보안시스템의 개발을 위한 시스템 형식론에 의한 사용자 인터페이스 시스템 표현과 DEVS 모델링

안영숙*, 조대호*

An Application of UI System Design Methodology based on System Formalism for a Large Security System

Ahn, Young Sook and Cho, Tae Ho

Abstract

지능적으로 공격 패턴을 달리하는 많은 공격들에 대응하기 위해 기존의 보안 시스템은 강력한 보안시스템으로 확장되고 있다. 보안 시스템과 같은 규모가 크거나 복잡한 소프트웨어 계발에 있어서 설계 및 구현 과정에서 있는 설계 변경 및 구현상의 변경으로 인한 전체 시스템에 미치는 영향이 크다. 본 논문에서는 규모가 큰 보안시스템 계발에 있어서 전체 시스템의 영향 관계를 체계적으로 정립한 소프트웨어 설계 방법론으로의 적용이다. 소프트웨어 설계 방법론은 시스템 형식론에 의거하여 소프트웨어가 표현되었으므로 논리적 연관관계를 추적하기 쉽다. 이는 소프트웨어 설계 또는 설계 변경 후 인터페이스 시스템의 중요한 동적 특성을 미리 파악 할 수 있게 하여 소모적일 수 있는 시간과 노력을 절약 할 수 있다.

1.서론

정보통신 네트워크의 발전과 더불어 외부인의 불법 침입, 정보의 유출 및 훼손, 컴퓨터 바이러스 및 서비스 거부 등 역기능이 날로 증대되어 피해 규모가 심각한 수준에 이르고 있다[4,5].

최근 들어, 네트워크의 보안성 향상을 위하여 이미 실용화되어 실제 적용 되고 있는 침입탐지 및 침입차단 시스템, 가상 사설망 등과 같은 보안 제품군들은 기술적으로 성장단계에 있으며, 지능적으로 공격 패턴을 달리하는 많은 공격들에 대응하기 위해 기존의 보안 시스템은 강력한 보안시스템으로 확장되고 있다. 보다 강력한 보안시스템 개발을 위해 설계변경, 기능추가, 재설계 등이 요구된다. 이에 따라 시스템 변경이 불가결하다. 보안시스템과 같이 응용시스템에 의존적인 사용자 인터페이스 개발에 있어서 시스템 변경은 전체 시스템에 미치는 영향이 크다. 이에 따른 많은 시간과 노력이 필요하다.

이를 효과적으로 해결하기 위한 방안으로 시뮬레이션 모델을 통해 소프트웨어 설계 또는 설계 변경 후 인터페이스 시스템의 중요한 동적 특성을 미리 파악 하는 것이다. 시뮬레이션 모델을 통해 소프트웨어 구성요소를 다양하게 변경하고, 시뮬레이션을 반복적으로 수행할 수 있으므로 인터페이스 시스템에 미치는 영향 등을 효과적으로 파악 할 수 있다. 본 논문의 목적은 보안 시스템과 같은 규모가 크거나 복잡한 소프트웨어 계발에 있어서 소프트웨어 설계 및 구현 과정에서 있을 수 있는 모듈 변경에 따른 전체 시스템에 미치는 영향을 자동으로 추적하도록 하고 시스템 형식론에 의한 소프트웨어 설계 방법론을 체계적으로 정립하는데 있다. 참고문헌[6]에서 사용자 인터페이스 설계 방법론이 제시되었다. 소프트웨어 설계 방법론은 시스템 형식론에 의거하여 소프트웨어가 표현되었으므로 논리적 연관관계를 추적하기 쉽다

2. 관련 연구

최근 객체 지향 방법론의 영향으로 객체의 효율적인 관리와 사용 방법에 관한 관심이 높아지고 있으며, Booch, OMT, Jacobson등의 객체 지향 방법론이 등장하였다[1]. 이에 맞추어 소프트웨어의 설계 방법론과 변경관리에 대한 연구가 활발히 진행되고 있다.

의미론적 데이터모형과 메타모형의 유용한 개념을 추가한, 확장된 객체지향 데이터모형을 이용한 변경관리 시스템[2]은 개체간의 상호의존성에 함축된 의미들을 몇가지의 요소관계성들로 추상화함으로써 다양한 관계성들을 데이터모형 내에 경제적으로 정의하여 광범위한 변경관리가 가능하게 하였다. 분석 단계에서 클래스 관계성 정보를 테이블로 구성하고, 테이블 구성 형태를 링크 형태로 표시하는 방법을 제시[1]한 논문에서는 개발자에게 프로그램의 이해와 유지보수를 효율적으로 수행할 수 있도록 하였다. 그러나 복잡한 클래스 계층 구조일 때 모든 경우를 개발자가 분석하는데 많은 시간을 필요로 한다.

기존의 변경관리 시스템들은 대부분 제한된 숫자의 관계성만을 다루기 때문에 변경처리 범위가 국부적이다. 본 연구에서는 시스템 형식론을 적용하여 인터페이스 시스템을 입출력 시스템, 구조적 입출력 시스템, DEVS 모델로 재구성함으로써 추상화된 DEVS 모델을 통해 인터페이스 시스템에서의 변경이 전체 시스템에 미치는 영향을 체계적으로 정립한 소프트웨어 설계 방법론이다.

