Journal of Korea Multimedia Society Vol. 8, No. 2, February 2005(pp. 233-248)

침입감내 요구사항을 수용한 응용 소프트웨어 개발

조은숙[†], 김철진[††], 이숙희[†††]

요 약

알려지지 않은 취약점이나 공격 방법에 대한 침해 사고를 방지하기 위해 침입감내 기술(Intrusion Tolerance Technology)이 이에 대한 한 가지 해결책으로 제시되고 있다. 그러나 실제 침입 감내 기술을 기반으로 한 침입 감내 기반의 응용 소프트웨어를 설계하고 개발하는 데 있어서는 아직도 체계적인 모델링 기법을 적용하고 있지 못하고 있다. 특히 가용성, 무결성, 신뢰성, 기밀성 등과 같은 항목들은 침입 감내 기반 응용 소프트웨어에 있어서 중요한 요구사항들이다. 그럼에도 불구하고 현재 대부분의 UML 기반의 모델링 기법에서는 이러한 요구사항들을 반영하여 설계하는 기법을 제시하고 있지 못하고 있다. 따라서 본 논문에서 이러한 취약점들을 인지하고 침입 감내 소프트웨어 개발에 있어서 침입 감내와 관련된 요구사항들을 체계적으로 반영하여 적용할 수 있는 프로파일과 설계 기법을 제시하고자 한다. 이는 침입 감내 기반의 응용 소프트웨어 개발을 위한 UML의 확장 뿐만 아니라 품질 향상을 가져올 수 있다.

A Development Technique for Application Software Based on Intrusion Tolerant Requirements

Eun Sook Cho[†], Chul Jin Kim[††], Sook Hee Lee[†††]

ABSTRACT

An intrusion tolerant technology has been introduced as a solution to prevent intrusion accident for unknown fragility or attack. However, a systematic modeling technique is not applied into a system design and development based on intrusion tolerant technology. Especially, elements such as availability, integrity, reliability, confidentiality, and so on are important requirements in intrusion tolerant system. Nevertheless, current most of UML-based modeling techniques pass over or don't provide design techniques reflecting those requirements. Therefore, we know these weaknesses and propose both profile and design technique reflecting and applying intrusion tolerant requirements systematically in the development of application software based on intrusion tolerance. We expect that proposed technique can extend not only current UML's limitations but also can improve the quality of application software based on intrusion tolerance.

Key words: Weakness(취약점), Intrusion Tolerance(침입 감내), Availability(가용성), Confidentiality (기밀성), UML

1. 서 론

최근에 보고 되는 침해 사고들은 점점 더 복잡하고 교묘한 공격 방법들을 사용하여 피해 규모를 키워가고 있다. 또한 침해 사고의 원인이 되는 시스템, 네트워크, 소프트웨어의 취약성들이 연간 3000여개 이상 발견되는 등 잠재적인 침해 사고의 원인들도 매우 급속하게 증가하고 있다. 이러한 침해 사고들을

※ 교신저자(Corresponding Author) : 조은숙, 주소 : 서울특별시 성북구 하월곡2동(136-714), 전화 : 02)940-4595, FAX : 02)940-4194, E-mail : escho@dongduk.ac.kr
접수일 : 2004년 10월 26일, 완료일 : 2005년 2월 25일
[†] 동덕여대 정보학부 데이터정보학과 전임강사
[††] 삼성전자 책임연구원
(E-mail : chujin777.kim@samsung.com)
[†††] 정회원, 서경대학교 인터넷정보학과 교수
(E-mail : oleesh@imail.skuniv.ac.kr)

예방하고 효과적인 대응 방법을 마련하기 위하여 침입차단기술, 침입탐지기술 등의 다양한 정보보호기술들이 개발되고 있다. 이런 기술들은 알려진 취약점에 대한 공격 예방과 탐지를 제공하며, 알려지지 않은 취약점에 대한 공격에는 적절하게 대응하지 못하는 단점이 있다. 그러므로 이와 같이 알려지지 않은 취약점이나 공격 방법에 대한 침해 사고를 방지하기 위한 기술이 필요하며 침입 감내 기술(Intrusion Tolerance Technology)이 이에 대한 한 가지 해결책으로 제시될 수 있다[1,2].

침입 감내 기술이란 중요한 서비스를 제공하는 시스템에 대한 공격이 발생하더라도 정상적인 서비스를 제공할 수 있도록 하는 기술이다. 이러한 침입 감내 기술은 결함허용 기술과 침입차단 기술이나 탐지기술 등의 정보보호 기술들이 결합된 형태로 미국과 유럽에서 비교적 최근에 시작되었다. 침입 감내 기술에 대한 연구는 오래전부터 수행되어 왔으나 많은 연구자들이 집중적으로 연구를 시작한 것은 비교적 최근의 일이다. 유럽에서는 FP5의 IST 프로그램을 통하여 관련 연구를 진행하였고, 미국에서도 DARPA (Defense Advanced Research Project Agency)의 지원을 통하여 연구들을 진행하고 있다.

그러나 현재 이러한 침입 감내 기술을 적용한 시스템 설계에 대한 연구는 거의 이루어지지 않은 실정이다. 본 논문에서는 이러한 침입 감내 시스템을 위한 개발 방법론을 제안하고 UML의 확장 메커니즘을 적용하는 기법을 제시한다.

본 논문은 다음과 같이 구성된다. 2장에서는 DARPA 프로젝트의 일환으로 연구된 침입 감내 기술과 기존의 개발 방법론을 살펴본다. 3장에서는 침입 감내 시스템을 구축하기 위한 개발 방법론을 제시한다. 4장에서는 사례 연구를 통해 제안된 방법론을 시스템 설계에 반영하기 위한 UML 적용기법을 제시한다. 5장에서는 본 논문에서 제시한 방법론과 기존 방법론들과의 비교 평가 결과를 제시하고, 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

2.1 침입 감내 기술

2.1.1 ITUA

ITUA 프로젝트는 200년 7월에 시작해서 2003년 12월에 종료되는 프로젝트로서[3], ITUA의 주된 특징으로는 두 가지를 들 수 있다. 첫째는 혼합 결합 복제 메커니즘이고 둘째는 예측 불가능에 대한 대응 메커니즘이다. 또 다른 특징으로는 ITUA는 여러 그룹들 간 통신이 가능하다.

ITUA 침입 모델은 4가지 요소인 항목(item), 행동

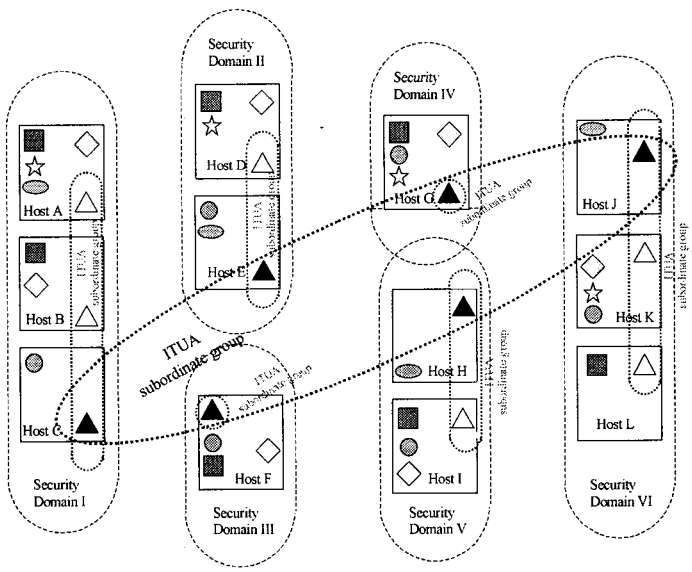

그림 1. ITUA 아키텍처

(action), 가정(assumption), 그리고 명세로 구성되어 있다. 또한 ITUA 아키텍처는 복제자(replica), 관리자(manager), 그리고 종속자(subordinate)들로 구성되어 있다. ITUA는 이러한 컴포넌트들로 구성되어 있으면서 in-band 와 out-of-band 적응을 통해 지원하는 여러 종류의 적응들을 관리하기 위해 분산화된(decentralized) 인프라스트럭처를 구현하고 있다[4]. 각각의 호스트는 하나의 관리자나 하나의 종속자를 실행할 수 있다. 각각의 보안 도메인(security domain)은 하나의 관리자를 실행하는 하나의 호스트를 갖는다. 따라서 결과적으로 동일 도메인에 있는 다른 호스트들은 종속자들을 실행하게 된다. 모든 관리자들은 "관리자 그룹"이라고 하는 하나의 그룹을 형성하게 된다. 한 도메인에 있는 종속자들과 그 관리자는 하나의 "종속자 그룹"을 형성한다. 종속자는 "보안 권고(security advising)"와 "복제 관리(replication management)"라는 두 가지 의무를 갖는다.

