Fault Tree의 저장을 위한 범용적인 형태의 XML 스키마

서영주 ○, 이동아, 유준범
건국대학교 컴퓨터공학부
{syjsmk, ldalove, jbyoo}@konkuk.ac.kr

XML Schema of General Form Definition for Fault Tree

Young-Ju Seo ○, Dong-Ah Lee, JunBeom Yoo
Konkuk University, Division of Computer Science and Engineering

요 약

고장수목분석이란 시스템의 위험성을 분석하는 기법이다. 본 논문에서는 고장수목분석을 수행할 때 생성되는 고장수목을 효과적으로 분석 및 저장하기 위한 XML 스키마의 정의를 소개한다. 이는 일반적인 Fault Tree와 이를 확장한 경우에도 이를 잘 나타내고 저장할 수 있다. 또한 Function Block Diagram의 고장수목을 XML 스키마에 따라 저장한 사례를 보인다.

1. 서 론

고장수목분석(FTA, Fault Tree Analysis)은 시스템의 위험성을 분석하는데 사용되는 기법으로 어떤 특정한 예상 사고에 대하여 그 사고의 원인이 되는 장치/기기의 결함이나 오류를 순차적, 도식적으로 검토 및 분석하여 시스템의 위험성을 평가하는 방법이다 [1]. 이 기법은 트리를 이용해서 시각적으로 분석 결과를 보여주며, 최상위 이벤트인 시스템의 고장으로부터 최하위 이벤트인 고장의 원인을 역으로 찾아가는 방식으로 시스템의 위험을 분석한다. 그림 1은 이러한 FTA의 예시를 나타내고 있다.

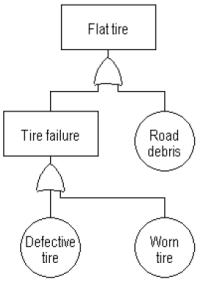

그림 1. FTA(Fault Tree Analysis)[1]

XML은 텍스트 기반의 표기법이며 구조적인 데이터를 표기하는데 자주 사용된다. XML은 단일하게 규정된 방식으로 정의되기 때문에 데이터에 대한 사전 정보가 없어도 알아보기 쉽게 되어있다 [2].

본 논문에서는 고장수목분석에서 생성되는 고장수목(FT, Fault Tree)의 저장을 위해 새롭게 정의한 XML 스키마를 소개한다. 또한 PLC(Programmable Logic Controller)에서 동작하는 소프트웨어를 개발하는데 사용되는 PLC용 표준 프로그램 언어 중 하나인 FBD(Function Block Diagram)로 개발된 소프트웨어를 대상으로 FTA를 실시하여 그 결과를 본 논문에서 정의한 XML 스키마에 따라 저장한 결과를 보여준다. 그림 2는 FBD로 작성된 프로그램의 예를 나타내고 있다.

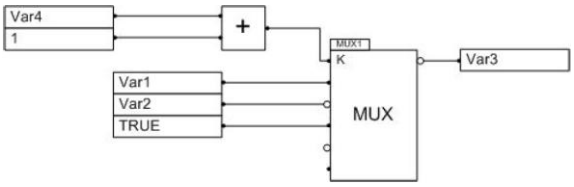

그림 2. FBD(Function Block Diagram)[3]

현재 FT의 저장에 대한 표준은 없다. 관련 분야의 기존 연구로는 BDD(Binary Decision Diagram)을 이용해 트리를 나타내고 이를 XML로 저장하는 방식의 XFTOM(Extensible Fault Tree Object Model)[4] 모델을 정의한 연구가 있다. 이 연구에서는 BDD를 사용했기 때문에 기존의 FT에 약간 변형이 가해진 Gate나 셋 이상의 Node가 달린 경우에 대해서의 처리가 어렵다. 본 논문에서 제시하는 FT 저장을 위한 XML 스키마를 통해 일반적인 FT의 저장 및 이를 확장한 경우에도 이를 잘 저장할 수 있다.

2. FT의 Node 및 Gate

본 연구에서는 FT를 나타내기 위하여 다음과 같은 형태의 Node 및 Gate들을 사용한다.