3. 배경 이론

이 장에서는 대상 시스템인 보안 시스템에 있어서 소프트웨어 설계방법론에 적용된 시스템 이론, DEVS 방법론, 객체지향 프로그래밍에 대한 기본 개념을 살펴본다.

3.1 시스템 이론 (System Theory)

시스템 이론에서 정의하는 시스템 형식론은 계층적 구조를 갖는 시스템 명세(System Specification)와 각 계층의 시스템간의 구조 관계(Interrelation)에 대해 정의한다. 시스템 명세는 입출력에 의한 동적 특성만을 파악하는 하위 계층에서부터 형식적인 구조를 갖는 상위 계층으로 구성되어 있다. 시스템간의 구조 관계는 같은 계층에서 한 시스템과 다른 시스템과의 관계성을 명시한다. 상위 계층의 구조 관계는 하위 계층의 구조 관계를 포함한다. 즉, 상위 계층은 하위 계층의 모든 구조적인 특징을 포함한다[7,8]. <표 1>은 계층성을 갖는 시스템의 종류와 형식을 나타낸다.

계층	시스템 명세
0	I/O Relation Observation
1	I/O Function Observation
2	I/O System Specification
3	Structured I/O System Specification
4	Network of Specifications

<표 1> 시스템 명세의 계층도[7,8]

<표1>의 5계층의 시스템 중에서 본 연구에서는 계층 2,3의 입출력 시스템과 구조적 입출력 시스템을 보안시스템에 적용하였다. 계층 2,3의 시스템 명세와 각 시스템간의 구조 관계에 대한 정의는 다음과 같다.

I/O System Specification은 연속적인 입력 세그먼트의 결합(Composition)된 형태의 세그먼트 집합을 고려한 시스템이다.

시스템 $S=<T,X,\Omega,Q,Y,\delta,\lambda>$에서 각 상태 $q\in Q$마다 $B_q:\Omega\rightarrow(Y,T),\omega\in\Omega$의 관계가 있고, 이를 상태 q에 대한 입출력 함수(I/O function)라 한다.

$$\beta_q(\omega) = OTRAJ_{q,\omega}$$

(I/O Function Observation) IOFO (T,X, Ω,Y,F)와 $S = <T,X,\Omega,Q,Y,\delta,\lambda>$의 구조 관계 $B_S=\{\beta_q|q\in Q\}$는 I/O behavior로서 $IOFO_S$ (T,X,Ω,Y,B_S)라고 표현 할 수 있다.

Structured I/O System Specification는 추상화된 집합과 함수들을 원시 집합과 함수들의 크로스 프로덕트(cross product)로써 표현한다.

$M = <Q,\delta>$은 다음과 같은 구조를 갖는다.

D (coordinates)

$\{Q_\alpha|\alpha\in D\}$ (range sets)

$\{I_\alpha|\alpha\in D,\ I_\alpha\subseteq D\}$ (influencers),

$\{\delta_\alpha|\alpha\in D,\ \delta_\alpha: \underset{\beta\in I_\alpha}{X} Q_\beta\rightarrow Q_\alpha\}$ (local transition functions)

$S = <T,X,\Omega,Q,Y,\delta,\lambda>$와 $M = <Q,\delta>$의 구조 관계 다음과 같이 정의된다.

$$Q \subseteq \underset{\alpha\in D}{X} Q_\alpha, \text{ and}$$

$$\delta = \underset{\alpha\in D}{X} \delta_\alpha \cdot \text{proj } I_\alpha \text{ restricted to } Q$$

3.2 DEVS 방법론

본 연구에서의 시뮬레이션 모델링은 Zeigler에 의해 정립되어 이산사건 모델들의 계층 구조적 모듈화 방법을 제공해주는 형식론인 DEVS (Discrete Event System Specification)에 따른다. DEVS는 시스템이 일반적으로 갖는 특성들을 정의하여 시스템을 모델링할 수 있는 프레임웍(Framework)을 제공하며 형식론을 채택하여 시스템을 관찰하고자 하는 초점으로 추상화하여 모델링함으로써

모델과 실제 문제와의 관련성을 높일 수 있다. DEVS의 기본(Basic) 모델은 다음과 같은 항들로 명세할 수 있다[9,10,11].

$M = <X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta>$

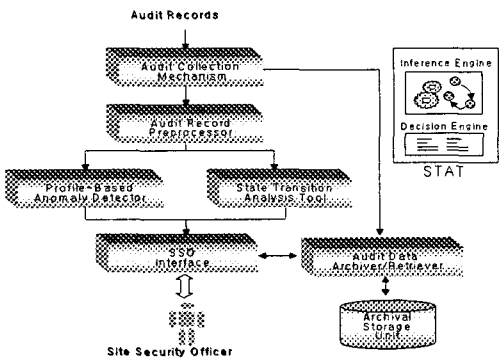
X : 입력 사건(Event)들의 집합
S : 사건의 변화에 따라 가질 수 있는 상태들의 집합
Y : 출력 사건들의 집합
δ_{int} : S→S, 내부 상태전이(State transition) 함수
δ_{ext} : Q×X→S, 외부 상태전이 함수
λ: S→Y, 출력 함수
ta : $S \rightarrow R^{+\infty}_{0}$, 시간 진행 함수
where $Q = \{(s,e) \mid s \in S, 0 \le e \le ta(s)\}$
e: 최근의 상태 전이 이후로 경과된 시간