2.1.2 HACQIT

HACQIT(Hierarchical Adaptive Control of Quality of service for Intrusion Tolerance) 프로젝트는 UC Davis와 Teknowledge사의 협력 하에 진행하고 있는 것으로 공격 시에 사용자 성능 25% 이상의 저하를 방지하면서 4 시간 동안의 침입 감내 제공을 목표로 하고 있다. 또한, 비침입 감내 COTS/GOTS 하드웨어와 소프트웨어에게 일정한 서비스의 시발점을 제공하며, 사용자로부터 생성되거나 사용된 주요 데이터의 기밀성과 무결성을 보호하는 것을 목표로 하고 있다[5].

그림 2는 HACQIT 아키텍처를 나타내고 있다. 주요한 어플리케이션들, 주요한 서버들 그리고 다른 서버(sandbox)들이 함께 보호된 서브 도메인 내에 있는 것을 강조하여 설명하고 있다.

HACQIT 시스템 내의 두 대의 서버, 주 서버와 백업 서버가 제공하는 서비스는 침입 차단 기능을 수행하는 게이트웨이를 통해서 사용자와 연결된다. 이들을 모니터하고 제어하는 모니터 & 어댑터는 결함 진단 소프트웨어를 포함하고 있다. 그림에서 점선으로 나타낸 연결은 별도의 out-of-band 네트워크로 구성되어 일반 사용자가 접근할 수 없도록 되어 있다. 샌드 박스(sand box)는 주 서버 및 백업 서버의 데이터 복제를 가지고 있어서, 두 서버의 결과가 일치하지 않는 경우 일시적인 문제인지, 공격에 의한 것인지를 파악하는데 이용된다. HACQIT은 오류 검출과 실패 차단을 위해 중복성과 다양성을 복합적으로 이용한다.

HACQIT 아키텍처의 샌드 박스는 강요성을 제공하며 결함 발생시 복제된 데이터나 서비스를 통해 신뢰성을 제공한다. 모니터와 어댑터는 요청이나 들어오는 패킷에 대한 무결성을 검사해주는 기능을 포함한다. HACQIT은 센서를 통해 침입을 감지하여 공격이 전파되는 것을 최소화시킬 수 있기 때문에 신뢰성과 무결성을 제공한다. 게이트웨이와 방화벽을 통해 기밀성과 무결성을 제공한다.

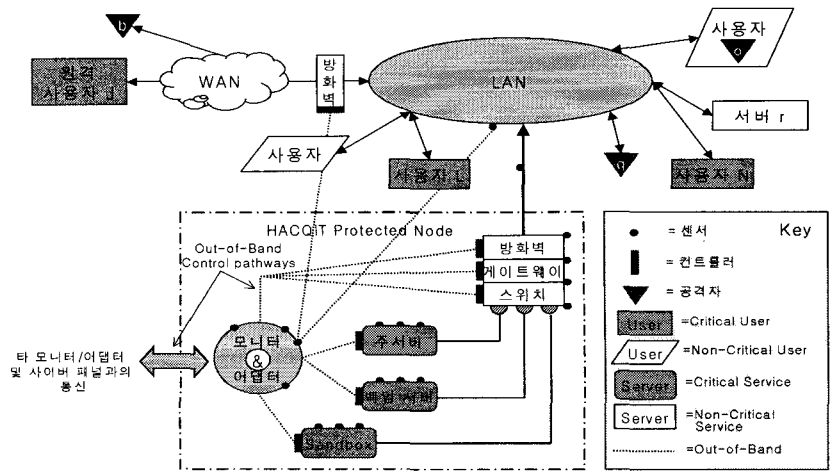

그림 2. HACQIT 아키텍처

2.1.3 SITAR

SITAR(Scalable Intrusion_tolerance Architecture)는 MCNC사와 Duke대학에서 분산 서비스를 제공하는 고가용성의 침입 감내 시스템을 위한 확장성 있는 아키텍처 및 프레임워크이다[6]. SITAR의 접근 방법에서는 침입에 대한 관점을 공격자와 공격들 자체로부터 보호해야 할 대상, 즉 본질적으로 제공하는 기능들과 서비스들로 이동시켰다. 이것은 위협의 주체가 악의적인 공격이거나, 자연적인 실패이거나, 혹은 인위적인 사고이거나를 결정하기 이전에 그것에 대한 영향에 대항하여 시스템이 지속적으로 살아남을 수 있도록 한다는 것이다[7]. 그림 3은 SITAR의 아키텍처이다.

SITAR는 적응 재설정 모듈(Adaptive Reconfiguration Module)을 통해 기밀성과 가용성 서비스를 제공한다. 프락시 서버 또한 COTS 서버와 사용자의 직접적인 접근을 막고 침입을 감내하기 위한 서비스를 제공하여 높은 가용성을 보장한다. 투표 모니터(Ballot Monitor)는 처리된 서비스의 결과 값, 즉 응답 값을 최종적으로 결정하여 높은 신뢰성을 지원한다. 응답 모니터를 통해 무결성을 제공하며, 감시 제어를 통해 기밀성을 제공한다.

2.1.4 MAFTIA

MAFTIA(Malicious-and Accidential-Fault Tolerance for Internet Applications)는 Information Society Fifth Framework Programme 의 지원으로 대규모 분산 시스템에서 우발적인 결함(accident faults) 과 악의적인 공격(malicious attacks)을 감내하기 위하여 포괄적인 접근방법의 조사가 유도된 프로젝트이다[8,9]. MAFTIA는 사람의 중재가 없어도 공격 중에서 시스템이 이전과 같이 작동하는 것을 가능하게 하고, 결함 감내, 분산 컴퓨팅, 암호 해독법, 정형적인 증명, 컴퓨터 보안과 침입 탐지 공동체들로부터 중요한 전문 지식들을 가져온다.

MAFTIA의 아키텍처는 그림 4에서 보는 바와 같이 로컬 토폴로지(local topology)와 모듈간의 의존성 관계(화살표 방향으로 나타낸다)를 표현하고. Site 계층과 Participant 계층으로 분류된 계층을 보여준다.

2.2 현존 설계 기법의 한계점

UML 기반의 시스템 설계 기법으로는 국내외적으로 다양한 방법론들이 제시되어 소개되고 있다. 대표적인 방법론으로 Catalysis[10], Fusion[11], Uniface [12] 등과 같은 외국 방법론과 국내에서는 마르미 III[13]가 있다. 그러나 이러한 방법론들은 일반적인 비즈니스 응용 소프트웨어 설계와 관련된 지침이나 기법들은 제시하고 있지만, 침입 감내 시스템과 같은 보안과 관련된 소프트웨어에 그대로 적용하기에는

