역할기반 접근통제 시스템에서 응용 프로그램의 설계 및 실행지원 프레임워크

이 형 효[†] · 최 은 복[††] · 노 봉 남[†††]

요 약

역할기반 접근통제 정책은 정보시스템에 의해 관리되는 정보의 단순한 보호기능 뿐만 아니라, 기업환경의 관리체계를 자연스럽게 모델링하는 장점과 함께 다수의 사용자와 정보객체들로 구성된 환경에서 효과적인 권한관리 기능을 제공하는 특징을 가지고 있어 일반 기업환경에서 채택, 활용되어 가고 있다. 역할기반 접근통제 모델이 제공하는 주요 기능은 모델 구성요소들에 대한 정보관리와 정보에 대한 사용자 접근요청의 허가여부 판단기능이며, 지금까지 진행된 대부분의 연구가 모델 자체의 정립, 모델 구성요소 및 임무분리 등과 같은 주요 보안특성들에 대한 정형적인 기술 위주로 진행되었다. 그러나, 현재의 역할기반 접근통제 모델은 응용 프로그램의 정의나 설계, 실행 기능을 지원하고 있지 않고 응용 프로그램을 중심으로 운용되는 자동화된 정보시스템 환경에서 적용하는데 문제점을 가지고 있으며, 이를 보완하기 위한 연구가 요구되고 있다. 본 논문에서는 역할기반 접근통제 모델의 구성요소 중 사용자에 의해 기동되는 세션을 응용 프로그램의 설계와 실행 단위로서 활용하기 위한 능동적 특성에 대하여 고찰한다. 또한 세션을 기반으로 한 임무분리 특성과 응용 프로그램의 설계 및 실행지원을 위한 전반적인 프레임워크를 제시한다.

Application Design and Execution Framework in Role-Based Access Control Systems

HyungHyo Lee[†] · EunBok Choi[††] · BongNam Noh[†††]

Abstract

Role-Based Access Control(RBAC) security policy is being widely accepted not only as an access control policy for information security but as both a natural modeling tool for management structure of organizations and flexible permission management framework in various commercial environments. Important functions provided by the current RBAC model are to administrate the information on the components of RBAC model and determine whether user's access request to information is granted or not, and most researches on RBAC are for defining the model itself, describing it in formal method and other important properties such as separation of duty. As the current RBAC model which does not define the definition, design and operation for applications is not suitable for automated information systems that consist of various applications, it is needed that how applications should be designed and then executed based on RBAC security model. In this paper, we describe dynamic properties of session which is taken for a passive entity only activated by users, as a vehicle for building and executing applications in an automated information systems. And, a framework for session-oriented separation of duty property, application design and operation is also presented.

† 정 회 원 : 전남대학교 대학원 전산학과
†† 준 회 원 : 전남대학교 대학원 전산학과
††† 종신회원 : 전남대학교 전산학과 교수
논문접수 : 1999년 4월 1일, 심사완료 : 1999년 10월 2일

1. 서 론

정보시스템에 저장, 관리되는 정보의 안전성 보장 기술중인 하나인 접근통제는 크게 자율적 접근통제(DAC : Discretionary Access Control)와 강제적 접근통제(MAC : Mandatory Access Control)의 두 가지 정책으로 대별된다. 강제적 접근통제 정책은 각 정보에 결합된 비밀 등급(classification level)과 사용자에 부여된 인가 등급(clearance level)을 미리 규정된 규칙에 따라 비교하여 그 규칙을 만족하는 사용자만에게 접근권한을 부여하는 보안정책으로서, 군사 환경과 같은 정보의 비밀성(confidentiality) 위주의 매우 통제된 환경에서만 주로 사용되고 있다[4, 5, 17]. 이에 반해 자율적 접근통제 정책은 정보 소유자의 자율적 판단에 의해 정보에 대한 접근권한이 결정되는 보안 정책으로 강제적 접근통제 정책에 비해 매우 유연한 권한부여 기능을 제공한다. 그러나, 기업 환경은 다른 기업들과는 다른 보안 정책과 요구사항들을 가지고 있어서, TCSEC (Trusted Computer System Evaluation Criteria)에서 규정된 자율적 접근 제어나 강제적 접근 제어 정책만으로 다양한 기업의 보안 요구를 만족시킬 수 없는 문제점을 가지고 있으며, 자율적 접근통제 정책 역시 기업 환경에 적용하기에 적합하지 않다고 평가되고 있다[3, 15].

