http://dx.doi.org/10.5392/JKCA.2011.11.10.049

# 상태 전이 모델 기반 결함 트리 분석

## Fault Tree Analysis based on State-Transition Model

**정인상**
한성대학교 컴퓨터공학과

In-Sang Chung(insang@hansung.ac.kr)

**요약**

결함 트리 분석(Fault Tree Analysis)은 결함 트리를 구축하여 시스템의 안전성 분석을 수행한다. 그러나 결함 트리를 구성하는 작업은 대상 시스템의 도메인에 대한 지식과 경험을 필요로 하며 많은 시간과 노력을 소요한다. 이 논문에서는 시스템 설계 산출물인 상태 전이 모델을 기반으로 결함 트리를 체계적으로 구성하는 방법을 제안한다. 이를 위해 시스템 상태 전이 모델의 안정성 확보에 필요한 조건들을 식별하고 결함 트리를 구성할 수 있는 템플리트를 개발한다. 이 논문에서는 제안된 방법을 철도 건널목 제어 시스템에 적용한 결과도 기술한다.

■ 중심어 : | 결함 트리 분석 | 상태 전이 모델 | 안전성 분석 |

**Abstract**

Fault Tree Analysis(FTA) builds fault trees to perform safety analysis of systems. However, building fault trees depends on domain knowledge and expertize on target systems and consumes lots of time and efforts. In this paper, we propose a technique that builds fault trees systematically based on state-transition models which are software design artifacts. For the end, this paper identifies conditions that should be satisfied to guarantee safety of state-transition models and develop templates for fault tree construction. This paper also describes the results of appling the proposed method to railway crossing control system.

■ keyword : | Fault Tree Analysis | State-Transition Model | Safety Analysis |

## I. 서론

안전성이 확보되어야 하는 여러 분야에서 소프트웨어의 역할이 증대되고 있음에 따라 위험을 식별하고 제어하는 방법에 대한 연구가 활발하게 진행되고 있다. 이러한 방법들 중에서 결함 트리 분석(FTA, Fault Tree Analysis)은 1960년대에 Bell 연구소의 H. A. Watson[1]에 의해 고안되고, Boeing 항공사에 의해 보완되어 항공기 설계에 사용되기 시작되었다. 현재에 이르러서는 자동차, 화학 공정, 철도, 원자력 및 로봇 산업에 이르기까지 응용 분야가 급속하게 넓어지고 있는 추세이다[2][3].

FTA에 드는 시간과 비용은 연루된 시스템의 복잡성에 따라 달라진다. 소규모의 단순한 시스템에 대해서는 1주로도 충분할 수 있지만 크고 복잡한 시스템의 경우에는 몇 주 또는 몇 달이 소요될 수 있다. 이는 FTA의

* 본 연구는 한성대학교 교내연구과제로 수행되었습니다.
접수번호 : #110727-006
접수일자 : 2011년 07월 27일
심사완료일 : 2011년 09월 20일
교신저자 : 정인상, e-mail : insang@hansung.ac.kr

핵심 요소인 결함 트리를 구축하는데 어떤 체계적인 방법을 사용하는 것보다도 대부분 과거의 경험이나 직관에 바탕을 두고 있기 때문이다[4].

특히 시스템의 위험이나 고장을 야기하는 상태가 여러 객체들의 정상적인 상태들이 결합되어 나타나는 경우에는 결함 트리를 구성하는데 어려움이 있다[2]. 이러한 경우는 여러 객체들이 병행적으로 실행되는 분산 시스템에서 흔히 발견 될 수 있다. 예를 들어, 철도 건널목 제어 시스템(railway crossing control system)에서 기차가 건널목을 건너고 있을 때('crossing') 게이트가 열려('open')있는 경우를 생각해보자. 기차가 건널목을 지나고 있는지는 트랙에 부착되어 있는 센서들을 통해 알 수 있다고 가정할 때 이 위험 상태(hazard)는 두 객체 즉, 트랙과 게이트의 정상적인 상태가 결합되었을 때 발생된다: 'crossing∧open'. 이 경우에 일반적인 결함 트리 구성 방법은 위험 상태를 구성하는 각 객체의 상태에 대해 원인이 되는 사건들을 독립적으로 식별하는 것이다. 즉, 트랙이 'crossing' 상태를 유발하는 사건들을 식별하는 과정과 게이트가 'open'으로 이끌게 하는 사건들을 식별하는 과정을 독립적으로 수행한다. 이는 하드웨어에 대한 결함 트리를 구성하는 경우에는 별다른 문제가 없지만 소프트웨어 대해 결함 트리를 구성하는 경우에는 이들 상태가 독립적이지 않기 때문에 문제가 된다. 만약 독립적으로 수행된다면 기차를 결코 건널목을 지나지 않게 하거나('not crossing') 게이트가 절대 열리지 않게 하는('not open') 시스템을 만들 수도 있는 것이다.

이 논문에서는 대표적인 시스템 설계 산출물인 상태 전이 모델을 이용하여 체계적으로 결함 트리를 구성하는 방법을 제안한다. 상태 전이 모델은 시스템의 기능이 상태에 의존적일 경우에 시스템이 의도하는 기능을 기술하는 대표적인 수단이다. 특히, 안전성이 필수적으로 요구되는 내장형 시스템을 기술할 때 널리 사용되고 있다. 그러나 상태 전이 모델로부터 결함 트리를 구성하기 위해서 고려할 점이 있다. 상태 전이 모델은 시스템의 정상적인 행위를 기술한다는 점이다. 이는 시스템이 위험 상태에 이를 수 있는 사건들의 조합을 고려하는 결함 트리와는 다르다. 이 점을 고려하여 이 논문에서는 상태 전이 모델이 주어진 위험 상태에 관해 안정성을 확보하기 위해 만족해야하는 두 가지 조건('위험 회피 조건'과 '도달성 조건')을 식별한다. 제안된 방법에서는 이러한 조건들로 부터 안정성에 영향을 미칠 수 있는 요인들을 식별하여 결함 트리 템플리트를 개발하고 이를 이용하여 결함트리를 구성한다.