3.3 객체지향 프로그래밍

객체지향 프로그래밍은 캡슐화를 통하여 소프트웨어의 복잡도를 제어하는데 도움이 되는 패러다임을 제공하기 때문에 사용자 인터페이스 개발에 매우 중요하다. 객체지향 프로그래밍은 상속에 의해 기존의 소프트웨어의 재사용을 증진시킨다. 객체지향에 의해 제공되는 기존의 부품을 수정하고 재사용하는 능력은 광범위한 프로그래밍 없이 사용자 인터페이스에 대한 프로토타입을 가능하게 한다. 객체지향 프로그래밍은 동일한 속성, 동일한 행위, 동일한 객체들과의 동일한 관련성, 동일한 시멘틱을 갖는 객체 집합에 대한 일반적인 기술체로써 클래스를 정의한다. 클래스내의 모든 객체들은 속성의 값만을 다르게 가질 뿐 동일한 속성과 행위를 갖게 되므로 이러한 개념으로 구성된 인터페이스 시스템의 각 클래스는 구조 관계에 의해 구조적 입출력 시스템의 구성 요소(coordinate)들로 변환된다. 그러므로 객체지향 프로그래밍 사고틀에 의해 인터페이스 시스템을 구현하는 것은 매우 중요한 의미를 갖는다[12,13,14,15,16].

4. 보안 시스템 모델

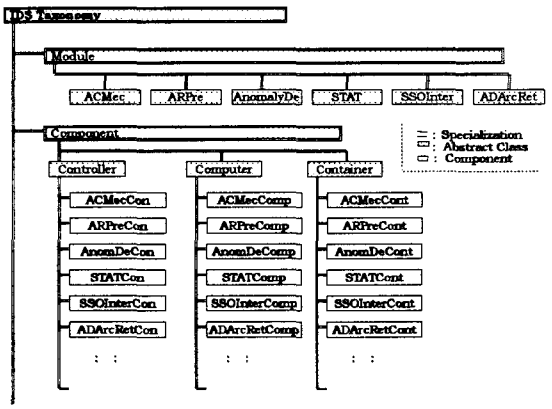
본 연구에서는 보안시스템인 침입탐지시스템의 효과적인 계발을 위해 시스템 형식론에 의거한 소프트웨어 설계 방법론을 적용하였다. 침입 탐지 시스템의 침입 탐지 접근 방법에는 오용 탐지(misuse detection)와 비정상 행위 탐지(anomaly detection)방법이 있다. 오용 탐지(misuse detection) 방법은 침입 탐지 시스템 내부에 저장된 오용 정보를 사용하여 사용자의 입력 행동 특성이 오용인지를 점검하는 방법이다. 일반적으로 오용 탐지 방법은 전문가 시스템을 사용하는데, 전문가 시스템은 사용자의 행동 특성과 이미 저장된 침입 정보를 사용하여 추론을 하게 된다. 비정상 행위 탐지(anomaly detection)기법은 잘 발생하지 않는 상황을 침입으로 간주한다는 가정을 기반으로 침입을 탐지하는 방법이다. 이는 침입자의 행위의 특성이 일반적인 사용자의 그것과 주목할 만큼 다르다는 것을 전제로 한다. [17,18]. <그림1>은 오용탐지와 비정상 행위 탐지 방법을 사용한 호스트 기반 침입탐지시스템의 flow diagram 이다[19]. 호스트 기반 침입 탐지 시스템은 시스템 내부의 로그 파일이나 감사 파일을 통해서 침입 정보를 찾아낸다.



<그림 1> Organization of intrusion detection components

4.1 IDS 인터페이스 시스템의 클래스 계층

<그림2>는 IDS 인터페이스 시스템의 클래스 계층을 나타낸다. 소프트웨어 공학적인 토대와 객체지향 프로그래밍 사고틀에 의해 클래스들로 구성되었다. 최상위에 IDS시스템이 있고, 이를 구성하는 모듈(Module)들이 있으며, 각 모듈 안에는 다양한 구성요소(components)가 있다.

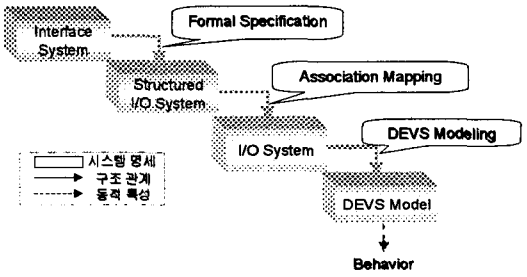


<그림 2> IDS 인터페이스 시스템의 클래스 계층

클래스내의 모든 객체들은 속성의 값만을 다르게 가질 뿐 동일한 속성과 행위를 갖게 된다. 이러한 개념으로 구성된 IDS 인터페이스 시스템의 각 클래스는 구조 관계에 의해 구조적 입출력 시스템의 구성요소(Coordinate)들로 변환된다.