그림 3은 트리를 만들 때 각 부분의 중간에 들어가게 되는 Node의 일반적인 형태로서, 윗부분에 해당 Node의 수식이 들어가고 아랫부분에 해당 Node가 어떤 FB를 FT로 만드는 과정에서 나오게 된 것인지

FB의 이름을 표기하고 있다. 각 Node들의 위나 아래로 다른 Node나 Gate들이 연결될 수 있다. 일반적인 FT에서 사용하는 것과 같이 상단에 Fault의 내용을 적고 하단에 그에 대한 설명을 적는 방식의 표기도 가능하다. 만약 나타내고자 하는 Node가 트리의 최하단에 나타나는 기본 이벤트라면 우측과 같은 형태로 나타내게 된다.

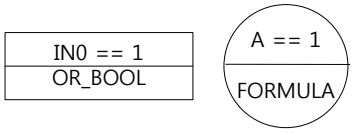

그림 3. Node

Gate는 그림 4에 나와있는 것처럼 기본적인 FT에 사용되는 ADD, OR외에 본 논문에서 사용한 FT에는 특수한 경우에 사용되는 Gate들로 ORDER, Temporal Gate 등이 있다. AND, OR Gate는 연산 결과를 기본적인 논리연산으로 나타낼 수 있는 경우에 사용되며 그렇지 못한 경우들의 표현을 위해 추가 정의한 Gate를 사용할 수도 있다.

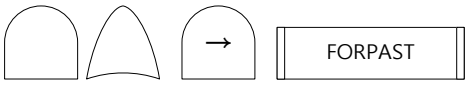

그림 4. 사용된 Gate의 일부

3. XML 스키마

FT의 결과를 저장하기 위한 XML의 스키마를 그림 5와 같은 형태로 정의하였다. 스키마는 FT의 각 Node가 가지는 정보를 저장할 수 있게 정의되었다. type Attribute에는 해당 Node가 BASIC이나 INTERMEDIATE 중 어떤 것에 속하는지, 또는 Gate의 경우에는 어떤 종류의 Gate인지에 대한 정보를 저장하며, id Attribute나 Node들의 중첩을 통해 트리의 계층구조를 저장한다. data Attribute에는 Node가 공식과 같은 형태로 값을 가지고 있을 경우에 해당 공식의 값을 또는 Fault에 대한 정보를 저장한다. 마지막으로 desc Attribute에는 해당 Node가 FT로 만들어지기 전에 어떤 데이터의 분석에 의해서 생성된 Node인지 formula Attribute에서는 다 표현하기 힘든 추가 정보를 나타내게 되어있다. 이는 하나의 Gate에 여러 개의 Node가 추가될 경우에도 FT의 형태를 효과적으로 나타낼 수 있다.

4. FTA 결과 저장방식의 예시

앞서 언급된 Node나 Gate들을 사용하여 FT를 나타낼 경우 그림 6에 나와있는 AND FB에 대한 FT로 그림 7과 같은 형태가 나오게 된다.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://dslab.konkuk.ac.kr/FaultTreeSchema/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://dslab.konkuk.ac.kr/FaultTreeSchema/">

    <xs:element name="root" type="tns:rootType"></xs:element>

    <xs:element name="node" type="tns:nodeType"></xs:element>

    <xs:element name="faultTree" type="tns:faultTreeType"></xs:element>

    <xs:complexType name="rootType">
        <xs:sequence maxOccurs="unbounded" minOccurs="0">
            <xs:element name="node" type="tns:nodeType"></xs:element>
        </xs:sequence>
        <xs:attribute name="desc" type="string"></xs:attribute>
        <xs:attribute name="data" type="string"></xs:attribute>
        <xs:attribute name="id" type="string"></xs:attribute>
        <xs:attribute name="type" type="string"></xs:attribute>
    </xs:complexType>

    <xs:complexType name="nodeType">
        <xs:sequence maxOccurs="unbounded" minOccurs="0">
            <xs:element name="node" type="tns:nodeType"></xs:element>
        </xs:sequence>
        <xs:attribute name="desc" type="string"></xs:attribute>
        <xs:attribute name="data" type="string"></xs:attribute>
        <xs:attribute name="id" type="string"></xs:attribute>
        <xs:attribute name="type" type="string"></xs:attribute>
    </xs:complexType>

    <xs:complexType name="faultTreeType">
        <xs:sequence>
            <xs:element name="root" type="tns:rootType"></xs:element>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

그림 5. FT를 위해 정의한 XML의 스키마

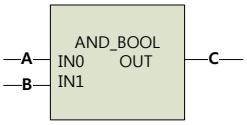

그림 6. FB의 예시(AND_BOOL)