이 논문은 다음과 같이 구성된다. 2장에서는 FTA에 대해 간략하게 기술한 후 관련 연구를 소개한다. 3장에서는 이 논문에서 제안한 상태 전이 모델을 기반으로 결함 트리를 구성하는 방법에 대해 설명한다. 4장에서는 철도 건널목 제어 시스템에 제안된 방법을 사용하여 결함 트리를 구성하는 방법을 설명한다. 마지막으로 5장에서 결론 및 향후 연구에 대해 기술한다.

## II. 관련 연구

FTA는 시스템의 안정성을 해칠 수 있는 특정 시스템 상태나 주요한 시스템 고장이 주어지면 이를 야기할 수 있는 원인들의 조합을 결정하는 연역적인 방법이다[2][3]. FTA는 시스템 고장을 발생시키는 이벤트와 그의 원인들과의 인과관계를 논리기호(AND-게이트나 OR-게이트)를 사용하여 결함 트리를 만든다. 최소 절단 집합(minimum cut set)이라 불리는 시스템의 위험 상태를 유발시키는 필요 불가결한 기본 이벤트들의 집합들을 결함 트리를 이용하여 구할 수 있으며 이에 따라 시스템의 문제가 되는 부분들을 효과적으로 처리할 수 있다. [표 1]은 이 논문에서 사용되는 FTA를 위한 주요 기호를 정리한 표이다.

Leveson 등은 Ada 언어로 작성된 프로그램의 안전성 분석을 위해 FTA를 적용하였다[5]. 이를 위해 Ada 언어 구문 및 의미에 적합한 여러 결함 트리 템플리트를 제안하였다. 이 연구는 소프트웨어에 FTA를 적용하는 시발점이 되었다.

또한, FTA는 프로그램 코드를 대상으로 뿐만 아니라 UML의 클래스 다이어그램, 시퀀스 다이어그램 및 상태도 등과 같은 여러 설계 산출물에 대해서도 적용되어 왔다. [6]에서는 구현 단계를 포함한 소프트웨어 개발

사이클 동안에 FTA를 수행하였을 때 얻을 수 있는 여러 이점에 대해 기술하고 설계 단계의 산출물의 하나인 UML

**표 1. 결함 트리에 사용되는 여러 기호**

| | |
|---|---|
|  | 〈기본 이벤트〉 결함을 발생시킬 수 있는 가장 낮은 수준에서의 사건이며 더 이상 분석이 이루어지지 않는다. |
|  | 〈미개발 이벤트〉 현재 정보의 부족 등 여러 이유로 확장되지 않은 사건. 앞으로 분리된 결함 트리로 확장될 가능성이 있음 |
|  | 〈조건 이벤트〉 논리 게이트에 적용되는 특정 조건이나 제약 등을 기술 |
|  | 〈AND 게이트〉 모든 입력 결함들이 발생될 때 출력 결함이 발생 |
|  | 〈OR 게이트〉 하나 또는 그 이상의 입력 결함들이 발생될 때 출력 결함이 발생 |
|  | 〈금지 게이트〉 입력이 조건 이벤트에 의해 기술된 활성화 조건을 만족한 상태에서 발생하면 출력이 발생 |

시퀀스 다이어그램을 결함 트리로 변환하는 방법을 개략적으로 제안하였다.

FTA와 직접적인 관련은 없지만 시스템의 안정성 분석을 위해 여러 형태의 설계 산출물에 HAZOP(Hazard and Operability Studies)[7]을 적용하였다. 이 방법은 상태도나 클래스 다이어그램 등에 기술된 여러 시스템 요소들에 소위 'guideword'를 사용하여 시스템의 행위를 분석한다. 예를 들면 한 클래스의 속성에 대해 가져야하는 값보다 더 크거나 더 작은 값('More/Less')을 가질 때 시스템 행위에 안전성 측면에서 어떤 영향이 있는가를 분석하는 경우는 클래스의 속성에 'More/Less' 'guidword'를 적용한 경우에 해당한다. 그러나 이러한 방법은 시스템의 안전성에 대한 요구사항을 식별할 때 유용하게 사용될 수 있지만 시스템의 모든 요소들에 적용될 것을 요구하기 때문에 매우 작은 시스템이라 할지라도 기본적으로 시간이 많이 소요되는 단점이 있다.

[8]에서 Gorski와 Norwicki는 상태 모델을 기반으로 하는 안정성 분석 모델을 제안하였다. 이 방식에서는 시스템 객체들이 설계 모델에 기술된 대로 행동하지 않는다는 가정을 한다. 이를 반영하여 시스템의 신뢰성을 해칠 수 있는 객체의 행위를 분석하기 위해 시스템 상태 모델의 전이에 변형을 가져오는 여러 변형(mutation) 규칙 등을 제안하였다. 그러나 어떤 객체의 상태 전이 모델에 대해 이러한 변형 규칙을 수행하는지가 명확하지 않다[9]. 따라서 [7]의 방법과 마찬가지로 모든 객체들에 대해 변형 규칙을 수행하는 경우에는 매우 많은 시간과 노력이 소요된다.

최근에 [10]에서 결함 트리를 상태 모델로 변환하는 방법을 제안하였다. 이 방법의 목적은 결함 트리에 나타난 바람직하지 못한 시스템 상태나 이벤트들을 야기할 수 있는 (정상적인) 행위들의 조합을 상태 모델로 나타내는 것이다. 따라서 변환된 상태 모델은 시스템의 안정성을 테스트할 수 있는 테스트 케이스의 생성 기반으로 사용될 수 있다. 이 방법에서는 변환된 상태 모델을 생성할 때 사용되는 결함 트리가 시스템 명세로서 사용된 상태 모델을 바탕으로 구성된다는 점이다. 즉, 시스템 명세로 상태 모델이 주어져있을 때 이를 바탕으로 결함 트리를 만든다. 그리고 시스템의 안정성에 영향을 줄 수 있는 행위들의 조합들을 추출하기 위해 결함 트리를 다시 상태 모델로 변환한다. 그러나 [10]에서는 상태 모델로부터 어떻게 결함 트리를 구성하는지에 대해 기술하지 않았다.