5. 시스템 명세의 계층구조

시스템 이론에서 정의하는 시스템 형식론은 계층적 구조를 갖는 시스템 명세(System Specification)와 각 계층의 시스템간의 관계를 정의하는 구조관계(Interrelation)에 대한 정의이다. 본 연구에서는 침입탐지시스템의 시스템 명세를 인터페이스 시스템(Interface System), 구조적 입출력 시스템(Structured I/O System), 입출력 시스템(I/O System), DEVS 모델 등 4가지 시스템으로 구분하여 적용되었다.



<그림 3> 시스템 명세의 계층구조도

형식 명세에 의해 인터페이스 시스템을 구조적 입출력 시스템의 요소들로 표현하고, 이를 관계 사상에 의해 입출력 시스템으로 변환한 다음 DEVS 모델링에 의해 DEVS 모델로 재구성하였다. DEVS 시뮬레이션 환경에서 제공하는 시뮬레이터를 통하여 적용 대상 시스템인 침입 탐지 시스템의 중요한 동적 특성을 소프트웨어 설계시 또는 설계 변경 후 미리 파악할 수 있다.

침입탐지시스템을 <그림3>과 같은 계층적 구조를 갖는 4가지 시스템으로 표현하고 이들의 사상을 통해 <표2>와 같은 상호 의존성

을 추적하는 것이 이 시스템 설계의 목표이다.

인터페이스 시스템	구조적 입출력 시스템	입출력 시스템	DEVS 모델
IDSUIApp.InitAction() ATAT.OnOK()	P_{O1-1},P_{O2-1},P_{O3-1},P_{O4-2},P_{O5-1},P_{O6-1},P_{O7-16}	q_{19}	STAT-1
IDSUIApp.InitAction() ATAT.OnOK()	P_{O1-1},P_{O2-1},P_{O3-1},P_{O4-2},P_{O5-1},P_{O6-1},P_{O7-17}	q_{20}	STAT-2
IDSUIApp.InitAction() ATAT.OnOK()	P_{O1-1},P_{O2-1},P_{O3-1},P_{O4-3},P_{O5-1},P_{O6-1},P_{O7-18}	q_{21}	STAT-3
IDSUIApp.InitAction() ATAT.OnOK() IDSUIApp.OnInitialize()	P_{O1-1},P_{O2-1},P_{O3-1},P_{O4-1},P_{O5-1},P_{O6-1},P_{O7-24}	q_{22}	STAT-4

<표 2> STAT 상호 의존성 추적

예를 들어 <표 1>에서 DEVS 모델의 상태가 'STAT-2'이면, 입출력 시스템의 상태는 'q_{20}', 구조적 입출력 시스템의 각 컴포넌트의 상태는 'P_{O1-1},P_{O2-1},P_{O3-1},P_{O4-2},P_{O5-1},P_{O6-1},P_{O7-17}'이고, 이때 인터페이스 시스템의 클래스는 IDSUIApp 와 ATATClass임을 알 수 있다.

5.1 시스템 명세

본 연구에서는 시스템 명세를 인터페이스 시스템, 구조적 입출력 시스템, 입출력 시스템, DEVS 모델 등 4가지 시스템으로 구분하였다. 각각의 내용과 적용된 예를 살펴보면 다음과 같다.

5.1.1 인터페이스 시스템 (Interface System)

IDS 인터페이스 시스템은 소프트웨어 공학적인 토대와 객체지향 프로그래밍 사고틀에 의해 클래스와 멤버함수로 구성되었다.

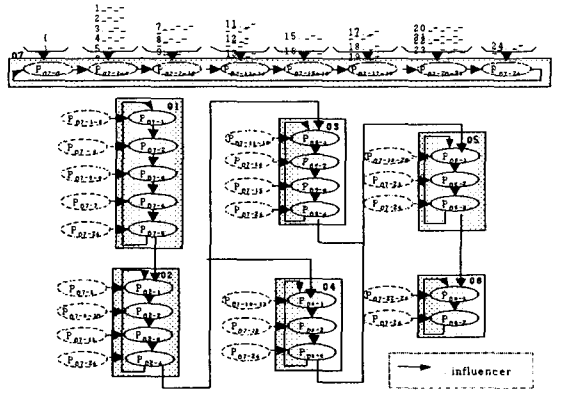
5.1.2 구조적 입출력 시스템 (Structured I/O System)

인터페이스 시스템을 <표 3>와 같이 원시집합과 함수들의 크로스 프로덕트(cross product)로써 표현함으로써 구조적인 시스템을 구성한다. 즉, 인터페이스 시스템을 컴포넌트와 컴포넌트간의 연관 관계, 영향도 등으로 표현한다[1,2].

<표 3>에서 적용된 예를 보면 인터페이스 시스템이 구조적 입출력 시스템 M의 Q와 δ는 상태집합과 함수들의 크로스 프로덕트로 표현되었고, M은 $(D,\{Q_\alpha\},\{I_\alpha\},\{\delta_\alpha\})$와 같은 구조(structure)를 갖는다. <그림 4>를 보면 7개의 컴포넌트(coordinate:D)와 컴포넌트의 상태(state set:Q_α)로 구성되어 있다. 각 상태에 영향을 주는 요소(Influencer:I_α)는 그림에서 화살표로 표현되어 있다. 또한 여러개의 Influencer의 영향을 받아 로컬 상태전이 함수(Local transition function:δ_α)에 의해 상태전이를 하게 된다.