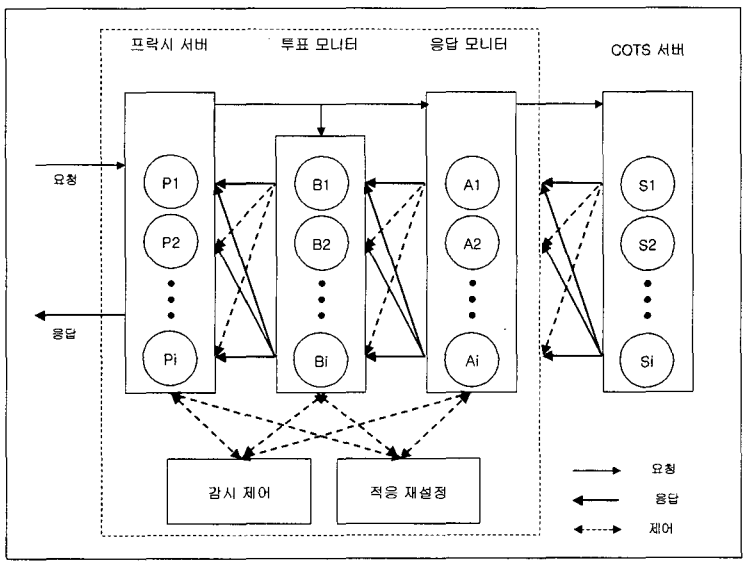

그림 3. SITAR의 아키텍처

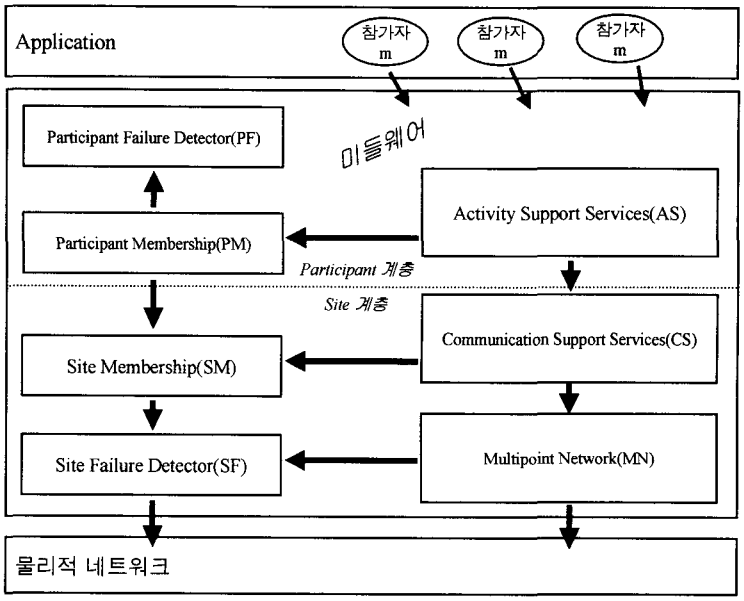

그림 4. MAFTIA 아키텍처

여러 가지 한계점을 가지고 있다.

첫째, 침입 감내 시스템이 요구하는 비기능적인 요소 혹은 특성들을 반영할 수 있는 설계 요소들을 제시하고 있지 않다. 따라서 현존하는 방법론을 그대로 적용해서 시스템을 설계하는 경우에 있어서는 이러한 비기능적인 요소들이 설계에 반영되어 표현할 수 없다.

둘째, 침입 감내 시스템은 응용 소프트웨어를 지원하는 미들웨어 성격의 소프트웨어라 할 수 있다. 따라서 이러한 미들웨어와 같은 시스템 소프트웨어의 경우에 있어서는 소프트웨어의 아키텍처가 시스템의 성능이나 안전에 있어서 매우 중요한 영향력을 가질 수 있다. 그러나 기존의 설계 기법들은 응용 소프트웨어의 아키텍처에 주로 집중되어 있어서 미들웨어와의 관련성에 대한 부분들을 아키텍처에 많이 고려하고 있지 못하다.

셋째, 현존 UML의 모델링 요소들을 보면 침입 감내 시스템이 갖는 특징들을 표현할 수 있는 장치들이 제시되어 있지 못하다. 따라서 UML을 기반으로 하는 현존 설계 방법론들 또한 이러한 모델링 장치들이 제시되어 있지 못하다. 침입 감내 시스템 설계에 있어서 침입 감내 시스템이 갖는 특징들이 설계에 반영될 수 없다면, 고품질의 침입 감내 시스템이 개발될 수 없다.

3. 침입 감내 기반의 응용 소프트웨어 개발 기법

이 장에서는 2장에서 제시한 현존하는 대표적인 침입 감내 시스템의 아키텍처와 구성 요소, 그리고 현존 설계 기법들이 갖는 문제점을 해결하기 위해 새로운 침입 감내 기반의 응용 소프트웨어 개발 기법을 제시한다. 특히, 현존 침입 감내 시스템들의 분석 결과를 바탕으로 UML에 침입 감내와 관련된 모델링 장치를 확장한 UML-IT Profile 또한 제시함으로써, 침입 감내 기반의 응용 소프트웨어 설계를 용이하도록 지원한다.

3.1 프로세스 개요

본 방법론에서 제안하는 프로세스는 크게 4개의 단계인 계획, 분석, 설계, 구현 단계로 구성되어 있으며, 각각의 단계에는 활동, 지침, 입력물, 산출물 등으로 구성되어 있다. 그리고 본 방법론에서 제안하는 프로세스 모형은 점진적이면서 반복적인 모형을 따르게 된다. 그림 5에 제시된 것처럼 계획 단계에서 일부는 구현 단계를 수행할 수도 있다. 이는 계획 단계에서 개발하고자 하는 기술의 검증 등을 위해서 프로토타입을 개발하고자 할 경우에 일부 구현 단계를 수행할 수 있다는 것을 의미한다. 또한 설계 단계

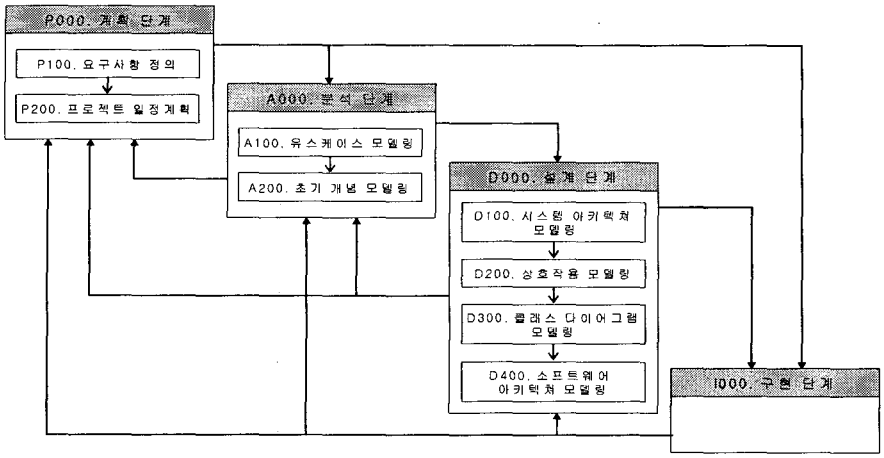

그림 5. 전체 프로세스

에서 계획 단계나 분석 단계로 돌아가서 일부 계획의 변경 및 조정을 할 수도 있다.

계획 단계는 전체 개발 계획을 수립하는 단계로서, 개발에 필요한 요구사항들을 도출하여 정의하는 과정이나 개발에 드는 비용, 개발에 투여할 인력, 개발 일정 등을 수립하는 일 등을 한다. 분석 단계는 계획 단계에서 정의된 요구사항 명세서를 기반으로 개발할 어플리케이션에 대한 기능 분석이나 구조 분석 등을 통하여 전체 시스템이 필요로 하는 기능적 요구사항에 대한 검증과 시스템의 내부 구조, 그리고 시스템의 작업 흐름 등을 파악하게 된다. 설계 단계는 분석 단계의 산출물들을 바탕으로 분석 단계 산출물을 좀더 상세화 하는 단계이다. 이 단계에서 개발할 시스템의 소프트웨어 아키텍처를 설계하게 되고, 시스템의 내부 구성 요소들 간의 상호 작용 등을 상세하게 설계하게 된다. 또한 침임 감내 시스템의 경우에는 하부 침입 감내 서비스들과 상위 어플리케이션의 서비스들 간의 상호 작용 등을 아키텍처 설계에 반영하게 된다. 구현 단계는 특정 언어나 플랫폼을 기반으로 하여 소프트웨어를 개발하는 단계를 의미한다. 이 단계에는 설계 단계의 산출물들을 기반으로 해당 구현 언어나 플랫폼을 적용하여 개발하는 것이다.

이 장에서는 본 연구에서 제안한 침입 감내 시스템을 위한 개발 방법론을 분석 단계와 설계 단계를 중심으로 상세 절차를 제시한다.

3.2 분석 단계

분석 단계는 계획 단계를 통해 도출된 기능적 요구사항과 비기능적 요구사항을 기반으로 유스케이스 모델링과 초기 개념 모델링을 수행한다. 유스케이스 모델링을 통해 목표 시스템의 기능성을 분석하고 초기 개념 모델링을 통해 시스템의 정적인 구조를 분석한다.