## III. 상태 전이 모델 기반 결함 트리 구성

이 장에서는 상태 전이 모델로 기술된 시스템에 대해 결함 트리를 구성하는 방법에 대해 기술한다. 3.1절에서는 이 논문에서 가정하고 있는 시스템 상태 전이 모델을 기술하고 결함 트리 구성 절차에 필요한 용어들을 정의한다. 3.2절에서는 상태 전이 모델의 안정성에 관한 조건을 기술한다. 3.3 절에서는 3.2절에서 기술한 상태 전이 모델의 안정성 조건을 기반으로 결함 트리를 구성할 때 사용하는 결함 트리 템플리트를 개발한다. 또한 결함 트리 템플리트를 사용하여 주어진 위험 상태에 대한 시스템의 결함 트리를 구성하는 절차를 기술한다.

## 1. 상태 전이 모델

이 논문에서는 대상 시스템이 기본적으로 이벤트 발생 및 공유 메모리를 통해 상호 협력하는 여러 객체들로 구성되어 있다고 가정한다. 각 객체 $O_i$의 동적 행위는 상태 전이 모델 $STM_i$로 기술된다. 상태 전이 모델 $STM_i$는 4개의 요소 구성된 튜플 ($S_i$, $TL_i$, $T_i$, $s_{i0}$)로 정의되며 $S_i$, $TL_i$, $T_i$, $s_{i0}$는 각각 다음과 같다. $S_i$ 객체 $O_i$의 상태들의 집합이다. $TL_i$는 전이 라벨(transition label)들의 집합이며 각 전이 라벨 $tl_k \in TL_i$은 $tl_k$를 트리거하는 사건 $e_k$, $e_k$가 발생했을 때 만족해야 하는 부울리언 표현식(boolean expression) $g_k$, 그리고 $tl_k$이 수행될 때 실행되는 액션 ak로 구성된다. 즉, $tl_k = (e_k, g_k, a_k)$. 실제 상태 전이 모델에서는 $[g_k]e_k/a_k$로 표시되며 $e_k$, $g_k$, $a_k$ 각각은 $tl_k$를 기술할 때 반드시 나타날 필요가 없다. $T_i$는 전이들의 집합이며 각 전이 $t_k$는 $S_i \times TL_i \times S_i$의 한 멤버이다. 즉, $t_k \in S_i \times TL_i \times S_i$. 전이 $t_k=(s, tl_k, s')$에 대해 상태 s와 s′를 전이 $t_k$의 출발 상태 및 목적 상태라 하며 각각 $pred(t_k)=s$와 $succ(t_k)=s'$로 나타낸다. 마지막으로 $s_{i0}$는 객체 i의 초기 상태를 나타낸다. [그림 1]은 4장에서 설명할 철도 건널목 제어 시스템을 구성하는 게이트 객체와 트랙 객체에 대한 상태 모델을 보여준다. 상태 'open'과 'no train'은 게이트 객체와 트랙 객체의 초기 상태이다.

각 객체의 행위가 상태 전이 모델로 모델링 된 후에 시스템 전체에 대한 상태 공간 $SG=(S, T, s_0)$가 정의될 수 있다. 시스템을 구성하는 객체가 $O_1,...,O_n$이라 할 때 상태 공간의 각 상태 $s=(s(1), s(2), ..., S(n))$는 집합 $S_1 \times ... \times S_n$의 한 요소이다. 즉, 시스템 상태 $s \in S$는 시스템을 구성하는 객체의 상태들이 결합되어 정의되며 $i \in \{1,..., n\}$에 대해 $s(i) \in S_i$가 된다. 지금부터 사용의 편의를 위해 시스템 상태 s를 벡터로 간주하지 않고 집합으로 간주한다. 따라서 시스템 상태 s에 대해 $l \in s$는 다음을 의미한다: $\exists i, 1 \le i \le n$ s.t $l=s(i)$. 상태 공간 SG에서 전이 $t \in T$는 집합 $T_1 \cup ... \cup T_n$의 한 요소이다. 'enabled(s)'는 상태 s에서 활성화되는 전이들의 집합을 나타낸다. 만약 $t \in enabled(s)$라면 t는 실행될 수 있다.

만약 전이 $t=(s_m, t_k, s_p)$가 실행된다면 상태 $s=(s(1),...,s(i)=s_m,...,s(n))$으로부터 $s'=(s(1),..., s'(i)=s_p, ...,s(n))$으로 도달하며 $s \rightarrow^{t} s'$로 표시한다. 또한 $s \Rightarrow^{w} s'$는 유한한 전이 시퀀스 w를 통하여 s로부터 s′까지 도달할 수 있다는 사실을 나타낸다. 상태 $s_0$는 시스템의 초기상태이며 모든 객체의 초기상태의 조합

으로 정의된다. 즉, $s_0=(s_{10}, ..., s_{n0})$.이 논문에서는 상태 공간 $S_G$를 고려할 때 초기 상태 s0로부터 도달 가능한 상태들로 한정한다.

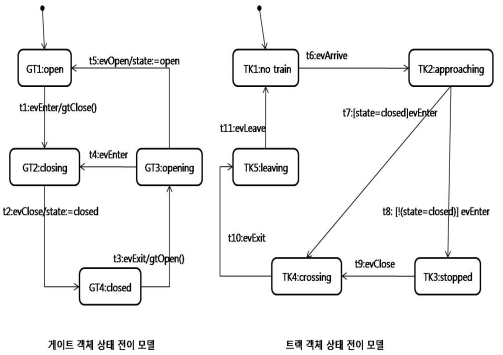


**그림 1. 객체의 상태 전이 모델**

또한 각 객체의 상태 모델에서처럼 SG의 각 시스템 상태 s에서 PRED(s)를 다음과 같이 정의할 수 있다: $PRED(s)=\{s' \mid \exists t \in T_i \bullet s=s' \backslash pred(t) \cup succ(t)\}$. SUCC(s)도 마찬가지 방식으로 정의할 있다.