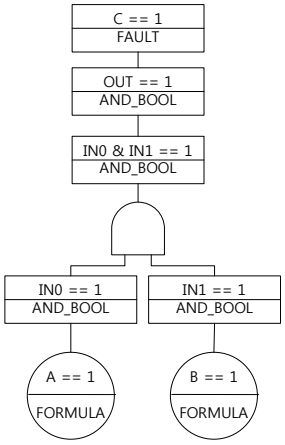

그림 7. 생성된 FT의 예시(AND_BOOL)

이 경우는 Fault가 발생하려면 AND FB의 연산 결과가 1이 나와야 하며 이를 위해서는 AND FB에 들어오는 입력 값인 A와 B가 1이어야 함을 나타내는 것이다. 위의 결과를 공식으로 나타낼 경우는 생성된 FT에 대해 중위순회(inorder traversal)을 하면 해당 결과를 얻을 수 있으며 C가 1이 되기 위한 조건으로 *((A == 1) & (B == 1))* 이라는 공식을 얻을 수 있다. 이를 앞서 언급한 XML 스키마를 이용해서 나타내게 되면 그림 8과 같은 형태로 저장되게 된다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<tns:faultTree xmlns:tns="http://dslab.konkuk.ac.kr/FaultTreeSchema/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://dslab.konkuk.ac.kr/FaultTreeSchema/
    ../com.DSlab.KU.FBDFTA/FaultTreeSchema.xsd ">
    <root data="Var3 == 1" desc="Fault" id="1" type="EVENT_BASIC">
        <node data="OUT1 == 1" desc="AND_BOOL" id="2" type="EVENT_INTERMEDIATE">
            <node data="IN1 &amp;amp; IN2 == 1" desc="AND_BOOL" id="3"
                type="EVENT_INTERMEDIATE">
                <node data="" desc="AND_BOOL" id="4" type="GATE_AND">
                    <node data="IN1 == 1" desc="AND_BOOL" id="5" type="EVENT_INTERMEDIATE">
                        <node data="Var1 == 1" desc="formula" id="6" type="EVENT_BASIC" />
                    </node>
                    <node data="IN2 == 1" desc="AND_BOOL" id="7" type="EVENT_INTERMEDIATE">
                        <node data="Var2 == 1" desc="formula" id="8" type="EVENT_BASIC" />
                    </node>
                </node>
            </node>
        </node>
    </root>
</tns:faultTree>
```

그림 8. XML의 스키마를 따라서 저장된 FT

이를 보면 트리의 최하단인 BASIC에 해당하는 값이 *Var1 == 1*과 *Var2 == 1*의 공식이 들어감을 알 수 있고, 해당 공식들이 AND Gate를 통해서 상위의 *IN1 & IN2 == 1*의 공식과 연결되어있음을 알 수 있다. 따라서 *IN1 & IN2*의 결과가 1이 나오기 위해서는 *Var1*과 *Var2*에 들어오는 입력 값이 전부 1이어야 함을 알 수 있다.

ADD에 대한 FT는 여러 개의 값을 더한 결과를 나타낸다. 이 경우 들어올 수 있는 입력 값이 셋 이상일 수 있고 값을 더한다는 연산은 AND나 OR Gate만으로는 나타내기가 어렵다. 이 경우 기존의 FT에는 없는 ADD에 대한 표현을 새로운 Gate를 정의해서 나타낼 경우 그림 9와 같은 형태가 될 수 있다.

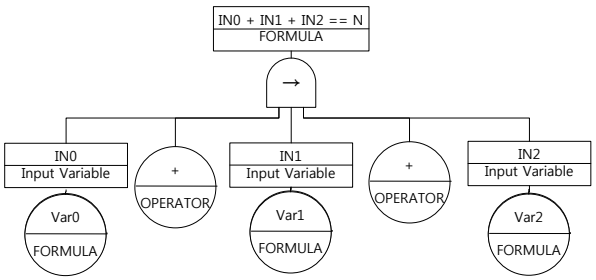

그림 9. ADD에 대한 FT의 예시

앞서 언급한 XFTOM의 경우에는 Binary Tree를 이용해서 FT를 나타내기 때문에 하나의 Node에 여러 개의 Node가 연결된 형태를 나타내기 어렵다. 하지만 우리가 새롭게 정의한 스키마를 사용할 경우 그림 10과 같은 형태로 나타낼 수 있다. 이를 보면 연산 결과인 *Var3 == 1*은 *(IN0 + IN1 + IN2)* 의 결과와 같음을 알 수 있다. 또한 일반적인 FT에는 없는 Gate를 추가할 경우에도 type Attribute의 정보를 보고 알 수 있다.

5. 결론 및 향후 연구

FTA는 오래된 기법이지만 위험성을 분석하는 부분에서는 여전히 효율적으로 사용되고 있다. 하지만 FT의 저장 방식에 대한 표준은 없는 상태이므로 본 논문에서는 FT의 표현과 이를 보다 효율적으로 저장하는 대한 방식에 대해 제안하였다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<tns:faultTree xmlns:tns="http://dslab.konkuk.ac.kr/FaultTreeSchema/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://dslab.konkuk.ac.kr/FaultTreeSchema/
    ../com.DSlab.KU.FBDFTA/FaultTreeSchema.xsd ">
    <root data="Var3 == 1" desc="Fault" id="1" type="EVENT_BASIC">
        <node data="OUT1 == 1" desc="ADD_INT" id="2" type="EVENT_INTERMEDIATE">
            <node data="IN0 + IN1 + IN2 == 1" desc="ADD_INT" id="3"
                type="EVENT_INTERMEDIATE">
                <node data="" desc="ADD_INT" id="4" type="GATE_ORDER">
                    <node data="IN0" desc="Input Variable" id="5" type="EVENT_INTERMEDIATE">
                        <node data="Var0" desc="formula" id="10" type="EVENT_BASIC" />
                    </node>
                    <node data="+" desc="Operator" id="6" type="EVENT_BASIC" />
                    <node data="IN1" desc="Input Variable" id="7" type="EVENT_INTERMEDIATE">
                        <node data="Var1" desc="formula" id="11" type="EVENT_BASIC" />
                    </node>
                    <node data="+" desc="Operator" id="8" type="EVENT_BASIC" />
                    <node data="IN2" desc="Input Variable" id="9" type="EVENT_INTERMEDIATE">
                        <node data="Var2" desc="formula" id="12" type="EVENT_BASIC" />
                    </node>
                </node>
            </node>
        </node>
    </root>
</tns:faultTree>
```