역할기반 접근통제(RBAC : Role-Based Access Control) 정책은 정보에 대한 사용자의 권한부여 여부를 각 사용자의 식별자나 이미 정의된 규칙에 의해 판단하지 않고, 사용자가 소속된 조직내에서의 역할에 의해 결정하는 특징을 가지고 있다. 즉, 역할기반 접근통제 시스템에서는 정보에 대한 접근 권한이 사용자에게 직접 부여되지 않고, 조직에서 규정된 역할들에게 배정된다. 이러한 역할기반 접근통제 정책은 정보에 대한 통제가 제한된 수의 관리자에 의해 수행하고, 각 사용자마다 접근 권한을 배정하여 관리하는 대신 조직내에서의 역할에 따라 접근 권한을 부여하는 방법을 채택하여 사용자가 대단히 많은 실제 기업 환경에 효과적으로 적용할 수 있는 특징을 가진다[6, 9, 11].

한편, 역할기반 접근통제 모델이 제공하는 주요 기능은 모델 구성요소들에 대한 정보관리와 정보에 대한 사용자 접근요청의 허가여부 판단기능이며, 지금까지 진행된 대부분의 연구가 모델 자체의 정립, 모델 구성요소 및 임무분리 등과 같은 주요 보안특성들에 대한 정형적인 기술 위주로 진행되었다. 그리고 현재의 역할기반 접근통제 모델은 사용자에 의해 요청된 정보에 대한 접근요구 처리기능 위주로 설계되었으며, 응용 프로그램의 정의나 설계, 실행에 대한 기능은 지원하고 있지 않다. 따라서, 현재의 역할기반 접근통제 모델이 응용 프로그램을 중심으로 운용되는 자동화된 정보시스템 환경에서 활용되기에는 제약사항이 있으며, 이에 대한 보완연구가 요구되고 있다[20]. 본 논문에서는 사용자에 의한 접근요청의 단순한 판단기능을 수행하는 현재의 모델이 워크플로우 형태의 응용 프로그램들로 구성된 자동화된 정보시스템 환경에 활용되기 위하여 현재 모델이 가진 문제점을 살펴보고, 지금까지 연구가 진행되지 않았던 응용 프로그램 설계 및 지원을 위한 전반적인 프레임워크를 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 역할기반 접근통제 모델의 특징과 각 구성요소별 정의, 모델이 지원하는 임무분리 특성, 그리고 현재 모델의 문제점에 대해 살펴본다. 3장에서는 본 연구에서 응용 프로그램의 설계 및 실행단위로 제시한 세션의 주요 동적 특성과 응용 프로그램의 무결성 지원을 위한 세션 임무분리 특성, 세션을 이용한 응용 프로그램 기술문법, 실행모듈에 대하여 기술한다. 마지막으로 4장에서는 본 연구의 결론과 향후 연구과제를 제시한다.

2. 역할기반 접근통제 정책

강제적 접근통제 정책은 군사 환경이나 매우 제한적인 응용분야에서 매우 제한된 수의 시스템 보안관리자(system security administrator)들에 의해 정의된 일정한 규칙에 따라 정보에 대한 사용자의 접근 여부를 결정한다. 이에 반하여, 자율적 접근통제는 각 정보의 소유자들의 자율적인 판단에 따라 그 정보에 대한 접근권한을 다른 사용자에게 위임하거나 취소할 수 있는 권한을 가지고 있어, 강제적 접근통제 정책보다 정보에 대한 훨씬 유연하고 분산된 접근통제 기능을 수행할 수 있는 특징을 가지고 있다. 그러나 이 두 정책 모두 실제 기업환경에 적용되기에는 부적합한 특성들이 있다[3].

역할기반 접근통제 정책은 기업 환경뿐만 아니라 데이터베이스, 운영체제 등에 적용될 수 있는 매우 유연한 접근통제 정책으로[7], 자율적 또는 강제적 접근통제 정책보다 정보에 대한 고수준의 접근통제와 효율적

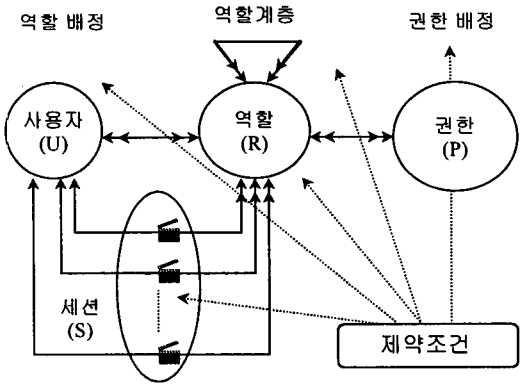
인 접근권한 관리를 수행할 수 있는 장점을 가지고 있다[1, 3, 12, 13, 14]. 또한 역할기반 접근통제 모델의 구성요소를 변경함으로써 이미 정의된 정책만을 지원하는 대신, 주어진 기업 환경의 특성과 필요에 따라 유연한 접근권한 정책을 구현할 수 있어 보안정책에 독립적(policy-neutral)인 특성을 제공한다[7, 8, 10, 16].