## 2. 상태 전이 모델의 안전성

이 논문에서는 대상 시스템이 기본적으로 이벤트 발생 및 공유 메모리를 통해 상호 협력하는 여러 객체들로 구성되어 있다시스템의 상태 전이 모델이 안전성을 확보하기 위해서는 시스템이 초기 상태로부터 위험 상태로 도달 가능하지 않거나 위험 상태로 도달 가능하더라도 위험 상태로부터 즉시 빠져 나오는 수단이 제공되어야 한다. 이 절에서는 이를 바탕으로 상태 전이 모델의 안전성에 대해 명확하게 정의한다. 이를 위해 우선 '실행-의존적' 개념을 정의한다. 두 개의 전이 t와 t′는 다음 조건 모두를 만족하였을 때 상태 s′에 관해 실행-의존적(execution-dependent)이라고 한다:

- if $s \rightarrow^{t} s'$, then $t' \in enabled(s')$
- if $s' \rightarrow^{t} s''$, then $s' \neq s''$

즉, 두 개의 전이가 실행-의존적이기 위해서는 한 전이의 실행(t)이 다른 전이(t′)의 실행을 강제하여야 한다는 것을 의미한다. 강제 실행된 전이는 실행을 야기한 전이와 다른 목적 상태를 가져야 한다, i.e., $s' \neq s''$. 'EXD(s, t)'는 상태 s에 관해 전이 t에 실행-의존적인 전이들의 집합을 나타낸다.

시스템이 h에 관해 안전하기 위해서는 상태 공간 $S_G$에서 다음과 조건 중의 하나를 만족해야 한다:

C1) $\exists s' \in S \mid \text{if } s \rightarrow^{t} h \text{ then } t' \in EXD(h, t)$

C2) $\nexists w \mid s0 \Rightarrow^{w} h$

조건 C1)은 만약 위험 상태에 도달하였다면 즉시 위험 상태가 아닌 상태로 빠져 나오는 전이가 존재하여야 한다는 사실을 의미한다. 이 논문에서는 이 조건을 '위험 회피 조건(Hazard Escaping Condition)'이라 한다. 조건 C2)는 시스템의 초기 상태로부터 어떤 위험 상태로도 도달 가능하지 않아야 한다는 사실을 의미하며 '도달성 조건(Reachability Condition)'이라 한다. 다음 [정리 1]은 시스템이 위 두 조건을 만족하지 않았을 때 시스템이 위험 상태에 지속적으로 머물게 된다는 사실을 의미한다.

**정리 1.** 시스템 상태 공간 SG에서 위험 회피 조건과 도달성 조건을 만족하지 않는 상태를 h라 하자. 이 때 h는 데드락(deadlock) 상태이다.

**증명.** [정리 1]의 증명은 매우 간단하게 할 수 있으므로 생략한다.

## 3. 결함 트리 템플리트 및 구성 절차

결함 트리는 3.2절에서 기술한 시스템 모델의 안정성 조건을 위배하는 모든 가능한 요인들에 대한 분석을 통해 이루어진다. 이 논문에서는 이러한 분석을 통해 전이에 대한 결함 트리 템플리트(template)를 개발한다. 이렇게 개발된 결함 트리 템플리트는 시스템의 결함 트리를 직관이나 경험에 덜 의존하면서 체계적으로 구성할 수 있게 한다.

전이 t에 대한 결함 트리 템플리트는 다음 두 가지 고장 모드(failure mode)를 고려한다. 첫 번째 경우는 활성화 되어야 할 상태일 때 활성화 되지 않는 고장 모드이며 이를 'fail(en(s,t))'로 나타낸다. 즉, 상태 s에서 전이 t가 활성화되어야 하는데 활성화되지 않는 경우이다. 두 번째는 활성화되지 않아야 할 때 활성화 되는 고장 모드이며 이를 'fail(di(s,t))'로 나타낸다. [그림 2]와 [그림 3]은 이 두 고장 모드들에 대한 결함 트리 템플리트들을 보여준다. [표 2]는 결함 트리 템플리트에서 사용된 고장 모드들을 포함한 여러 용어들에 대해 기술한 것이다.

고장 모드 'fail(en(s,t))'는 템플리트는 전이 t=(e, g,-)가 활성화되어야 할 때 활성화 되지 않은 경우들을 고려하였다. 전이가 활성화되지 않기 위한 요인들로 이벤트와 관련된 고장('fail(evt(s,e))')과 가드와 관련된 고장('fail(guard(s,g))')로 구분하였다. 'fail(evt(s,e))'는 이벤트 e가 아예 발생하지 않거나 발생하였다 하더라도 해당 객체가 인식하지 못한 경우를 고려하였다. 가드 g에 관해서는 가드가 거짓이어야 하는 경우에 참으로 평가되는 잘못된 평가가 이루어지는 경우를 고려하였다. 또한 가드가 올바르지 않게 설정된 경우('fail(set(s′,t′,g))')와 올바르게 설정되었다 할지라도 시스템 실행 중에 값이 부당하게 변경되는 경우('fail(trans(s′,t′,g))')를 고려하여 템플리트에 반영하였다.