정의	M (machine) = < Q,δ> Q : the cross product of the component states sets δ : the cross product of component functions structure of M : $(D,\{Q_\alpha\},\{I_\alpha\},\{\delta_\alpha\})$ D (coordinates) , $\{Q_\alpha \mid \alpha\in D\}$ (range sets) $\{I_\alpha \mid \alpha\in D,\ I_\alpha\subseteq D\}$ (influencers) $\{\delta_\alpha \mid \alpha\in D, \delta_\alpha: \times_{\beta\in I_\alpha} Q_\beta \rightarrow Q_\alpha\}$(local transition function) such that, $Q \subseteq \times_{\alpha\in D} Q_\alpha$, and $\delta = \times_{\alpha\in D} \delta_\alpha$ 。 proj I_α restricted to Q
적용	M = < Q,δ> where Q = { P_{O1}, P_{O2}, P_{O3}, ... , P_{O7} } δ = { O1, O2, O3, ... , O7 } $(D,\{Q_\alpha\},\{I_\alpha\},\{\delta_\alpha\})$ where D = { O1, O2, O3, ... , O7 } $\{Q_\alpha\}$ = { P_{O1}, P_{O2}, P_{O3}, ... , P_{O7}} such that P_{O1} = (P_{O1-1}, P_{O1-2}, P_{O1-3}, P_{O1-4}, P_{O1-5}) P_{O2} = (P_{O2-1}, P_{O2-2}, P_{O2-3},P_{O2-4}), ... P_{O7} = (P_{O7-0}, P_{O7-1}, P_{O7-2}, ... , P_{O7-24}) $\{I_\alpha\}$ = { I_{O1}, I_{O2}, I_{O3} , ... , I_{O7} } such that I_{O1} = {O1,O7}, I_{O2} = {O1,O2,O7}, ..., I_{O7} = {O7} $\{\delta_\alpha\}$ = { δ_{O1},δ_{O2}, δ_{O3}, ... ,δ_{O7} } such that δ_{O1} : $P_{O1}\times P_{O7} \rightarrow P_{O1}$, δ_{O2} : $P_{O1}\times P_{O2}\times P_{O7} \rightarrow P_{O2}$, ..., δ_{O7} : $P_{O7} \rightarrow P_{O7}$

<표 3> 구조적 입출력시스템의 정의 및 적용예



<그림 4> 구조적 입출력 시스템의 구성 요소간의 영향도

5.1.3 입출력 시스템 (I/O System)

<표 4>과 같이 입력 세그먼트에 의한 상태 전이와 특정 상태에서 발생하는 출력 함수 등의 추상화된 집합과 함수로 구성된 시스템이다[7,8].

정의	S(system) = < T,X,Ω,Q,Y,δ,λ> T: time base, X: the input value set Ω: the input segment set (a subset of (X,T)) Q: the state set, Y: the output value set δ: the state transition function, λ: the output function
적용	S = < T,X,Ω,Q,Y,δ,λ> where T = Integer, λ= {$\lambda_1,\lambda_2,\lambda_3$, ... ,$\lambda_7$ } X = {ACMecCon(),ARPrecon(),AnomDeCon(,, ...,} Ω = {ω|ω:<0,1> → X } Q = {q_0, q_1, q_2, q_3, ... , q_{29}} such that q_0=(P_{O1-1},P_{O2-1},P_{O3-1},P_{O4-1},P_{O5-1},P_{O6-1},P_{O7-1}) q_1=(***P_{O1-2}***,P_{O2-1},P_{O3-1},P_{O4-1},P_{O5-1},P_{O6-1},***P_{O7-1}***) q_2=(P_{O1-2},P_{O2-2},P_{O3-1},P_{O4-1},P_{O5-1},P_{O6-1},***P_{O7-2}***) ... q_{29}=(P_{O1-1},P_{O2-1},P_{O3-3},P_{O4-1},P_{O5-2},***P_{O6-1}***,***P_{O7-24}***) Y = {ACMecCon(),ARPrecon(),AnomDeCon(), ...,} δ(q) = (δ_{O1}*proj I_{O1}(q), δ_{O2}*proj I_{O2}(q), ... ,δ_{O7}*proj I_{O7}(q)), q ∈ { q_0, q_1, q_2, q_3, ... , q_{29} }

<표 4> 입출력시스템의 정의 및 적용 예

5.1.4 DEVS 모델 (DEVS Model)

DEVS 형식론을 기반으로 시간의 흐름에 따라 입력, 상태, 출력, 상태전이 등을 <표 5>처럼 추상화하여 모델링한 시뮬레이션 모델이다[6,10,11].