2.1 역할기반 접근통제 모델

역할기반 접근통제 모델에서 정보에 대한 연산을 수행할 수 있는 권한(permission)들은 사용자에게 직접 할당되지 않고, 주어진 기업 환경에서 정의된 역할에 대해서만 배정되는 특징을 가지고 있다. 따라서 사용자가 원하는 정보에 대한 연산을 수행하기 위해서는 먼저 해당 정보에 대한 연산을 실행할 수 있는 권한을 가진 역할의 소속원(member)이 되어야 하며, 사용자를 특정 역할의 소속원으로 배정하는 권한은 미리 정해진 시스템 보안관리자들에 의해 수행된다. 따라서 전체 시스템의 보안 관리를 위한 제어가 몇 명의 보안관리자에 의해 이루어지게 되므로, 자율적 접근통제 정책에서 발생할 수 있는 접근권한 통제의 어려움을 해결할 수 있다.

이처럼 역할기반 접근통제 모델에서는 권한의 관리를 사용자와 정보 객체간의 관계로 인식하는 대신 기업 환경에서의 역할과 정보 객체간의 관계로 설정, 관리함으로써 사용자와 정보 객체의 수가 대단히 많은 실제의 기업 환경에 매우 적합한 특성을 제공한다[3, 12]. 또한 최소특권 원칙(least privilege principle), 임무분리(SOD : separation of duty), 정보객체에 대한 고수준의 처리기능 제공 등과 같은 주요 보안 원칙들 역시 지원하고 있다[3]. (그림 1)은 역할기반 접근통제 모델의 구성요소를 나타내고 있다.

역할(role)은 주어진 기업 환경에서 정의된 업무의 기능으로서, 각 역할이 수행 가능한 권한들로 구성된다. 역할에 배정된 권한은 조직의 규정이나 규칙에 의해 정의되며, 역할에 소속된 사용자들에게 동일하게 제공된다. **역할 계층**(role hierarchy)은 역할에 배정된 권한들사이에 포함관계가 있는 역할들간의 부분순서(partial order) 관계로서 기업의 권한과 책임 체계와 매우 유사하여 기업의 권한체계를 모델링하는데 매우 유용하게 활용된다. **사용자**(user)는 컴퓨터 시스템을 통하여 시스템내의 정보를 사용하는 객체로서 한 사용자는 한 명의 사람에 대응된다. **권한**(permission)은 정보객체에 대해 실행 가능한 연산의 집합으로 구성된다.



(그림 1) 역할기반 접근통제 모델의 구성요소

세션은 한 사용자와 여러 개의 역할들로 구성된 집합으로 구성되며, 사용자는 세션을 통하여 자신에게 배정된 역할들 중의 일부 또는 전체를 수행할 수 있다. 특히, 세션과 결합된 역할들을 **활성 역할**(active role)이라 하며, 활성 역할은 세션의 사용자에 배정된 역할들의 집합에 포함되는 특성을 만족해야 한다. **사용자 배정**(user assignment)과 **권한 배정**(permission assignment)은 다대다 관계이며 역할기반 접근통제 모델에서 매우 중요한 구성요소이다. 역할기반 접근통제 모델의 가장 특징중의 하나는 사용자가 정보 객체들에 대해서 실행할 수 있는 연산들을 직접 사용자에게 부여하는 대신 조직의 업무 수행에 필요한 역할과 각 역할에 대한 권한을 배정하고(권한배정), 사용자는 해당 역할에 구성원이 됨으로써(사용자 배정) 정보객체에 대한 원하는 연산을 수행하도록 하는 것이다. 이러한 방법은 사용자와 정보 객체의 수가 많은 일반 기업 환경에서 권한의 관리를 매우 용이하게 수행할 수 있는 장점을 제공한다.

