-cycle 모든 단계에서 사용되며, 사용자가 직접 손으로 작성한다. 실수(mistake), 교훈(lessen)을 적음으로써 피드백을 제공한다.

○ Output: 테이블 형태의 문서를 제공하고, 테이블 에는 Function, Failure Condition, Phase, Effect, Class, Verification 등의 속성이 있다.

○ 단계: (1) 위험으로 이어질 시스템 요소나 이벤트를 식별한다. (2) 사고로 이어질 위험의 이벤트 sequence 고려한다. (3) 사고의 결과와 수정 방법을 고려한다.

위와 같은 방법론으로 각각의 시스템 구성 컴포넌트의 동작 또는 고장(failure)이 시스템의 safety에 어떻게 영향을 미치는지 조사하는 **SHA**(System Hazard Analysis), 컴포넌트 동작의 조합 또는 고장 모드(mode)가 시스템의 safety에 영향을 미치는지 조사하는 **SSHA** (Subsystem Hazard Analysis), 사람과 기계의 인터페이스에 있는 위험을 조사하는 **OSHA** (Operating and Support Hazard Analysis) 등도 있다.

2.3 FMEA

Failure Mode and Effect Analysis(FMEA)는 존재하고 있는 또는 잠재적인 고장, 문제, 오류들이 사용자에게 닿기 전에 찾고(identify), 정의하고(define), 제거하는(eliminate) 안전성 분석 기법이다[6]. FMEA의 단계는 (1) 모든 컴포넌트를 리스트 형태로 정의를 하고, (2) 각각의 고장 모드가 영향을 미칠 모든 컴포넌트, 시스템을 정의 하고, (3) 각각의 고장 모드의 가능성과 심각성을 계산하는 것이다[3].

소프트웨어 개발 수명주기의 이른 단계에 이 분석 기법을 사용하는 것이 비용 측면에서 유리하다. 단계에서의 적용은 경고(warning) 또는 예방(preventive)을 통해 safety를 위한 설계 수정을 최소화 할 수 있다.

○ Output: 테이블 형태의 문서를 제공하고, 테이블 에는 컴포넌트, 고장 가능성, 고장 모드, 모드별 고장, 고장 효과 등의 속성이 있다.

2.4 FSD

FMEA는 컴포넌트의 고장을 확인하기에 아주 좋은 분석기법이다. 하지만 컴포넌트 의 상호작용에 대해서는 설명하지 못한다. 그래서 이런 FMEA의 단점을 보완하기 위해 UML의 Sequence diagram을 사용한 Failure Sequence Diagram(FSD)[7]이 제시되었다. FSD은 FMEA의 조사 단계가 끝난 후 수행되고, 컴포넌트 간의 상호작용을 보여주기 위해 UML Sequence diagram을 작성한다. 이것을 통해 현재의 제어 상태와 참조하고 있는 행동(Action)에 대해서 기술할 수 있다. 그림 3을 통해 Hazard Actor에서 발생된 입력이 컴포넌트들을 거치면서 사용자에게 어떻게 고장의 형태로 전달되는지를 그 과정을 볼 수 있다.

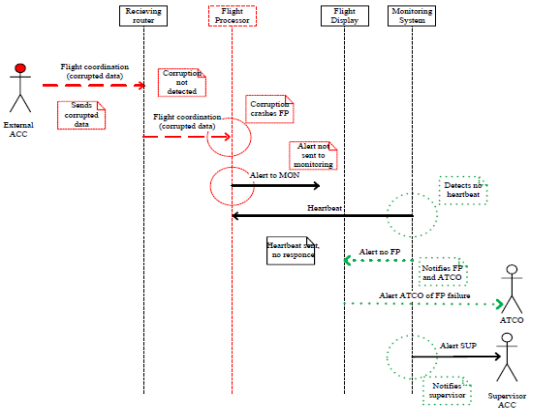

그림 3. FSD의 예
Fig 3. Example of FSD

2.5 HAZOP

HAZard and OPerability Analysis(HAZOP)은 이름에서 알 수 있듯이 위험과 시스템의 운영상의 위협 요소를 조사하는 기법이다. 전통적으로 성공한, 널리 알려진 방법 중에 하나이다. 기본적으로 시스템을 검토하고 잠재적인 위험을 찾는 것이 목적이고, 브레인스토밍 단계에서 Guide word[8]를 이용한다는 특징이 있다.

○ Approach: 설계와 운영성에서의 차이(편차)로 인해 사고는 발생된다. 따라서 이런 편차를 발생시킬 수 있는 설계나, 편차와 연관된 모든 위험을 찾는 것이 목적이다.