3.2.1 유스 케이스 모델링

유스 케이스 모델링은 시스템에서 제공하는 기능성을 사용자의 입장에서 분석하는 활동이다. 요구 사항 명세서에 명시되어있는 요구 사항에 따라 시스템의 기능성을 분석하고 침입 감내 요구 사항을 분석하여 결과물에 반영한다. 본 방법론에서는 기능 분석을 위해 UML의 유스케이스 다이어그램을 사용한다. 첫번째로, 시스템을 사용하게 되는 액터를 식별하고, 시스템의 기능성 추출을 위해 유스 케이스를 식별한다. 그리고 난 후, 유스 케이스 간의 관계를 식별하고, 이러한 결과를 토대로 유스 케이스 다이어그램을 정제한다. 유스케이스 다이어그램 정제는 작성된 다이어그램에서 누락된 사항이나 변경된 사항을 수정하고 유스 케이스가 동일한 수준의 굵기(granularity)를 갖도록 조정하는 활동이다. 이처럼 기능적인 측면을 식별한 후, 침입 감내와 관련된 요구사항을 식별한다. 이는 작성된 유스 케이스 다이어그램에 요구사항 명세서에 명기된 비기능적인 요구 사항을 반영하는 활동이다. 주로 유스케이스에 침입 감내 요구사항을 표기한다. 특히 침입 감내 요구 사항을 분석 모델에 반영하기 위한 UML의 표준 표기법이 없어 UML의 확장 메커니즘을 이용하여 침입 감내 시스템을

위한 UML-IT Profile을 정의하여 분석 모델에 이를 사용한다[14].

마지막으로 유스 케이스 명세서를 작성한다. 이는 시스템 내부의 작업 흐름이 아닌 사용자와 시스템 간에 어떠한 상호 작용을 통해 유스케이스를 수행하는지 관찰하고자 하는데 목적이 있으며, 명세서에는 식별된 침입 감내 메커니즘에 따라 추가적인 사건 흐름을 추가하여 기술한다. 그림 6의 유스 케이스 다이어그램은 시스템의 기능적인 요구사항과 침입 감내 요구사항을 설명하기 위한 다이어그램이다. 본 방법론에서 제공하는 표기법에 따라 메커니즘을 기술하여 유스케이스에 침입 감내 요구 사항을 반영한다.

3.2.2 초기 개념 모델링

초기 개념 모델링 활동은 실세계에 존재하는 추상화 개념을 식별하는 작업을 기초로 해서 시작된다. 초기 개념 모델링 활동에는 개념적인 객체를 식별하기 위해서 강건성 분석(robustness analysis) 방법을 이용한다. 작업 절차는 다음과 같이 수행한다. 우선 클래스 식별을 한다. 클래스 식별 작업은 강건성 분석을 통해서 수행된다. 이 작업을 통해서 산출되는 클래스들은 시스템의 기능적, 비 기능적 요구사항을 반영할 뿐만 아니라, 시스템의 기능을 지원하는 하위 기능들도 나타낸다. 강건성 분석은 시스템의 내부적인 구조나 행위를 분석하는 것이 용이하고, 초기 개념들을 좀더 정확하게 모델링 하는 것을 가능하게 한다. 다음으로 요구 사항 명세서를 분석하여 클래스들이 어떤 속성을 포함하는지를 결정한 후, 클래스간의 관계를 식별한다. 이러한 결과를 바탕으로 개념적 클래스 다이어그램을 작성한다. 그림 7은 개념적 클래스 다이어그램 표현 양식이다.

3.3 설계 단계

설계 단계에서는 분석 단계까지 도출된 산출물들을 통해 시스템을 이해하여, 초기 시스템 아키텍처를 모델링하고 상호 작용 모델링과 클래스 다이어그램 작성을 하면서 어플리케이션을 설계한다. 최종 단계에서는 초기 시스템 아키텍처를 정제하여 어플리케이션과 미들웨어에 중점을 둔 소프트웨어 아키텍처를 작성한다.

3.3.1 시스템 아키텍처 모델링

시스템 아키텍처는 수집된 요구사항과 분석된 자료를 바탕으로 하여, 앞으로 설계 및 구현될 시스템에 대해 개괄적인 구조를 모델링 하는 것으로서 시스템의 기능적인 측면뿐만 아니라 그에 수반되는 비기능적 측면의 시스템 요구 사항들이 모두 반영되어야 한다. 이는 단순히 어플리케이션 설계상으로만 제공할 수 있는 정도의 품질을 넘어서 더욱 높은 품질의 시스템을 구축할 수 있도록 해준다.

첫번째로, 아키텍처 스타일을 선정하는 것이다. 한 가지 뷰 만을 가지고 전체 시스템을 표현할 수는 없

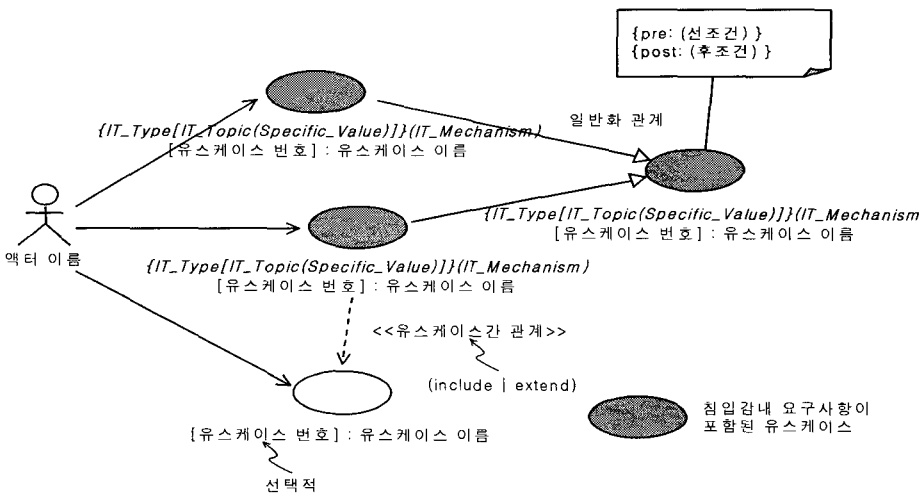

그림 6. 유스 케이스 다이어그램 양식

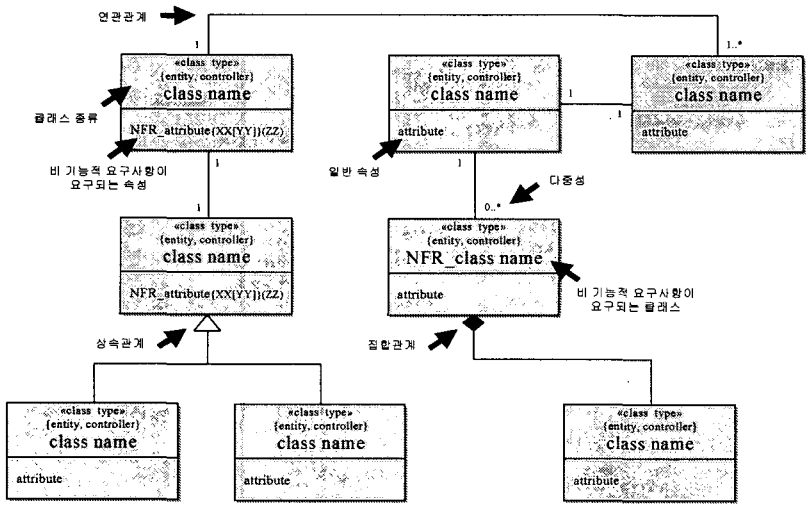

그림 7. 개념적 클래스 다이어그램

다. 따라서 다양한 측면의 뷰를 제공할 수 있도록 하나 이상의 아키텍처 스타일을 사용하게 되며, 이때 시스템 관련자들의 관심과 요구 사항에 따라 서로 다른 아키텍처 스타일이 요구된다. 두번째로는 선정된 아키텍처 스타일에 포함될 요소들을 식별한다. 요구 사항 명세서와 유스 케이스 명세서 및 유스 케이스 다이어그램, 그리고 개념적 클래스 다이어그램을 참고하여 각 스타일에 맞는 요소들을 식별한다. 식별되는 요소들은 상세한 클래스 단위가 아니라 업무나 큰 기능 단위의 요소들이다. 세번째로는 식별된 각 아키텍처 스타일의 요소들이 가지는 연관 관계를 식별하고 아키텍처에 모델링 한다. 연관 관계는 요구사항 명세서의 아키텍처에 대한 비 기능적 요구 사항이나 개념적 클래스 다이어그램, 그리고 시퀀스 다이어그램에서 쉽게 파악할 수 있으며, 유스 케이스 다이어그램에서도 업무 및 기능 단위의 연관성이 식별될 수 있다. 최종적으로는 수행된 결과물을 검토하는 단계이다. 즉, 여러 시스템 관련 관계자들이 필요로 하는 아키텍처들이 적합하게 표현되어 있으며 그들의 요구 사항이 잘 반영되어 있는지를 검토한다.