정의	M(model) = <X,S,Y,δ_{int},δ_{ext},λ,ta> where X : set of external (input) event types S : sequential state set, Y : output set δ_{int} : S→S, the *internal transition function* δ_{ext} :Q×X→S, the external transition function ta : S→$R^+_{0,\infty}$ the time advance function λ : S→Y, the output function where Q={(s,e)|s∈S, 0≤e≤ta(s)}
적용	M = <X,S,Y,δ_{int},δ_{ext},λ,ta> where, X = {ACMecCon(),ARPrecon(),AnomDeCon(), STATCon(),SSOInterCon(), ...,) , x∈X S = {Passive,ACMec-1,ACMec-2,..,Processed} , s∈S Y = {ACMecCon(),ARPrecon(),AnomDeCon(), STATCon(),SSOInterCon(), ...,), y∈Y δ_{int}(s) = s , δ_{ext}(s e x) = s + x , λ(s) = {$\lambda_1,\lambda_2,\lambda_3$, ... ,$\lambda_7$ }

<표 5> DEVS 모델의 정의 및 적용 예

5.2 구조 관계

5.1절에서 살펴본 각 시스템간의 관계를 정의해주는 것이 구조 관계이다. 형식 명세에 의해 인터페이스 시스템을 구조적 입출력 시스템의 요소들로 표현하고, 이를 관계 사상에 의해 입출력 시스템으로 변환한 다음 DEVS 모델링에 의해 DEVS 모델로 재구성한다. 각각의 내용과 적용된 예를 살펴보면 다음과 같다.

5.2.1 형식 명세 (Formal Specification)

인터페이스 시스템을 구조적 입출력 시스템의 요소들로 추상화하여 표현한다. <표6>을 보면 인터페이스 시스템과 구조적 입출력 시스템의 구조 관계를 알 수 있다.

5.2.2 관계 사상 (Association Mapping)

구조적 입출력 시스템에서 입출력 시스템으로의 상태전이와 상태전이함수를 <표7>에서 볼 수 있다.

정	객체지향 프로그래밍에 의해 구성된 인터페이스 시스템과 구조적 입출력 시스템 (D,{Q_d},{I_d},{δ_d})과의 구조 관계는 인터페이스 시스템의 각 클래스마다 오브젝트를 생성하여 구조적 입출력 시스템의 D(coordinate)로 정의하고, 각 클래스는 상태변수에 의해 오브젝트의 상태집합 Q_d로 표현된다.
적용	D : ACMecClass × ARPreClass × ... × IDSUIApp ⇒ O1 × O2 × ... × O7 Q : ACMecClass.P1 × ARPreClass.P2 × . .× IDSUIApp.P7 ⇒ P_{O1} × P_{O2} × P_{O3} × ... × P_{O7}

<표 6>형식 명세의 정의 및 적용 예

정	구조적 입추력 시스템 M = <Q,δ> D (coordinates), {Qα|α∈D}(range sets) {Iα|α∈D, Iα⊆D}(influencers) {δα |α∈D, δα: $\times_{\beta\in I_\alpha} Q\beta \to Q\alpha$} (local transition functions) 구조적 입출력 시스템과 입출력 시스템과의 관계는 다음과 같이 정의 된다. 입출력 시스템 S = < Q', δ' > Q' ⊆ $\times_{\alpha\in D} Q\alpha$, δ' = $\times_{\alpha\in D} \delta\alpha$ 。 proj Iα restricted to Q
적용	S=<Q',δ'> Q': {P_{O1}, P_{O2}, P_{O3}, ... , P_{O7}} ⇒ {q_0, q_1, q_2, q_3, ... , q_{29}} such that q_0=(P_{O1-1},P_{O2-1},P_{O3-1},P_{O4-1},P_{O5-1},P_{O6-1},P_{O7-1}) q_1=(***P_{O1-2}***,P_{O2-1},P_{O3-1},P_{O4-1},P_{O5-1},P_{O6-1},***P_{O7-1}***) q_2=(P_{O1-2},***P_{O2-2}***,P_{O3-1},P_{O4-1},P_{O5-1},P_{O6-1},***P_{O7-2}***) q_3=(P_{O1-2},P_{O2-3},***P_{O3-1}***,P_{O4-1},P_{O5-1},P_{O6-1},***P_{O7-3}***) ... , and q_{29}=(P_{O1-1},P_{O2-1},***P_{O3-3}***,P_{O4-1},P_{O5-2},***P_{O6-1}***,***P_{O7-24}***) δ': δ(q)= (δ_{O1}*proj I_{O1}(q), δ_{O2}*proj I_{O2}(q), δ_{O3}*proj I_{O3}(q), ... ,δ_{O7}*proj I_{O7}(q)), q ∈ { q_0, q_1, q_2, q_3, ... , q_{29} }

<표 7> 관계 사상의 정의 및 적용 예

5.2.3 DEVS 모델링 (DEVS Modeling)

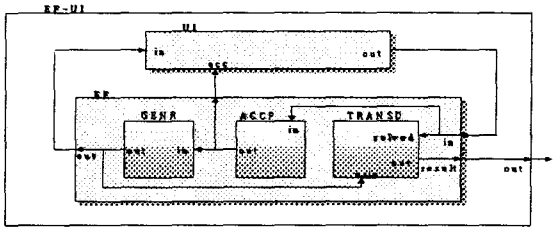
입출력 시스템에서 <표 8>와 같이 DEVS 형식론을 적용하여 DEVS 모델로 모델링하였다.