편차 즉, less flow, more flow, pressure와 같은 요소를 밝히기 위해 파라메터(온도, 압력 등)로 구성된 Guidewords(NO, NOT, NONE, MORE, LESS, PART OF)를 사용한다. 표 1 은 Guidewords와 이들의 의미를 보여준다.

○ Output: Guideword, Deviation, Possible Causes, Possible Consequences 등의 속성을 포함하는 테이블 형태이다.

HAZOP은 위험을 찾아내는 것뿐만 아니라 운영상에서의 문제까지 성공적으로 찾아내었기 때문에 자연스럽게 여러 종류의 분야(의학 진단 시스템, 도로 안전성 측정, 광 발전 시설의 위험 분석)로 확장 되고 있다[9].

표 1. Guidewords의 예
Table 1. Example of Guidewords

Guideword	Meaning
NO, NOT, NONE	의도된 결과를 얻지 못하였다. 하지만 아무 일도 일어나지 않았다.
MORE (LESS)	의도한 것보다 실제 수치가 더 많이(적게) 나왔다. (higher pressure, higher temperature)
REVERSE	의도한 것과 논리적으로 반대의 것이 발생 되었다. (forward flow 대신 backflow 발생)

2.6 FTA

Fault Tree Analysis(FTA)[10]는 시스템의 기능과 고장에 대한 정보를 그림 4와 같이 트리(tree) 구조로 제공한다. 현재 항공, 전자공학, 원자력 등 다양한 분야에서 사용 되고 있다. 개발의 모든 단계에서 사용 가능하다는 장점이 있지만, 지금까지 언급한 기법들과 달리 FTA는 위험이 발생할 원인들을 분석할 수는 있지만, 위험을 찾아내는 기법은 아니다.

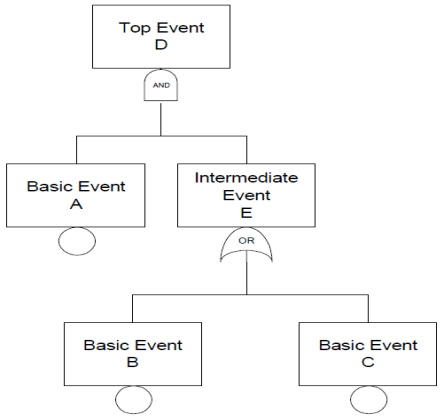

그림 4. FTA의 예
Fig 4. Example of FTA

○ Approach: 최상위 수준(root)에 의도하지 않은 이벤트를 두고, 이를 발생시킬 수 있는 잠재적인 faulty event 또는 normal mode를 노드(node)로 표현하고, 이것을 Boolean logic(AND, OR)을 사용해 조합하여 비쥬얼하게 보여준다.

2.7 TFT

FTA는 시간의 변화에 따른 조건을을 표현하지 못한다는 단점이 있다. 이것은 실시간 시스템에서는 문제가 된다. 따라서 FTA에 Temporal Gates를 추가한 TFT가 제시 되었다. Temporal Fault Tree(TFT)[11]는 FTA에 결함과 이벤트 사이의 임시적인 의존성(temporal dependency)을 Temporal Gate를 통해 표현할 수 있다.

그림 5는 PREV의 Gate 형태의 표현과 현재 시간과의 관계를 보여주고 있다. PREV는 현재 시간보다 p 시간 전을 표현한다. 그림 6은 PREV 2 에 해당하는 non-TFT의 모습이다.

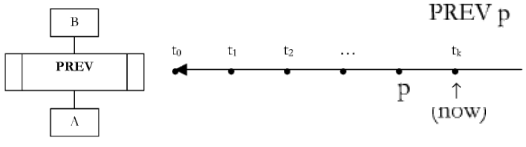

그림 5. PREV에 관한 temporal gate와 시간 표현
Fig 5. Temporal gate of PREV and express of time

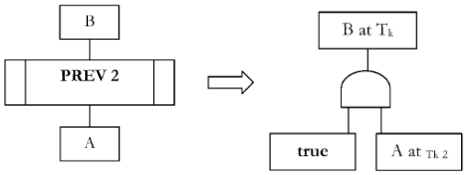

그림 6. PREV에 temporal gate와 non-TFT
Fig 6. Temporal gate of PREV and replacement

2.8 STAMP

System-Theoretic Accident Model and Processes (STAMP)[14]은 단순히 컴포넌트 고장 뿐만 아니라 컴포넌트간의 상호작용도 포함하여 고장의 원인을 분석한다.