3.3.2 상호 작용 모델링

상호 작용 모델링은 시스템이 제공하는 기능성을 수행하기 위한 객체간 메시지 흐름을 설계하는 활동이다. 즉, 목표 시스템의 행위적인 측면을 설계하는 것으로 시스템을 구성하게 되는 객체 간에 어떠한 메시지 전달의 집합을 통해 시스템의 업무를 제공하게 되는지를 구현 전에 미리 계획하는 활동이다. 구현에 앞서 상호 작용 모델링을 통해 로직을 검증하고 본 방법론에서는 UML의 시퀀스 다이어그램을 사용하여 상호 작용을 가시화하고 문서화한다.

상호 작용 모델링의 첫 번째 작업은 유스 케이스를 선택하는 것이다. 유스 케이스 다이어그램에서 식별된 유스 케이스 중 상호작용 모델링을 수행하고자 하는 유스 케이스를 선택한다. 다음으로 이전 작업에서 선택한 유스 케이스를 실행시키는 액터를 식별하여 그린다. 액터는 유스 케이스 다이어그램과 유스 케이스 명세서에 명시되어 있다. 그런 다음에, 클래스를 식별한다. 클래스 식별은 선택된 유스 케이스의 기능 수행에 참여하는 객체들을 식별하는 작업이다. 바운더리, 컨트롤러, 엔티티, 미들웨어 및 기타 추가적인 객체들이 기능 수행에 참여하게 된다. 일반적으로 관심 분리(Separation of concern)의 원리에 따라 어플리케이션을 사용자 인터페이스 계층, 컨트롤 계층, 데이터 계층으로 나누어 객체의 변경이나 유지보수의 관리를 용이하게 한다. 사용자 인터페이스 계층은 사용자와의 직접적인 상호 작용을 담당하는 계층으로 바운더리 클래스가 이 계층에 속하게 된다. 컨트롤 계층은 사용자 인터페이스 계층으로부터 받은 요청이나 데이터를 데이터 계층에게 전달하거나 중간에서 중재하는 역할을 담당하는 계층으로 컨트롤러 클래스가 이에 속한다. 데이터 계층은 업무 도메

인의 핵심적인 기능을 제공하거나 데이터의 영구성을 관리하는 계층으로 엔티티 클래스가 이에 속한다. 엔티티 클래스는 주로 하위의 미들웨어 클래스의 기능을 호출함으로써 트랜잭션, 보안, 침입 감내와 같은 시스템 서비스를 제공받게 된다. 따라서 액터는 직접적으로 데이터 계층의 클래스를 접근하지 않고 바운더리 및 컨트롤 클래스를 거쳐 도메인의 핵심 클래스를 접근하다.

마지막으로 메시지 흐름을 식별한다. 메시지 흐름 식별은 선택된 유스 케이스에 대해 유스케이스의 기능을 제공하기 위해 추출된 액터와 추출된 클래스들 사이에서 시간적인 순서로 어떠한 메시지를 주고받는지를 식별하는 작업이다. 메시지의 흐름은 일반적으로 유스 케이스 명세서에 작성된 주 흐름, 부 흐름 및 예외 흐름을 보고 작성한다. 유스 케이스 명세서에서 작성된 각각의 흐름은 시퀀스 다이어그램에 필수적으로 나타나야 한다. 그러나 유스 케이스 명세서보다는 시스템 내부에서의 객체들 간의 상호작용이 상세히 기술되어야 한다. 그림 8은 시퀀스 다이어그램에 나타나는 객체 양식이다.

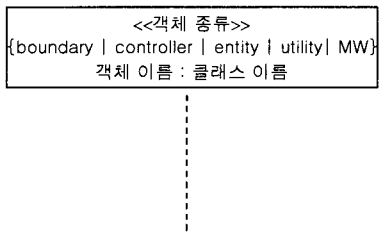

그림 8. 객체 양식

객체에 대한 양식은 위와 같지만 시퀀스 다이어그램에는 일반적으로 계층별로 표기하여 액터, 바운더리 객체, 컨트롤러 객체, 엔티티 객체, 추가적인 객체, 미들웨어 객체의 순서로 작성한다. 시퀀스 다이어그램에 대한 양식은 아래 그림 9와 같다. 기존의 시퀀스 다이어그램에 비해 침입 감내와 관련된 요소들이 반영되어 표현되게 되었다.

3.3.3 클래스 다이어그램 모델링

클래스 다이어그램 모델링 활동은 비즈니스 업무 관점에서 개발된 초기 개념 모델링을 기반으로 하여 시스템의 관점에서 필요한 클래스를 정제 혹은 추출하는 활동이다. 뿐만 아니라, 구현 환경에 종속적으로 클래스를 모델링하게 된다. 따라서 초기 개념 모델링과는 달리 클래스 다이어그램을 모델링 할 때에는 다음과 같은 사항을 준수해야한다. 첫째, 클래스 다이어그램을 명세하기 위해서 사용되는 언어는 구현언어와 동일하다. 둘째, 클래스의 속성과 가시성을 표현한다.

첫번째 작업은 클래스 정제이다. 클래스 식별작업에서는 초기 개념모델링 활동에서 식별된 클래스 외에 추가되어야 할 클래스가 있는지 살펴본다. 두번째로는 속성 정제이다. 속성 정제 활동에서는 설계된 클래스에서 요구되는 속성들을 식별하고, 식별된 속성들을 구현언어에 맞는 문법으로 명세한다. 세번째로는, 오퍼레이션 정제이다. 오퍼레이션 정제 활동에서는 설계된 클래스에서 제공해야 하는 오퍼레이션을 식별하고, 구현언어의 문법에 맞게 오퍼레이션의 시그너쳐를 명세한다. 네번째로는 관계 정제이다. 관

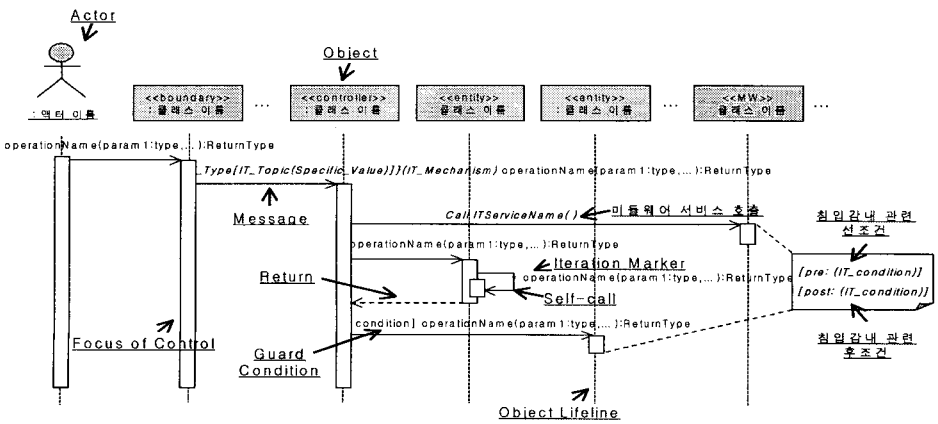

그림 9. 시퀀스 다이어그램 양식

계 정제에서는 초기 개념 모델링 활동에서 식별된 관계 및 관계와 관련된 부가적인 정보들을 설계 관점에서 상세히 기술하는 작업을 수행한다. 마지막으로, 상세 클래스 다이어그램 작성이다. 앞의 활동을 통해서 클래스 다이어그램의 모든 요소가 식별되면, 정제된 클래스간의 관계를 통해서 상세 클래스 다이어그램을 작성한다. 상세 클래스 다이어그램의 양식은 그림 10과 같다.

3.3.4 소프트웨어 아키텍처 모델링

소프트웨어 아키텍처는 이전에 작성되었던 시스템 아키텍처에서 어플리케이션과 미들웨어에 관련된 부분들을 정제하는 단계이다. 수집된 요구 사항과 분석 자료들만을 통해 개괄적으로 표현했던 시스템 아키텍처를 설계된 클래스 다이어그램과 시퀀스 다이어그램 등을 통해 더욱 상세하게 표현하게 된다.