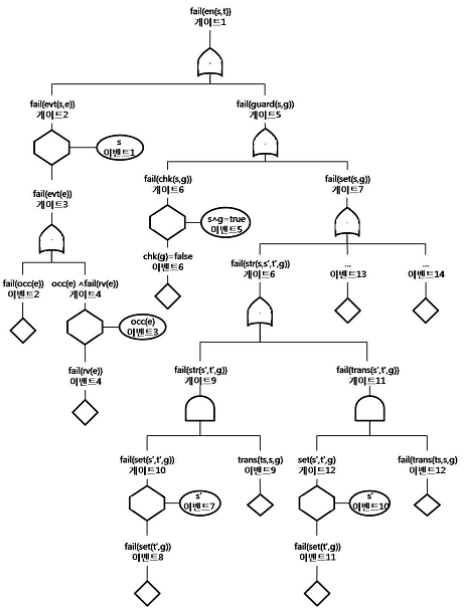


**그림 2. 고장 모드 'fail(en(s,t))' 에 대한 결함 트리 템플리트**

[그림 2]에서 게이트 9와 게이트 10의 s, s′, t의 관계는 다음과 같다: 전이 시퀀스 $ts=t_{i+1}t_{i+2}\cdots t_m$에 대해 $s'=s_{i-1}\rightarrow^{t'}s_i\Rightarrow^{ts}s_m=s$. 또한 전이 t′는 전이 t에 관해 의존적(dependent)이며 전이 시퀀스 ts의 모든 전이들은 전이 t와는 무관해야(independent) 한다. 전이간의 의존관계에 대해서는 [11]을 참조하기 바란다. 이러한 전이 시퀀스($ts'=t'\bullet ts$)를 자료 흐름 측면에서 전이 t′가 가드 g의 값을 정의하고 ts′는 g의 값을 전달하는 경로, 즉 'def-clear path'[12]로 간주할 수 있다. t와 t′ 사이에 이러한 전이 시퀀스, ts′가 존재할 때 "t′는 ts′를 통해 t에 영향을 미친다."라고 표현한다.

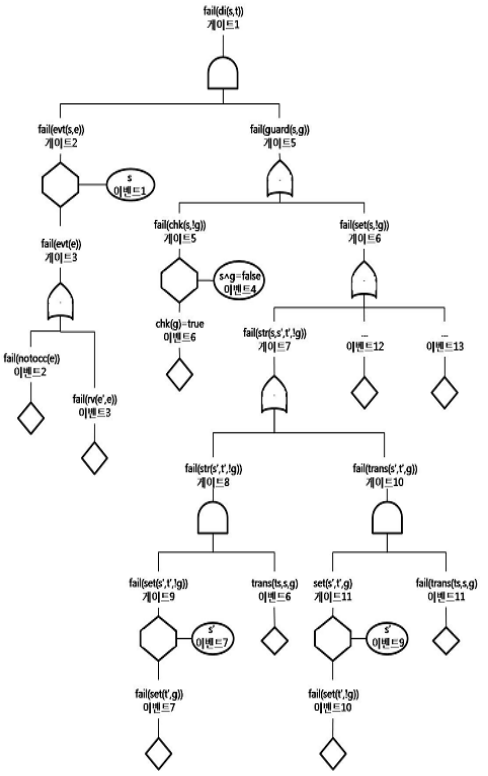


**그림 3. 고장 모드 'fail(di(s,t))' 에 대한 결함 트리 템플리트**

주어진 두 전이 사이에 이러한 값 전달 경로는 하나 이상 존재할 수 있기 때문에 모두 고려하는 것은 많은 비용이 소요될 수 있다. 이 때 자료 흐름 기반 테스트에서 사용되는 여러 테스트 적합성 기준[12]을 적용하여 선택적으로 일부 전이 시퀀스들만을 고려할 수 있다. 또한 전이 시퀀스뿐만 아니라 이러한 조건을 만족하는 전이도 하나 이상 존재할 수 있으므로 조건을 만족하는 각 전이에 대해 게이트 8을 루트로 하는 결함 트리와 동일한 방식으로 구성한다(이벤트 13 및 이벤트 14 등).

[그림 2]에 있는 고장 모드 'fail(di(s,t))'의 결함 트리 템플리트는 전이 t가 비활성화 되어야 할 때 활성화되는 경우들을 고려하여 구성한다. 이 논문에서는 'fail(di(s,t))' 구성 방법이 'fail(en(s,t))' 구성 방법과 매우 흡사 하기 때문에 이에 대한 자세한 설명은 생략한다.

[그림 2]와 [그림 3]에서 볼 수 있듯이 결함 트리 템플리트의 가장 바닥에 위치한 이벤트들은 모두 미개발 이벤트로 처리하였다. 그 이유는 상세 설계 정보 및 구현 정보를 이용할 수 있을 때 이를 기반으로 결함 트리가 더 확장될 수 있다는 의미이다.

예를 들면 [그림 2]의 이벤트6('chk(g)=false')은 실제 가드를 검사하는 모듈이 구현된 후에 구현 모듈의 해당 부분을 분석하여 확장될 수 있다. 이 때 [5]에서 제안한 것처럼 구현에 사용된 프로그래밍 언어 및 구문에 따른 결함 트리 템플리트를 사용할 수 있을 것이다.

[그림 4]는 전이에 대한 고장 모드 템플리트를 이용하여 결함 트리를 구성하는 절차를 보여준다. 우선 위험 상태 h를 가장 상위 이벤트로 하는 결함 트리를 [그림 5]와 같이 구성한다(라인 1-2). 이 때 h를 야기할 수 있는 경로들을 분석하기 위해 h의 바로 이전 상태 s($s\in PRED(h)$)들에 대한 분석을 진행한다(라인 3). 라인 3-9는 전이 t가 위험 회피 조건을 만족하는지 검사하여 만족한다면 위험 회피를 강제하는 전이 t′에 대해 고장 모드 'fail(en(s,t′))'에 관해 결함 트리를 구성한다. 즉, t′가 활성화 되어야 할 때 활성화 될 수 없는 여러 원인들에 대한 분석을 진행한다.