○ Approach: 과거 안전성 분석 기법들은 현재 새로운 시스템에 잘 맞지 않는다 - 빠른 속도의 기술 변화, 새로운 위험 발생, 사람과 기계 사이의 관계가 복잡해짐 등. 따라서 새로운 관점의 안전성 분석 기법이 필요하게 되었다. 사고의 원인을 사람과 환경, 사회 까지 폭 넓게 생각한다. STAMP의 기본적인 개념은 이벤트가 아니라 제약사항을 중심으로 보는 것이다. 즉, Safety constraint가 제대로 지켜지지 못했기 때문에 손실로 이어지는 이벤트가 발생하는 것으로 볼 수 있다. 현재 다양한 분야에서 STAMP의 적용이 활발히 이루어지고 있는 중이다 [13, 14].

2.9 STPA

System-Theoretic Process Analysis (STPA) [14]는 다른 기법에서는 다루지 않은, STAMP로부터 정의된 새로운 결함 원인 인자(causal factor)를 사용하는 것이다. 전자기적, 기계적 특성을 갖는 컴포넌트에만 초점을 맞추었던 기존의 방법들과는 달리, 설계 오류, 소프트웨어 결함, 컴포넌트 간의 상호작용에 의한 사고, 그리고 사람, 기관, 관리 요소들이 만든 오류 등과 같은 새로운 원인들까지 관심을 가지고 있다.

STPA 기법의 목표는 (1) 사람, 기관, 환경 등의 전체적인 것을 포함하는 사고 시나리오를 만든다. (2) HAZOP과 마찬가지로 Guideword를 사용해 사용자에게 가이드를 제공한다. (3) 설계단계에서 Safety를 위해 필요한 정보를 개발자에게 제공해 Safety한 설계를 도와준다.

이 기법은 시스템의 전체 수명주기 단계에서 사용할 수 있지만, 설계가 만들어지기 전에 사용하는 것이 가장 비용 효과적이다.

○ 단계: (1) 기본적인 사항을 작성한다(accident, hazard, constraints, high-level의 safety control 구조). (2) 잠재적인 unsafe control action을 식별한다. (3) hazardous control action이 어떻게 발생하게 되었는지 (4) 시스템 위험을 상위수준에서의 Safety 요구사항으로 변환한다. (5) 상위 수준의 Safety 제약사항을 컴포넌트 수준의 상세 Safety 요구사항으로 재정의 하고, 손실에 대한 시나리오를 작성한다. (6) 결과를 이용하여 시스템 설계를 작성하고, Safety를 향상시키는데 반영한다.

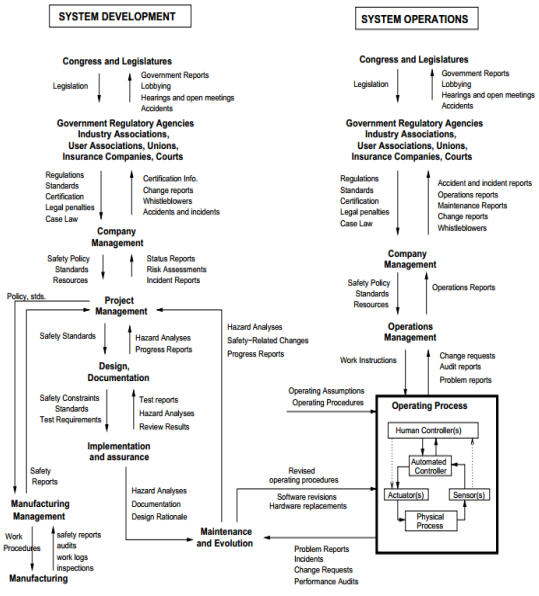

그림 7. Socio-Technical Control의 일반적인 모델
Fig 7. General Form of a Model of Socio-Technical Control

3. 안전성 분석 기법 기반 확장 모델

3.1 CHASSIS

Combined Harm Assessment of Safety and Security for Information System(CHASSIS) [15]은 Safety와

Security 두 가지 측면에 초점을 맞추고 있다. 이 두 속성을 모두 만족하고 싶은 사용자에게 유용하게 사용 될 수 있는 확장 모델이다. Security 측면은 Misuse Case (MUC), Misuse Sequence Diagram(MUSD)을 사용하였고, Safety 측면은 MUC와 Failure Sequence Diagram(FSM)을 사용하였다. 그리고 기본 바탕으로는 HAZOP을 사용한 안전성 분석기법을 기반으로 하는 통합 모델이다.