정	S(System) = <T', X',Ω', Q', Y',δ',λ'> ⇒ M (model) = <X_M, S_M, Y_M, δ_{int}, δ_{ext}, λ_M, ta> where X_M = X', S_M = S', Y_M = Y', δ_{int} : $Q_M \times \not\in \to Q_M$, δ_{ext} : $Q_M \times \Omega' \to Q_M$, where Q_M ={(s,e)|s∈S_M, 0≤e≤ta(s)) ta : S→Real , λ_M : λ'
적용	S = <Q', δ'> S': { q_0,q_1,q_3, ... ,q_{29} } ⇒{PASSIVE, ACMec-1, ACMec-2, ... ,PROC} δ':δ(q) ⇒ δ_{ext} , q ∈ { q_0, q_1, q_2, q_3, ... , q_{29} }

<표 8> DEVS 모델링의 정의 및 적용 예

6. 시뮬레이션 결과

침입탐지시스템의 인터페이스 시스템을 DEVS 모델로 모델링하고 시뮬레이션 되었다. <그림5>는 시뮬레이션 하기위한 EF-UI 모델이다. 이는 DEVS-Scheme을 사용하여 시뮬레이션하고 결과를 통해 유용성을 검증 하였다. UI 모델은 4장에서 기술된 시스템 명세의 계층 구조에 의해 DEVS 모델로 추상화된 인터페이스 시스템이다. EF(Experimental Frame)는 UI 모델과 연결된 coupled 모델로서 입력 세그먼트를 발생시키고, 시뮬레이션의 흐름을 제어하거나, 시뮬레이션 된 내용이나 통계적인 결과를 생성하는 등의 역할을 하고, 3개의 컴포넌트 모델(GENR, ACCP, TRANSD)로 구성된다[9,10].



<그림 5> EF-UI 구조

6.1시뮬레이션이 비정상 종료된 경우

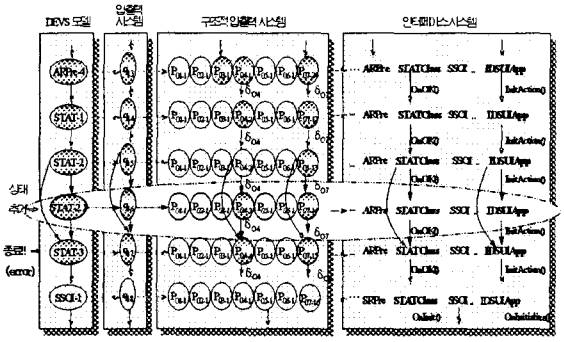
DEVS 모델을 DEVS 시뮬레이션 환경에서 제공하는 시뮬레이터를 통해 시뮬레이션 했을

때 비정상적으로 종료 된 경우, 에러(error) 위치에서 모델의 상태를 살펴보고 비정상적인 종료로 인해 인터페이스 시스템에 미치는 영향을 계층적 시스템의 상호 의존성 추적을 통하여 파악할 수 있다.

예1)DEVS 모델에서 특정 상태(state) 추가

DEVS 모델에서 특정 상태(State)의 추가는 대상시스템인 침입탐지 시스템에서 강력한 보안을 위해 현 시스템에 기능 추가를 할 경우에 일어난다.

STAT-3에서 시뮬레이션이 비정상 종료되었다. DEVS 모델에서 중단된 부분 각 시스템간의 상호 의존성을 추적해보면 DEVS 모델은 'STAT-3', 입출력 시스템은 'q_{17}', 구조적 입출력 시스템의 각 컴포넌트의 상태는 '$P_{01\text{-}1}$,$P_{02\text{-}1}$,$P_{03\text{-}1}$,$P_{04\text{-}4}$,$P_{05\text{-}1}$,$P_{06\text{-}1}$,$P_{07\text{-}15}$ 이다. 그러므로 인터페이스 시스템에서 시뮬레이션이 중단된 부분에 해당되는 클래스는 STATClass와 IDSUIApp 이다. 위와 같은 클래스에서 오류가 발생한 이유는 DEVS 모델에서 State 'STAT-2.1' 추가를 함으로써 영향을 받는 STATClass의 OnOK()와 IDSUIApp의 InitAction()의 변경이 제대로 이루어지지 않았기 때문이다. 이러한 시스템간의 상호 연관성 추적과 변경 관리는 계층적 시스템 명세의 구조 관계(그림4)를 통해 알 수 있다.

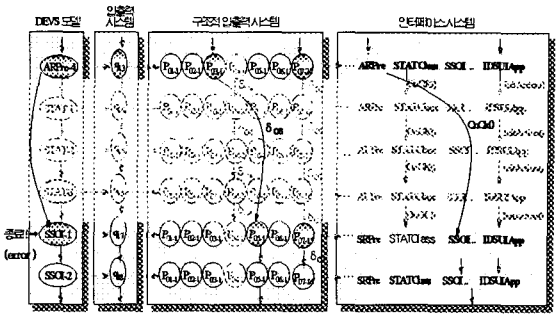


<그림 8> 특정 상태(state) 추가 후의 시뮬레이션 결과

예2)DEVS 모델에서 특정 모듈(module) 삭제

DEVS 모델에서 모듈(module)의 삭제는 대상시스템인 침입탐지 시스템에서 계발도중 불필요한 객체를 삭제 할 경우에 일어난다.