작업 절차로는 첫번째로 아키텍처의 요소 정제이다. 이는 시스템 아키텍처의 업무 및 기능 단위로 표현된 요소들을 설계된 클래스다이어그램을 참고하여 상세화 한다. 두번째로는 아키텍처의 요소간 연관 관계 정제로서, 이는 시스템 아키텍처 상에 존재하는 업무 단위 및 기능 단위 요소들 간의 연관 관계를 상세 수준 요소의 연관 관계로 정제한다. 이 때 클래스 다이어그램에서 그룹 단위의 클래스들이 그룹 내부 혹은 외부 그룹 클래스들과의 연관 관계를 참고하며, 시퀀스 다이어그램에서도 동일한 방법으로 참고하도록 한다.

4. 사례 연구

이 장에서는 3장에서 정의한 침입 감내 시스템 설계를 위한 방법론의 개발 프로세스와 업무에 따라 은행업무의 수신관리 시스템에 대한 사례를 현존 UML을 확장한 UML-IT 프로파일(Profile)을 분석 및 설계 단계에서 적용한다. 특히 침입 감내와 관련된 요구사항은 소프트웨어의 기능적인 요구사항 보다 비기능적인 요구사항에 해당하기 때문에 이를 표현하기 위해 UML-IT 프로파일을 개발하여 정의하였다. 이러한 비기능적인 요구사항의 표기법은 ***{IT_Type[IT_Topic(Specific_Value)]}(IT_Mechanism)***와 같이 한다.

여기서 IT_Type은 비기능적인 요구사항의 종류를 *IT_Type* : 비기능적인 요구사항의 종류를 의미하는데, 비기능적인 속성의 종류로는 가용성, 신뢰성, 무결성, 기밀성이 해당된다.

IT_Type은 표 1에 정의된 IT_Type을 사용하며, IT_Topic은 IT_Type의 세부 속성으로서, 가용성, 신뢰성, 무결성, 기밀성의 세부 속성들인 파일 기밀성, 메시지 기밀성, 어플리케이션 메타 정보 기밀성 등이 해당된다. 이러한 IT_Topic은 표 1에 정의되어 있으며, 적합한 세부 속성이 없을 경우에는 침입 감내 요구사항 전문가가 식별하여 표 1을 확장하여 표기한다. Specific_Value는 IT_Topic에 대한 구체적인 값이나 조건으로서 이는 선택사항이다.

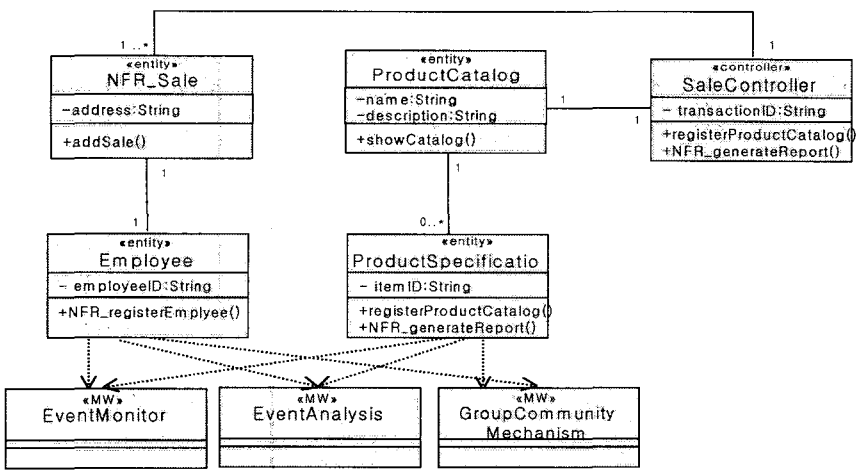

그림 10. 상세 클래스 다이어그램의 양식 예

표 1. IT_Topic의 종류

IT_Type	IT_Topic	설 명
가용성	layerIndependence	다른 계층에게 부작용을 최소화시킬 수 있는 정도
가용성	mwMessage	미들웨어로 전달되는 메시지가 변조되지 않는 정도
가용성	namingService	올바른 네이밍 서비스를 제공하는 정도
...	...	...
신뢰성	interaction	미들웨어 계층간에 명시된 상호작용을 준수하는 정도
신뢰성	performance	명시된 성능을 유지하는 정도
신뢰성	serviceIndependence	상호의존적이지 않은 서비스를 제공하는 정도
...	...	...
무결성	applMessage	어플리케이션 계층에서의 메시지의 무결성
무결성	method	어플리케이션 계층에서의 메소드의 무결성
무결성	data	어플리케이션 계층에서 사용되는 데이터의 무결성
...	...	...
기밀성	applMessage	서비스를 제공하기 위해 어플리케이션에서 수행하는 메시지 플로우의 기밀성
기밀성	opResult	서비스를 제공하기 위한 어플리케이션 수행 결과의 기밀성
...	...	...

신뢰성 serviceIndependence 상호의존적이지 않은 서비스를 제공하는 정도.

이는 추출되지 않을 경우도 있기 때문에 선택적으로 표기하는 것이다.

IT_Mechanism은 {IT_Type[IT_Topic(Specific_Value)]}를 지원하기 위한 침입 감내 메커니즘으로서, 표 2에 제시되어 있다. 적절한 침입 감내 메커니즘이 없을 경우, 추가적으로 식별하여 표 2를 확장하여 표기할 수 있다. 표 2에서 UC와 Class는 유스케이스와 클래스를 의미하는 것으로서, 각각의 해당 침입 감내 메커니즘이 모델링 시 적용되는 대상을 의미한다.

4.1 UML-IT를 적용한 유스케이스 명세

본 논문에서 제공하는 표기법에 따라 메커니즘을 기술하여 유스케이스에 침입 감내 요구 사항을 반영하도록 한다. 기존 유스 케이스 다이어그램은 단지 기능적인 부분만 명세하지만 본 논문에서는 침입 감내와 관련된 요구 사항들을 유스케이스에 반영하여 명세하였다. 그림 11은 유스 케이스 모델링의 산출물인 유스 케이스 다이어그램이다. 그림 11에 보면, 각각의 유스 케이스에서 침입 감내와 관련된 요구사항이 해당 유스 케이스에 적용될 경우에 각 유스 케이스 별로 이전 절에 정의된 침입 감내 요구사항들이 기술되어 있다. 즉, IT_Type, IT_Topic, Specific_Value, 그리고IT_Mechanism이 명시되어 있다. 이렇게 함으로써, 기능적인 요구사항을 표현하는 유스케이스에 비기능적인 요구사항 특히 침입 감내와 관련된 요구사항들을 함께 표현할 수 있기 때문에, 시스템 개발자들로 하여금 각각의 기능에서 필요하는 침입 감내 요구 사항들이 어떠한 것이며, 미들웨어 계층에서 어떠한 침입 감내 기능들이 호출되어야 하는 지를 명확히 이해할 수 있도록 해준다.

4.2 UML-IT를 적용한 객체 모델 명세

기존 UML의 클래스 다이어그램에 침입 감내 부분을 담당하는 클래스 혹은 객체에 대해서는 <<MW>>를 정의하여 표현하는데 이는 미들웨어의 약어로서 침입 감내 서비스를 담당하는 클래스나 객체들이 미들웨어 계층에 존재하기 때문이다. 시스템에서 비 기능적인 요구 사항을 만족해야 하는 속성 혹은 오퍼레이션의 경우 각 이름 앞에 NFR_ 가 추가되어 표기 된다. 속성에 대한 비 기능적인 요구사항을 상세하게 기술하기 위해서 클래스 식별 작업과 동일한 방법인 Tagged values를 사용한다.

그림 12는 UML-IT 를 적용하여 수신 관리 시스템에 대한 개념적 클래스 다이어그램을 보인 것이다.