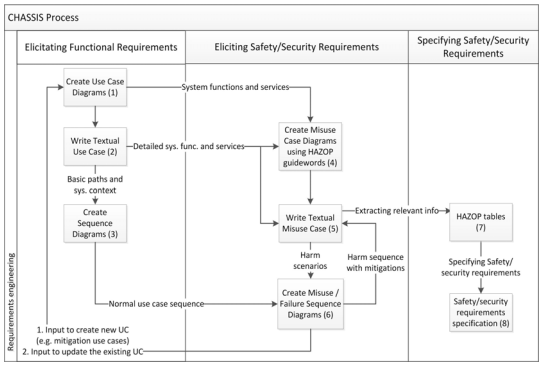

그림 8. CHASSIS의 주요 과정과 기법들
Fig 8. Main activities and Techniques of CHASSIS

○ Approach: Safety와 Security를 통합적으로 만족하고 위험과 고장을 보다 시각적으로 보여주기 위해 UML의 개념을 사용한 통합 모델이다.

단점은 CHASSIS는 위험을 찾고, 조사하고, 완화시키지만 리스크를 관리할 수는 없다.

그림 8은 CHASSIS의 주된 과정과 기법을 보여준다. 이 과정은 (1) 시스템 기능과 서비스를 정의한다.(UML을 이용해 Use Case diagram [step 1]과 textual use case[step 2], sequence diagram[step 3]을 작성하게 된다.) (2) Safety 와 Security의 요구사항을 얻는다. MUC 다이어그램을 통해 misusers, harm use case를 찾는다[step 4]. HAZOP의 Guideword를 통해 체계적으로 misuse case를 찾을 수 있다. MUC 다이어그램을 작성한다[step 5]. FSD, MUSD를 이용하여 사용자는 위험 시나리오를 정의 할 수 있다. [Step 6] (3) HAZOP 테이블을 얻는다[Step 7]. 이를 통해 Safety, Security 요구사항을 얻을 수 있다[Step 8].

3.2 HiP-HOPS

Hierarchically Performed Hazard Origin and Propagation Studies(HiP-HOPS)[2]은 하드웨어나 소프트웨어의 실행 단계인 함수 수준으로부터 안전성에 대한 정보를 통합 분석할 수 있도록 하는 계층적 통합기법이다.

이 기법에서는 Functional Failure Analysis (FFA), Interface focused-FEMA(IF-FAME), FTA를 사용하고 있다.

FFA는 설계를 시작하는 단계에서 사용된다. 이 단계에서 기능 고장을 찾을 수 있고, FFA의 결과는 설계를 구성하는데 Safety 요소 강화에 도움을 준다. 설계의 분할 단계에서 시스템은 하위 시스템, 컴포넌트로 나뉘게 되고, 하위 시스템, 컴포넌트는 Interface Focused-FMEA를 통해 컴포넌트 고장을 분석을 하게 된다. 이것을 통해 지역 고장 모델을 정의하게 된다.

HiP-HOPS는 기존 FFA의 단점(단일의 기능 고장만 확인)을 보완하기 위하여 탐지 및 복구 기능을 추가, 확장한 FFA를 제공한다. 그리고 FMEA의 단점(컴포넌트 사이의 고장 발견)을 보완하기 위해 IF-FMEA를 사용하였다. 또한 결함 트리를 자동적으로 생성하게 하는 도구도 제공하고 있다.

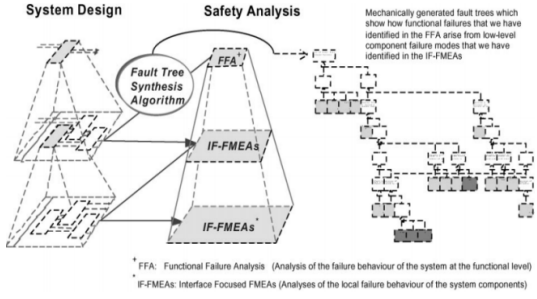

그림 9. HiP-POPS의 safety analysis와 design 개요
Fig 9. Design & safety analysis overview in HiP-HOPS

○ Output: 시스템의 실행적인 측면의 자세한 사항을 기록한 계층 모델을 얻게 된다.

HiP-POPS에서는 시스템의 구조와 컴포넌트의 지역적 고장 동작(IF-FMEA)을 알고 있다면 시스템의 고장 동작에 대한 정보(결함 트리)를 자동적으로 얻을 수 있는 도구를 제공하고 있다. 이 결함 트리를 통해서 시스템의

취약한 부분을 찾을 수 있다.