라인 10은 상태 s에서 위험 회피 조건을 만족하지 않은 전이가 활성화될 수 있는지를 검사한다. 만약 이러한 전이가 존재한다면 s를 통해 h에 도달할 수 있는 경로 상에 위험 회피 조건이나 도달성 조건을 만족하는 상태가 존재하여야 한다. 이는 시스템 모델이 h에 관해 안전성을 보장되어야 하기 때문이다. 따라서 이 경우에는 상태 s에 관해 재귀적으로 [그림 4]의 절차를 수행한다.

```
PROCEDURE CONST_FT(h) begin
1: h를 가장 상위 이벤트로 하는 결함 트리를
2: [그림 5]와 같이 구성
3: for each s in PRED(h) do
4:   if (t∈enabled(s)) then
5:          if (∃t′∈EXD(h,t)) then
6:         'fail(en(s, t′))'를 가장 상위 사건으로
7:            설정한 결함 트리를 [그림 2]와 같이 구성한
8:         후에 이를 [그림 5]의 or-게이트의
9:         한 입력으로 제공한다.
10:      else CONST_FT(s)
11:  else
12:    'fail(dis(s,t))'를 가장 상위 사건으로 설정한
13:    결함 트리는 [그림 3]과 같이 구성한 후에
14:    이를 f1이라한다.
15:    if (∃t′∈EXD(h,t)) then
16:          'fail(en(h, t′))'를 가장 상위 사건으로 설정
17:        한 결함 트리를 [그림 2]와 같이 구성한 후
18:         이를 f2라 한다.  f1과 f2를 입력으로
19:         하는  AND-게이트를 [그림 5]의 or-게이트의
20:      한 입력으로 제공한다.
END PROCEDURE
```

**그림 4. 결함 트리 구성 절차**

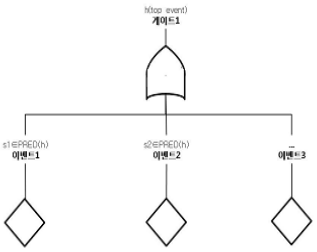


**그림 5. 결함 트리 상위 레벨**

라인 12-14는 전이 t가 도달성 조건을 만족하는지 검사하여 만족한다면 전이 t에 대해 고장 모드 'fail(di(s,t))'에 관해 결함 트리를 구성한다. 즉, 전이 t가 활성화 되는 경우에 h로 도달가능하다. t가 활성화 될 수 있는 요인들을 고장 모드 'fail(di(s,t))'에 대한 결함 트리를 구성하여 분석을 진행한다. 여기에서 주의할 점은 전이 t가 활성화된다는 가정에서 분석이 진행되기 때문에 t를 통해 위험 상태 h에 도달하였을 때 위험 회피 조건을 만족하는 전이 t′가 존재할 수 있다. 이 경우를 위해 위험 회피를 강제하는 전이 t′에 대해 고장 모드 'fail(en(s,t′))'에 관해 결함 트리를 구성하여 라인 12-14에서 구성한 결함 트리와 AND-게이트로 결합한다. 이 과정이 라인 15-20에 나타나 있다.

**표 2. 고장 모드 및 관련 용어 정의**

| | |
|---|---|
| fail(en(s,t)) | 상태 s에서 활성화되어야 하는 전이 t가 활성화 되지 않는 고장 모드 |
| fail(di(s,t)) | 상태 s에서 비활성화 되어야 하는 전이 t가 활성화 되는 고장 모드 |
| fail(evt(s, e)) | 상태 s에서 이벤트 e에 관련된 고장 모드 |
| fail(evt(e)) | 이벤트 e에 관련된 고장 모드 |
| fail(occ(e)) | 발생해하는 이벤트 e가 발생하지 않는 고장 모드 |
| fail(nottocc(e)) | 발생하지 않아야 하는 이벤트 e가 발생하는 고장 모드 |
| fail(rv(e',e)) | 이벤트 e'의 발생을 이벤트 e의 발생으로 인식하는 고장 모드 |
| fail(guard(s,g)) | 상태 s에서 가드 g에 관련된 고장 모드 |
| fail(chk(s,g)) | 상태 s에서 가드 g의 검사에 관련된 고장 모드 |
| chk(g) | 가드 g를 검사한 결과 |
| fail(set(s, g)) | 상태 s에서 가드 g의 값에 대한 고장 모드 |
| fail(str(s,s',t',g)) | fail(set(s', t', g)) 이 거나 fail(trans(s',t',g)) [아래 참조] |
| fail(set(s',t',g)) | 상태 s'에서 전이 t'에 의해 가드 g의 값이 잘못 설정되는 고장 모드 |
| fail(set(t',g)) | 전이 t'에 의해 가드 g의 값이 잘못 설정되는 고장 모드 |
| set(s',t',g) | 상태 s'에서 전이 t'에 의해 가드 g의 값이 설정 |
| fail(trans(s',t',g)) | 상태 s'에서 전이 t'에 의해 설정된 가드 g의 값이 잘못 전달되는 고장 모드 |
| trans(ts,s,g) | 전이 시퀀스 ts에 의해 가드 g의 값의 변화 없이 상태 s로 전달 |
| fail(trans(ts,s,g)) | 전이 시퀀스 ts에 의해 가드 g의 값이 상태 s로 잘못 전달되는 고장 모드 |

## IV. 사례 연구

이 절에서는 제안된 결함 트리 구성 방법을 사용하여 철도 건널목 제어 시스템에 적용하는 과정을 기술한다. 철도 건널목 제어 시스템의 트랙 객체와 게이트 객체의 상태 전이 모델을 보여 준다. 트랙에는 곳곳에 센서가 장치되어 있어 기차가 건널목 접근('evArrive')/진입('evEnter')/진출('evExit')/통과('evLeave') 여부를 알 수 있다. 또한 게이트에도 센서가 장착되어 있어 게이트의 개폐 여부를 알 수 있다. 게이트가 열릴 때 이벤트

'evOpen'이 발생하고 닫힐 경우에는 이벤트 'evClose'가 발생한다. 철도 건널목 제어 시스템의 주요 목적은 충돌을 방지하는 것이다. 즉, 기차가 건널목을 지나가고 있을 때 게이트가 닫혀 있어야 한다. 따라서 시스템은 게이트가 개방되어 있을 때 기차가 건널목을 지나가는 상태('open∧crossing')에 도달되지 않도록 하거나 도달되더라도 즉시 다른 안전한 상태로 전이가 되는 것을 보장하여야 한다.