4.3 UML-IT를 적용한 행위 모델 명세

행위 모델링에 있어서 침입 감내 관련 서비스와 이를 제어하는 제어 객체 그리고 다른 객체들과의

표 2. 침입 감내 메커니즘

IT_Mechanism	Applies To	설 명
NetMon	UC, Class	라우터, 통신 채널 등 네트워크 자원들을 감시한다.
SysResourceMon	UC, Class	CPU, 메모리, I/O 등의 시스템 자원들에 발생하는 증상들을 실시간으로 감시한다.
ServMon	UC, Class	어플리케이션이 제공하는 서비스들의 행위나 시스템 자체에서 발생되는 예측되지 않은 행위들을 실시간으로 감시하고 기록한다.
RespVal	UC, Class	서비스 요청에 대한 응답이 적합한 지에 대해 테스트 등을 통해 확인 한다.
ReqVal	UC, Class	시스템이 가지는 서비스에 대한 요청이 적합한지에 대해 확인한다.
TriggAnal	UC, Class	트리거 정보를 분석하여 침입의 대상, 파급 효과, 패턴 등을 파악하는 기능이다.
NetLogAnal	UC, Class	네트워크 상에서 발생된 로그를 분석한다.
TranLogAnal	UC, Class	트랜잭션 상에서 발생된 로그를 분석한다.
SensCfg	UC, Class	침입에 대해 이벤트를 발생시키거나 경고 기능을 위한 방법이다.
DataIntgrVal	UC, Class	데이터가 가지는 값이 요구되는 범위 내에서 유효한 것인지를 판단한다.
MsgIntgrVal	UC, Class	요청 메시지가 유효한지를 판단한다.
AuthenVal	UC, Class	서비스를 이용하고자 하는 사용자를 인증한다.
AuthorVal	UC, Class	서비스 요청자가 특정 행위에 대해 권한을 가지고 있는지를 검사한다.
Encryption	UC, Class	데이터나 메시지를 암호화 한다.
PlatDiv	UC, Class	다양한 플랫폼을 사용하여, 특정 위협에 대한 동시적인 침해를 막는다.
SecComm	UC, Class	개체간 안전한 통신 채널을 구축하거나 설정한다.
SourceLoc	UC, Class	인증과 유사하게 시스템에 접근하는 자의 소스를 식별한다.
Fragmentation	UC, Class	서비스나 데이터를 분리하도록 한다.
Scatter	UC, Class	서비스나 데이터를 분산해 놓는다.
Redundancy	UC, Class	서비스나 데이터를 중복하도록 한다.
NetDiv	UC, Class	다양한 네트워크 루트를 제공한다.
MultAddr	UC, Class	논리적인 접근 지점을 물리적인 접근 지점과 동적으로 연결한다.
ServiceMig	UC, Class	하나의 서비스 제공자 처리에 실패했을 경우, 처리되던 서비스가 다른 서비스 제공자에 의해서 처리되도록 한다.
Isolation	UC, Class	침해된 요소를 고립시킨다.
Broadcasting	UC, Class	침입에 대해 다른 서비스 모듈들이 피해 받지 않도록, 다른 모듈들에게 통보해 준다.
DFCtrl	UC, Class	정보에 대한 흐름에 있어 이상현상이 발생하지 않도록 감시하고 통제한다.
WFCtrl	UC, Class	서비스 처리를 위한 일련의 연속된 흐름에 있어서 비정상적인 흐름이 발생하는 지를 감시하고 통제한다.
Retrans	UC, Class	요청된 서비스 및 데이터가 확실하게 처리되도록 서비스와 데이터를 재전송한다.
DataBackup	UC, Class	서비스되는 상태나 설정된 데이터의 상태를 저장하여 침해당했을 경우에 저장된 상태를 통해 회복한다.
FileAuthonReCfg	UC, Class	파일 접근을 위한 보안 수준 등을 재설정하여 안전한 파일 사용을 보장한다.
ServAuthonReCfg	UC, Class	서비스를 이용하기 위한 보안 수준을 재설정하여 서비스에 대핸 비 권한자의 접근을 차단한다.
OSReCfg	UC, Class	시스템 운용상의 환경 설정값들을 조정하여, 안전한 운용 환경을 설정한다.
ServRecovery	UC, Class	잘못된 서비스 처리에 대해, 처리된 서비스의 행위를 추적하여 그로 인한 파급 효과를 모두 원상태로 회복하도록 한다.
DataRecorvery	UC, Class	잘못된 데이터 처리를 회복하기 위해, 데이터 변경에 대한 기록을 추적하여 올바른 상태로 회복한다.
OSRestart	UC, Class	서비스를 제공하는 운용 환경을 초기화하여 시스템을 회복시킨다.

IT_MechanismApplies To설명.

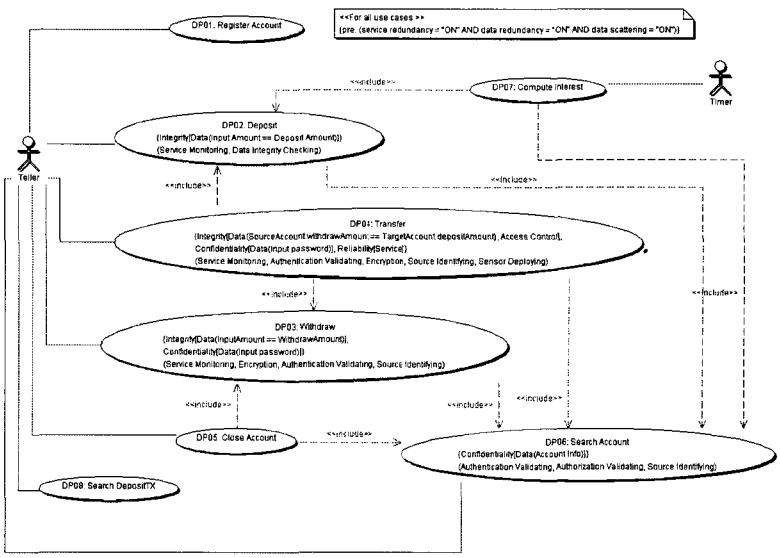

그림 11. 수신 관리 유스 케이스 다이어그램

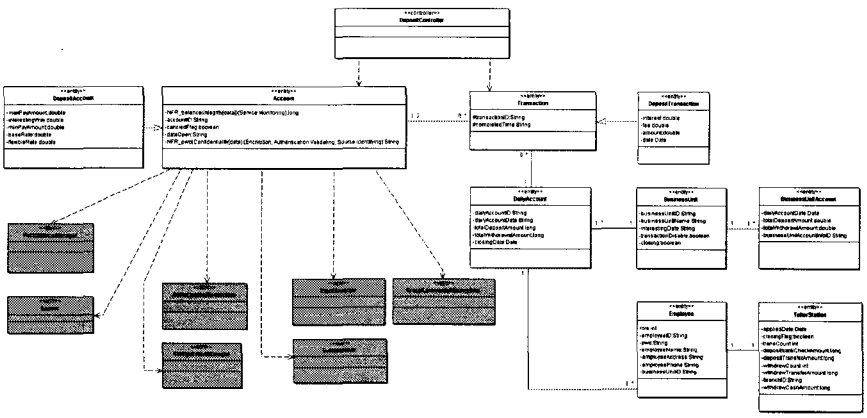

그림 12. 수신 업무의 개념적 클래스 다이어그램

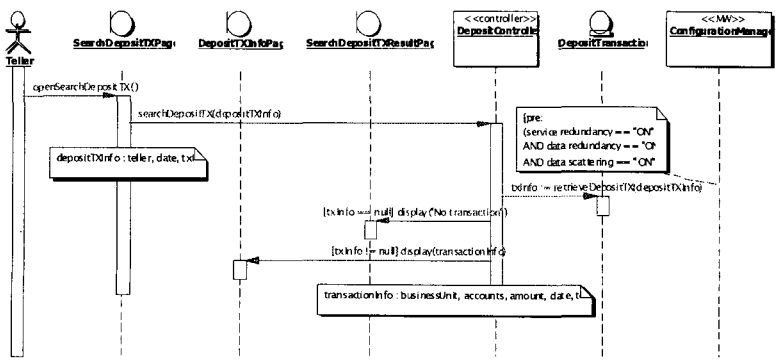

그림 13. 수신관리 거래 내역 조회 상호작용도

표 3. 방법론의 비교 평가

	Advisor	Catalysis	Fusion	RUP	제안한 방법론
침입 감내 요구사항 반영	-	-	-	-	○
프로세스 지원	○	○	○	○	○
UML 기반 모델링	○	○	○	○	○
침입 감내 관련Profile 정의	-	-	-	-	○
구체적인 작업 절차 제시				○	○
미들웨어 특성 고려	-	-	-	-	○
침입 감내 설계 기법 제시	-	-	-	-	○

메시지 전송에 있어서 UML-IT 메커니즘을 적용한 것이다.

행위 모델링을 위해 상호 작용도를 이용하였다. 이 다이어그램에도 침입 감내 서비스들을 담당하는 객체들을 <<MW>>로 표현하였고, 이들 객체들에 대한 메시지는 거의 침입 감내 관련 서비스 호출로 명세된다.

그림 13은 수신관리 업무에서 수신거래 내역 조회 유스케이스에 대한 행위 모델링을 한 예제이다.