그림 10. XML스키마를 따라서 저장된 ADD의 FT

현재 FTA의 생성과 관련된 연구 및 도구로는 NuFTA같은 도구가 있으며 FTA에 관한 기존 연구는 FBD를 FT로 만드는데 사용되는 템플릿[5]이나 NuSCR로 작성된 결과를 FT로 만드는 것[6] 등이 있다. 하지만 결과로 나온 FT를 저장하는 방식에 대해서 다루고 있는 논문은 그 수가 매우 적고 범용으로 사용하기에는 유연성이 부족하였다. 따라서 본 논문에서 제안한 방식을 통해 기존의 FT의 분석에 사용되는 도구에도 해당 결과를 저장하는데 사용될 수 있을 것이다. 이는 트리의 생성 과정이나 분석 결과를 더욱 이해하기 쉽게 도와줄 것이며, XML을 이용함으로써 이 결과를 다른 시스템에서도 쉽게 사용할 수 있을 것이다.

사사

본 연구는 지식경제부의 지원 및 한국원자력연구원의 안전등급제어기기 엔지니어링 도구 성능개선 기술개발 사업 (KETEP-2010-T1001-01038) 과 한국원자력연구원 주요사업 "원자력계측제어 적합성평가, 감시 및 대응체계 구축" 사업의 지원으로 연구한 결과입니다.

참 고 문 헌

[1] U.S.NRC, "Fault Tree Handbook (NUREG-0492)", 1981

[2] W3C, "Extensible Markup Language(XML) 1.1 (Second Edition)", 2006

[3] PLCOpen, "XML Fotmats for IEC 61131-3", 2009

[4] Ren Yi, Liu Linlin, Zeng Shengkui, "Fault Tree Data Structure Based on XML and the Conversion Method to BDD", World Congress on Computer Science and Information Engineering, Vol.2, pages 264-268, 2009

[5] 이동아, 김의섭, 유준범, "고장수목을 이용한 Function Block Diagram 의 위험성 분석 기법 연구", 한국정보과학회 학술발표논문집, Vol.39, 2B, Page 76-78, 2012

[6] 윤상현, 이동아, 유준범, "NuFTA : A CASE Tool for Automatic Software Fault Tree Analysis", Transactions of Korean Nuclear Society, vol.1, Pages 855-857, 2010