위와 같은 클래스에서 오류가 발생한 이유는 DEVS 모델에서 STAT모듈 삭제를 함으로써 영향을 받은 STATClass의 OnOK()와 IDSUIApp의 InitAction()의 변경이 제대로 이루어지지 않았기 때문이다. 이러한 시스템간의 상호 연관성 추적과 변경 관리는 계층적 시스템 명세의 구조 관계<그림5>를 통해 알 수 있다.



<그림 9> 특정 모듈 삭제 후의 시뮬레이션 결과

7. 결 론

본 연구에서는 보안시스템인 침입탐지시스템의 효과적인 계발을 위해 시스템 형식론에 의거한 소프트웨어 설계 방법론을 적용하였다. 형식 명세에 의해 침입탐지시스템의 인터페이스 시스템을 구조적 입출력 시스템의 요소들로 표현하고, 이를 관계 사상에 의해 입출력 시스템으로 변환한 다음 DEVS 모델링에 의해 DEVS 모델로 재구성하였다. DEVS 시뮬레이션 환경에서 제공하는 시뮬레이터를 통하여 적용 대상 시스템인 침입 탐지 시스템의 중요한 동적 특성을 소프트웨어 설계시 또는 설계 변경 후 미리 파악할 수 있었다.

제안된 보안시스템 개발로의 소프트웨어 설

계 방법론 적용은 다음과 같은 주요 장점을 얻을 수 있다.

1)소프트웨어 개발 시에 전체 시스템의 구조나 흐름을 파악하고 프로그램의 이해를 높여 시스템 성능 향상을 얻을 수 있다.

2)보안 시스템의 확장에 의한 인터페이스 시스템의 기능 추가, 변경, 삭제 등이 올수 있고 이에 따른 전체 시스템에 미치는 영향을 신속하고 정확하게 파악 할 수 있다.

3)선정된 보안정책이 모델에서 잘 표현 되었는지 검증(verification)이 가능하다. 이는 예기치 못한 프로그램 버그를 미연에 방지 할 수 있다.

본 연구에서는 DEVS 모델로부터 각각 구조적 입출력 시스템, 입출력 시스템, 인터페이스 시스템의 중요한 동적 특성을 수동으로 추적 또는 파악하였지만 향후 DEVS 모델과 인터페이스 시스템을 연결하는 자동 추적 기능을 갖는 시스템의 설계와 구현이 이루어져야 한다.

참고 문헌

[1] 장덕철, 박장한, "C++ 코드로부터 클래스 관련 정보 생성 도구의 설계 및 구현", 정보처리학회 논문지 제7권 제3호, 2000.

[2] 김덕현, 박성주, "확장된 객체지향 데이터 모델을 이용한 소프트웨어 변경 관리 시스템", 정보과학회 논문지, Vol 22, No 2, pp.249-260, 1995.

[3] Booch, G., "Object-Oriented Analysis and Design with Applications", 2nd Edition, Addison Wesley Longman, Inc, 1994.

[4] Joel Scambry, "HACKING EXPOSED 2nd Ed. : Network Security Secrets & Solutions", McGraw-Hill, 2001.

[5] F. Cohen, "Simulating Cyber Attacks, Defences, and Consequences", Computer & Security, Vol.18, pp. 479-518, 1999.

[6] 김은하, 조대호, "시스템 형식론에 의한 사용자 인터페이스 시스템 표현과 DEVS 모델링", 한국시뮬레이션학회 논문지, Vol 8, No 4, 1999.

[7] Bernard.P. Zeigler, Herbert Praehofer and T.G Kim,"Theory of Modelling and Simulation", 2nd Ed., Academic Press, 2000.

[8] L.Padulo and M.A.Arvbib, System Theory, Suders, Philadelpia, Pa, 1974.

[9] Bernard.P.Zeigler, "Multifacetted Modelling and Discrete Event Simulation Orlando", FL, USA: Academic, 1984.

[10] Bernard.P.Zeigler, "Object-Oriented Simulation with Hierarchical, Modular Models", San Diego, CA, USA: Academic Press, 1990.

[11] 조대호, 이칠기, "계층의 구조를 갖는 시뮬레이션 모델에 있어서 단계적 접근을 위한 모델 연결 방법론과 그 적용 예", 한국시뮬레이션학회 논문지, Vol 5, No 2, 1996.

[12] 김상근, "객체지향 기술에 의한 사용자 인터페이스 구축기 개발", 중앙대학교 석사학위 청구논문, 1995.

[13] 허계범, 최영근, "객체지향 소프트웨어 공학", 한국실리콘, 1995

[14] G. Booch, "Object-Oriented Design", Bengamin/Cummings Publishing, 1989.

[15] J. Raumbaugh et al., "Object-Orented Modeling and Design Prentice-hall",Inc.,

1991.

[16] S. Shaler and S. Meller, "Object-Oriented Systems Analysis: Modeling the World in Data", Englewood, Cliffs, NJ: Yourdon Press, 1988.

[17] R. Bace, "Intrusion Detection", Macmillan Technical Publishing, 2000.

[18] 김형종, "지식 기반 시뮬레이션 환경을 사용한 분산침입탐지 시스템의 계층적 모델링", 성균관대학교 박사학위 청구논문, 2000.

[19] K. Llgun, R. A. Kemmerer, and P.A.Porras, "State Transition Analysis: A Rule-based Intrusion Detection Approach", IEEE Transactions on Software Engineering, 1995.