4. 결론

소프트웨어는 많은 발전을 이루어 왔다. 규모가 커지고, 복잡해지고, 사용 분야도 일상과 밀접한 곳부터 안전필수 시스템까지 다양한 곳에 쓰이게 되었다. 특히 안전필수 시스템에서는 안전성을 위해 안전성 분석 기법의 적용이 필수적이다.

하지만 안전성을 분석 기법의 적용은 많은 시간과 노력이 필요하고, 안전성 분석 기법들이 각각 대상으로 하는 개발 단계가 다르기 때문에 전문가는 이런 다양한 안전성 분석 기법 중에서 시스템에 맞는 적절한 안전성 기법을 신중히 선택해야 한다.

현재 다양한 안전성 분석 기법들이 제시되어 왔으며, 새로운 안전성 분석 기법들도 계속해서 제시되고 있다. 이런 것들을 모두 배우고 조사하는 것 역시 많은 시간과 노력이 필요하다. 본 논문은 이런 시간과 노력을 줄여주기 위해 전통적으로 많이 쓰였던 안전성 분석 기법과 최신기술, 통합 모델의 개요를 나열 하였다. 이것을 통해 전문가는 시스템에 맞는 안전성 분석 기법을 선택하는데 도움을 받을 수 있을 것이다.

Acknowledgement

"본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음"(NIPA-2012-(H0301-12-3004)). 또한 한국원자력연구원 주요사업 "원자력 계통 건전성 선진화 체계구축" 사업의 지원으로 연구한 결과입니다.

참 고 문 헌

[1] Daniel Jackson "Software for Dependable System: Sufficient Evidence?" National Academy of Sciences 2007

[2] Y. Papadopoulos "Analysis and synthesis of the behavior of complex programmable electronic system in conditions of failure" Reliability Engineering and System Safety 71 (2001), pp.229-247

[3] Nancy G. Leveson "SAFEWARE: System Safety and Computers" Addision-Wesley 1995

[4] EUROCONTROL "Guidelines for the identification of hazards, How to make unimaginable hazards imaginable?" NLR-CR-2004-094 March 2004

[5] P J Wilkinson "Functional Hazard Analysis for Highly Integrated Aerospace System" Certification of Ground/Air Systems Seminar (Ref. No. 1998/255), IEE

[6] D. H. Stamatis "Failure Mode Effect Analysis: FEMA from Theory to Execution" Second Edition 2003

[7] Christian Raspotnig "Supporting Failure Mode and Effect Analysis: A Case Study with Failure Sequence Diagram" International Journal of Secure Software Engineering (IJSSE), Volume 3, Issue 1. 2012. 17 pages.

[8] Giuseppe Mauri "Integrating Safety Analysis Techniques, Supporting Identification of Common Cause Failures", September 2000

[9] Jordi Dunjo "Hazard and Operability (HAZOP) analysis. A literature review" Journal of Hazardous Materials 173 (2010) 19-32

[10] Vesely WE. Fault tree handbook, US Nuclear Regulatory Committee Report NUREG-0492, US NRC, Washington DC, United States, 1981. p. X.15 - 8.

[11] Girish Keshav Palshikar "Temporal fault trees" Information and Software Technology 44 (2002) 137-150

[12] Haruka Nakao "Safety Guided Design of Crew Return Vehicle in Concept Design Phase Undesig STAMP/STPA" STAMP/ STPA Workshop, MIT, April 2012

[13] Safety Assurance in NextGen by Cody Fleming, Melissa Spencer, Nancy Leveson, and Chris Wilkinson (Honeywell). NASA Technical Report NASA/CR-2012-217553.

[14] Nancy G. Leveson "Engineering a Safer World" July 2009

[15] Christian Raspotnig "A combined process for elicitation and analysis of safety and security requirements" Not published.

저 자 소 개

김 의 섭(Eui-Sub Kim)

▪ 2012년 2월: 건국대학교 컴퓨터공학부 학사

<관심분야> : 소프트웨어 공학, 안전성분석, 정형기법

윤 상 현(Sanghyun Yoon) [비회원]

- 2010년 2월: 건국대학교 컴퓨터공학부 학사
- 2012년 2월: 건국대학교 컴퓨터공학부 석사
- 2012년 3월 ~ 현재: 건국대학교 컴퓨터공학부 박사과정

<관심분야> : 소프트웨어 공학, 안전성분석, 정형기법

유 준 범(Junbeom Yoo) [정회원]

- 2005년 8월: KAIST 전자전산학과 전산학전공 박사
- 2008년 2월: 삼성전자주식회사 통신연구소 책임연구원
- 2008년 3월 ~ 현재: 건국대학교 컴퓨터공학과 조교수

<관심분야> : 소프트웨어 공학, 안전성분석, 정형기법