5. 평 가

이 장에서는 본 논문에서 제시한 개발 기법과 이를 적용하여 개발한 시스템의 사례 연구 자료들을 통하여 얻어진 결과들을 바탕으로 기존 연구들과의 차이점과 본 연구에서 제시한 기법의 특징을 제시한다.

표 3에서 제시한 방법론들의 비교 평가를 보면 현존 방법론들은 방법론으로서, 프로세스나 UML은 적용하고 있으나, 침입 감내와 같은 요구사항들은 방법론에 제대로 반영하고 있지 않고 있다. 따라서 침입 감내 기반의 응용 소프트웨어를 개발하기 위해서는 현존 방법론들을 그대로 적용하기가 어렵다. 그래서 본 논문에서는 이러한 문제점을 인식하여 침입 감내와 관련된 요구사항들을 반영한 방법론을 제시하였기 때문에 침입 감내와 관련된 요구사항을 설계 기법에 반영하였을 뿐만 아니라, 기존 UML에 확장하여 UML-IT 메커니즘 또한 새롭게 개발하여 정의하고 있다.

특히 침입 감내 기능들은 주로 미들웨어 서비스로 제공된다. 따라서 본 논문에서 제시한 방법론에서는 이러한 침입 감내 서비스들은 미들웨어 객체로 분리하여 표현하고 있으며, 침입 감내의 비기능적 요구사항인 가용성이나 무결성 등에 대한 속성들로 함께 표현하여 제시하고 있다.

그림 14는 본 논문에서 제시한 개발 기법을 통하여 소프트웨어를 개발할 경우에 있어서 소프트웨어 개발의 생산성, 유지보수성, 재사용성, 비용 절감 등의 효과를 측정한 표를 보여주고 있다. 본 논문에서 제시한 기법은 소규모일 경우에는 기존의 기법과 크게 생산성이나 유지보수성 측면에서 차이가 나지 않지만, 시스템의 규모가 크고 복잡도가 클수록 개발의 생산성과 유지보수성, 그리고 재사용성과 비용 절감의 효과가 크게 향상된다.

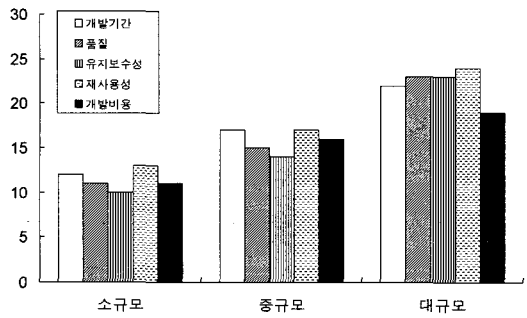

그림 14. 개발 측면에서의 기대효과

6. 결 론

지금까지 침입 감내 소프트웨어 개발에 있어서 UML을 기반으로 한 프로세스를 제시하고, 또한 기존 UML에서 제시하고 있지 못하는 침입 감내 관련 속성들을 확장하여 반영한 UML 프로파일을 제시하였다. 또한 본 연구에서 제시한 침입 감내 기반의 응용 소프트웨어 개발 기법을 실제 적용한 사례 연구를 제시하였다.

현재까지 이루어진 대부분의 연구들은 비즈니스

소프트웨어에 초점을 맞춘 개발 기법들이지만, 이에 반해서 본 연구에서는 보안과 관련된 침입 감내 요소들을 반영한 소프트웨어 개발 기법을 제시하고 있다. 특히 인터넷의 발달과 웹 어플리케이션의 확대로 인하여 보안에 대한 관심이 높아지고 있으며, 예측치 못한 침입에 대해 어떻게 대응할 것인가에 대한 여러 가지 요소 기술들과 이러한 기술들에 부합하는 여러 가지 비기능적인 속성들이 소프트웨어 개발에서 요구되어지고 있다. 이러한 관점에 볼 때, 본 논문에서 제시한 설계 기법은 기존의 설계 기법에 침입 감내와 관련된 여러 속성들을 반영하여 제시함으로써, 보다 사용자의 요구사항과 침입 감내 시스템의 요구사항을 만족하는 고품질의 소프트웨어 개발을 유도할 수 있다고 기대한다.

향후 연구과제로는, 본 연구에서 제시한 프로파일을 반영하여 실제 침입 감내 소프트웨어 설계를 자동화 할 수 있는 개발 도구의 설계 및 개발 기법에 대한 연구와, 이러한 설계 기법을 보다 체계적으로 검증할 수 있는 정형화 기법을 연구하는 것이다.

참 고 문 헌

[1] Deswarte, Y., et. al, "Intrusion Tolerance in Distributed Systems", In IEEE Symp. on Research in Security and Privacy, Oakland, CA, USA, pp.110-121, 1991.

[2] Dutertre, B., Crettaz, V., and Stavridou, V., "Intrusion-tolerant Enclaves", Proceedings of the IEEE International Symposium on Security and Privacy, 2002.

[3] Courtney, T., et. al, "Providing Intrusion Tolerance with ITUA", Proceedings of the ICDSN 2002 Supplementary p.C-5-1, June 2002.

[4] Cukier, M., et. al, "Intrusion Tolerance Approaches in ITUA", In Supplement of the 201 International Conference on Dependable Systems and Networks, pp. 64-65, July 2001.

[5] Just, J.E, and Reynolds, J.C., "HACQIT (Hierarchical Adaptive Control of QoS for Intrusion Tolerance)", In 17thAnnual Computer Security Applications Conference, 2001.

[6] Wang, F. and Killian, C., "Design and Implementation of SITAR Architecture: A Status Report", Proceedings of the ICDSN 2002 Supplementary p.C-3-1, June 2002.

[7] Wang F., et. al, "SITAR: A Scalable Intrusion Tolerance Architecture for Distributed Server", IEEE 2nd SMC Infromation Assurance Workshop, West Point, New York, 2001.

[8] Neves, N. F. and Verissimo, P., "The Middleware Architecture of MAFTIA: A blueprint", DI/FCUL TR 99-6, Department of Computer Science, University of Lisboa, Sept. 2000.

[9] Powell, D., et. al, "MAFTIA(Malicious- and Accidental-Fault Tolerance for Internet Applications)", Sup. Of the 2001 International Conference on Dependable Systems and Networks (DSN2001), pp.D-32-D-25, July 2001.

[10] D'Souza and Wills, *Objects, Components, and Frameworks with UML,* Addison Wesley, 1999.

[11] Hewlett Packard Company, "Engineering Process Summary: Fusion 2.0:", Hewlett Packard Company, 1998.

[12] Compuware Corp., "Uniface Development Methodology: Uniface 7.2", Compuware Corporation, 1998.

[13] ETRI, "마르미III 방법론", 한국전자통신연구원, at URL:http://www.component.or.kr/, 2001.

[14] Grady Booch, Jim Rumbaugh, and Ivar Jacobson, *UML User Guide*, Addison-Wesley, 1999.

조 은 숙

1993년 동의대학교 전산통계학과 이학사
1996년 숭실대학교 대학원 컴퓨터학과 공학석사
2000년 숭실대학교 대학원 컴퓨터학과 공학박사
1999년~2000년 ㈜객체정보기술 과장
2000년 9월~ 2004년 2월 동덕여대 정보학부 강의전임교수
2002년 1월~2003년 10월 한국전자통신 연구원 초빙연구원
2004년 3월~현재 동덕여대 정보학부 데이터정보학과 전임강사
관심분야 : 객체지향 모델링, 컴포넌트기반 개발 방법론, 소프트웨어 아키텍쳐, 제품계열 공학

김 철 진

1996년 경기대학교 전자계산학과 (학사)
1998년 숭실대학교 대학원 컴퓨터학과(공학석사)
2004년 숭실대학교 대학원 컴퓨터학과(공학박사)
2004년 3월~2004년 6월 서울카톨릭대학교 초빙교수
2004년 11월~현재 삼성전자 책임연구원

이 숙 희

1979년 숙명여자대학교 독문학과 졸업(학사)
1982년 동국대학교 경영대학원 정보처리학과 졸업(석사)
1991년 성균관대학교 대학원 통계학과 졸업(박사)
1987년~1993년 동신대학교 전자계산학과 교수
1993년~현재 서경대학교 인터넷정보학과 교수
관심분야 : 객체지향 소프트웨어 개발 방법론, 소프트웨어 테스트, 컴포넌트기반 소프트웨어공학