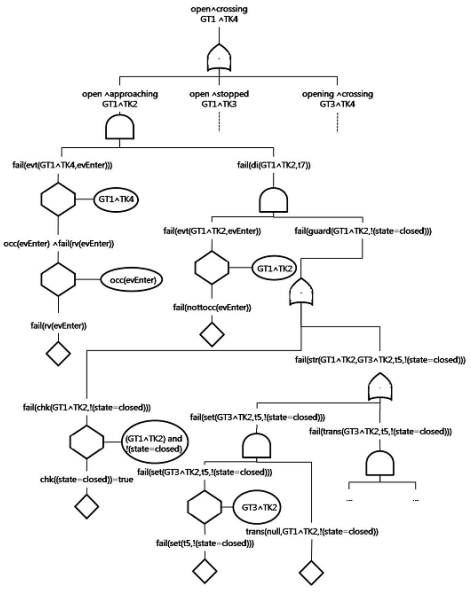


**그림 6. 철도 건널목 제어 시스템의 결함 트리**

[그림 6]은 철도 건널목 제어 시스템에 대해 'open∧crossing'의 선행 상태 중의 하나인 'open∧approaching'에 대해서만 [그림 4]의 결함 트리 절차를 적용하여 구성한 결함 트리의 일부를 보여준다. 전이 t7('[state=closed] evEnter')에서 현재 게이트 객체가 'open' 상태에 있기 때문에 'open∧approaching'에서 가드 조건 '[state=closed]'이 만족되지 않아 활성화되지 않는다. 즉, t7∈enabled (open∧approaching). 따라서 [그림 4]의 라인 9-14에 따라 결함 트리를 구성하게 된다.

우선 t7에 대해 고장 모드 'fail(di(open∧approaching, t7)'에 대한 결함 트리를 [그림 3]의 템플리트를 기반으로 구성을 진행한다. 현재 고려해야 하는 전이는 t1과 t7이다.

전이 t7은 이벤트 'evEnter'와 가드 '[state=closed]'로 구성되어 있다. 따라서 고장 모드 'fail(di(open∧approaching,t7)'에 대한 결함 트리는 이벤트 'evEnter'와 가드 '[state=closed]'에 대한 결함 트리를 구성하여 AND-게이트로 결합한다. 이벤트 'evEnter'에 대한 결함 트리는 [그림 3]에서 볼 수 있듯이 이벤트 'evEnter'가 발생하지 않아야 할 때 발생하는 고장 모드, 즉 'fail(nottocc(evEnter))'나 'evEnter'가 아닌 이벤트 e′의 발생을 'evEnter'의 발생으로 인식하는 고장 모드 'fail(rv(e′, evEnter))'를 고려하여 구성한다.

전이 t7은 이벤트 'evEnter'와 가드 '[state=closed]'로 구성되어 있다. 따라서 고장 모드 'fail(di(open∧approaching,t7)'에 대한 결함 트리는 이벤트 'evEnter'와 가드 '[state=closed]'에 대한 결함 트리를 구성하여 AND-게이트로 결합한다. 이벤트 'evEnter'에 대한 결함 트리는 [그림 3]에서 볼 수 있듯이 이벤트 'evEnter'가 발생하지 않아야 할 때 발생하는 고장 모드, 즉 'fail(nottocc(evEnter))'나 'evEnter'가 아닌 이벤트 e′의 발생을 'evEnter'의 발생으로 인식하는 고장 모드 'fail(rv(e′, evEnter))'를 고려하여 구성한다.

[그림 6]에서 볼 수 있듯이 고장 모드 'fail(rv(e′, evEnter))'에 대해서는 결함 트리를 구성할 때 고려하지 않았다. 그 이유는 전이 t1과 t7이 상태 'open∧crossing'에 대해 실행 의존적이기 때문이다. 즉, t1∈EXD(open∧crossing,t7). 이는 t7이 활성화되어 실행된다면 전이 t1이 실행되어 위험 상태 'open∧crossing'를 즉시 빠져 나올 수 있게 한다.

따라서 전이 t1에 대해 고장 모드 'fail(en(open∧approaching, t1)'을 수행할 필요가 있다. 이 고장 모드에 대한 결함 트리를 구성하기 위해서는 전이 t1의 이벤트 'evEnter'가 발생해야 하는 것을 전제로 한다. 따라서 'occ(evEnter))'가 'fail(en(open∧approaching, t1)'의 결함 트리에서 조건 이벤트의 하나로 존재하는 것을 볼 수 있으며 'fail(di(open∧approaching, t1)'에서 'evEnter' 외의 이벤트 발생을 고려하지 않는다.

[그림 6]에서 보여준 고장 모드 'fail(en(open∧approaching, t1)'의 결함 트리가 [그림 2]의 결함 트리

템플리트와 일대일 대응이 되지 않는 것을 볼 수 있다. 이는 전이 t1이 이벤트 'evEnter'로 만 이루어져 있어 가드에 대한 고장 모드를 고려할 필요가 없기 때문이다. 또한 앞에서 기술하였듯이 'evEnter'가 반드시 발생한다는 전제를 하고 있으므로 고장 모드 'fail(occ(evEnter))'에 대한 결함 트리 구성은 하지 않는다.

전이 t7에 대한 고장 모드 'fail(di(open∧approaching, t7)'에 대한 결함 트리를 구성할 때 가드에 대한 고장 모드 'fail(guard(open∧approaching, state=closed))'도 고려하여야 한다. 우선 가드의 평가가 잘못되는 경우에 대한 분석을 진행한다. 지금 예에서는 가드 '[state=closed]'가 거짓이 되어야 하는 상황에서 참으로 평가가 되어야 하는 상황을 고장 모드 'fail(chk(open∧approaching, !(state=closed))'를 부모 노드로 한 부결함 트리에 반영하였다.

또한 가드의 값의 설정에 대한 고장 모드 분석 'fail(set(open∧approaching,!(state=closed))'은 전이 상태 'opening∧approaching'에서 전이 t5가 실행되는 경우만 고려하여 수행된다. 그 이유는 t5에서 설정된 'state'의 값이 전이 t7에 전달되는 (전이 t5만으로 구성된) 전이 시퀀스가 존재하기 때문이다. 나머지 전이, 예를 들면, t2에서도 변수 state의 값을 설정하지만 전이 t7에 영향을 미치는 전이 시퀀스가 존재하지 않는다. 또한 전이 t5만으로 구성된 전이 시퀀스가 유일하게 t7에 영향을 미치는 것을 알 수 있다. 이 예제에서는 전이 시퀀스가 유일하므로 특정 전이 시퀀스만을 선정하는 기준에 대해서는 언급하지 않는다. [그림 6]에서 전이 t5에서 설정된 가드 평가 값이 상태 'open∧approaching'에 잘못 전달되는 고장 모드, 'fail(trans(opening∧approaching,t5, (state=closed))'에 대한 결함 트리는 생략하였다. 이와 같은 과정을 위험 상태의 모든 선행 상태에 대해서 수행하여 결함 트리를 구성한다.

## V. 결론 및 향후 연구

이 논문에서는 시스템의 상태 전이 모델을 기반으로 결함 트리를 구성하는 방법에 대해 기술하였다. 기본적으로 결함 트리를 구성하는 작업은 직관과 경험에 많이 의존되어 왔던 것이 사실이며 매우 많은 시간과 노력을 필요로 한다. 이 논문에서는 시스템 상태 전이 모델이 안전성을 확보하기 위해 갖추어야 하는 두 가지 조건('위험 회피 조건'과 '도달성 조건')을 식별하였고 이 조건들이 만족되지 않을 수 있는 요인들의 분석을 통해 결함 트리 템플리트를 개발하였다. 또한, 철도 건널목 제어 시스템에 제안된 결함 트리 구성 방법을 적용한 결과 체계적으로 결함 트리를 구성할 수 있음을 확인하였다.

향후 연구에서는 제안된 방법을 가능한 자동화 할 수 있는 도구를 개발할 계획이다. 이를 위해 시스템 상태 전이 모델을 정적으로 분석하여 위험 회피 조건과 도달성 조건을 자동으로 식별할 수 있어야 한다. 이에 대한 방안으로 모형 검사(model checking)[11] 기술을 이용하는 방법에 대한 연구가 진행 중이다. 또한, 결함 트리를 구성할 때 시스템을 구성하는 상태의 개수가 많을수록 결함 트리가 매우 복잡하고 커질 수 있기 때문에 결함 트리를 보다 단순화 할 수 있는 방안에 대한 연구도 필요하다.

## 참 고 문 헌

[1] H. A. Watson, *Launch Control Safety Study*, Technical report, Bell Telephone Laboratories, Murray Hill, NJ, 1961.

[2] W. Vesely, F. Goldberg, N. Roberts, and D. Haasl, *Fault Tree Handbook*, Technical Report NUREG-0492, U.S. Nuclear Regulatory Commission, 1981.

[3] M. Stamatelatos and W. Vesely, F*ault Tree Handbook with Aerospace Applications*, Technical Report of NASA, 2002(8).

[4] J. Xiang, K. Futatsugi, and Y. He. "Fault tree and Formal Methods in System Safety Analysis," In Proc. of The 4th International Conference on Computer and Information

Technology, pp.1108-1115, Wuhan, China, 2004(9).

[5] N. G. Leveson, Stephen S. Cha, and Timothy J. Shimeall, "Safety Verification of Ada Programs Using Software Fault Trees," IEEE Software, pp.48-59, 1991(7).

[6] M. Towhidnejad, D. Wallace, and A. Gallo. "Fault Tree Analysis for Software Design," In Proc. of 28th Annual IEEE/NASA Software Engineering Workshop, 2003.

[7] K. Lano, D. Clark, and K. Androutsopoulos, "Safety and Security Analysis of Object Oriented Models," Lecture Notes in Computer Science, p.2434, pp.82-93. 2002.

[8] J. Gorski and B. Nowicki, "Object Oriented Approach to Safety Analysis," In Proc. of ENCRESS, pp.338-350, 1995.

[9] R. Hawkins, Ian Toyn and Iain Bate, "Critical Systems Development with UML," In Procs. of UML 2003 workshop, San Fransisco 2003.

[10] H. J. Kim, W. E. Wong, D. Vidroha Debroy, and B. Doohwan, "Bridging the Gap between Fault Trees and UML State Machine Diagrams for Safety Analysis," In Proc. of APSEC, pp.196-205, 2010.

[11] P. Godefroid, *Partial-Order Methods for the Verification of Concurrent Systems-An Approach to the State-Explosion Problem*. University of Liege, Computer Science Department. Ph.D. Thesis.

[12] P. G. Frankl and E. J. Weyuker, "An applicable family of data flow testing criteria," IEEE Transactions on Software Engineering, Vol.14, No.10, pp.1483-1498, 1988.

## 저 자 소 개

**정 인 상(In-Sang Chung)** 정회원

- 1987년 : 서울대학교 컴퓨터공학과 졸업(학사)
- 1989년 : 한국과학기술원(KAIST) 전산학과 졸업(석사)
- 1993년 : 한국과학기술원(KAIST) 전산학과 졸업(박사)
- 1999년 ~ 현재 : 한성대학교 컴퓨터공학과 교수

<관심분야> : 소프트웨어 공학, 소프트웨어 테스팅