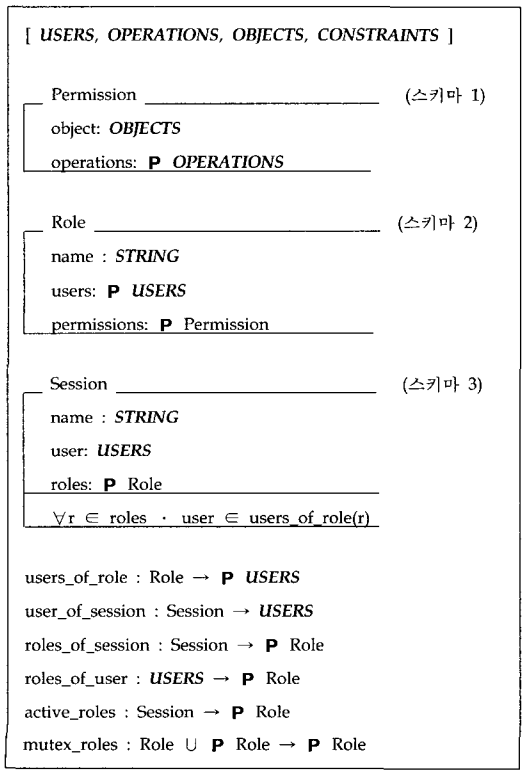
제약조건(constraints)은 위에서 정의된 모든 구성요소들에 대하여 적용될 수 있으며, 각 구성요소가 가지는 특성을 제한사항이나 조건 등을 기술한다. 제약조건의 예로서는 임무분리, 한 역할에 할당될 수 있는 최대 사용자 수(cardinality), 선수 역할(prerequisite roles), 시간제약사항(temporal constraint) 등이 있다.

2.2 정형적 기술

정보 시스템에 대한 정형적인 기술은 시스템 사양이나 기능에 대한 잘못된 이해를 방지하는 기능 외에도

시스템의 정적인 구조 및 동적인 동작 특성을 명확히 기술하고 분석하는데 활용된다. 본 논문에서는 역할기반 접근통제 모델과 구성요소, 그리고 다양한 보안특성을 기술하는데 Z 형식언어를 이용한다. Z 형식언어는 집합론과 predicate calculus를 기초로 시스템의 전반적인 정적, 동적 특성을 스키마를 이용하여 정형적으로 기술하는 기능을 제공한다[18, 19]. (그림 2)는 역할기반 접근통제 모델과 관련된 특성들을 Z 형식언어를 이용하여 기술하는데 필요한 기본 자료형과 함수들에 대한 정의들을 표현하고 있다. 기본자료형 ***USERS, OPERATIONS, OBJECTS, CONSTRAINTS***는 각각 사용자의 집합, 연산의 집합, 정보를 저장하고 있는 객체 또는 시스템 자원의 집합, 제약조건들의 집합을 나타내고 있다.

스키마 Permission, Role, Session은 기본자료형을 이용하여 정의된 역할기반 접근통제 모델의 권한, 역할, 세션 구성요소를 나타내고 있다. 예를 들어, 스키마 Permission은 ***OBJECTS*** 자료형을 갖는 object 변수와 ***OPERATIONS*** 기본 자료형의 멱집합을 자료형으로 갖는 operations 변수로 구성된다는 의미이다. 그리고 스키마 Session은 name, user, roles 변수로 구성되어 있다는 정보외에, user에 의해 지정된 사용자가 roles에 포함된 모든 역할에 배정이 되어 있어야 한다는 제약사항을 포함하고 있다.



(그림 2) 기본 자료형과 함수

한편, Z 형식언어에서 함수는 **'함수이름 : 입력 자료형 → 출력 자료형'**의 형태로 기술되며, 자료형 앞의 기호 '**P**'는 멱집합(powerset)을 의미한다. 함수 users_of_role은 입력으로 주어진 역할에 배정된 사용자 집합을, user_of_session은 세션에 배정된 사용자를 반환하는 기능을 제공한다. 함수 roles_of_session은 세션에 배정된 역할들을, roles_of_user는 입력으로 주어진 사용자에게 배정된 역할의 집합을, active_roles는 세션에 의해 활성화된 역할들을 계산한다. 함수 mutex_roles은 입력으로 주어지는 역할 또는 역할의 집합에 대하여 상호 배타적인 역할의 집합을 반환기능을 수행한다.

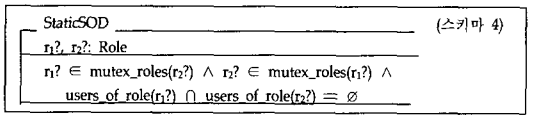
2.3 임무분리 성질

정보의 무결성 성질은 정보의 비밀성 못지 않게 일반 기업환경에서 매우 중요한 보안 속성로서 정보 시스템의 무결성 유지를 위한 정책들이 필요하게 된다[15]. 역할기반 접근통제 정책을 기반으로 한 정보시스템들은 정보의 무결성(integrity)에 영향을 미치는 권한들이 역할 단위로 관리되므로, 정보의 무결성에 손상을 입힐 수 있는 권한을 가진 역할들, 즉 상호배타적 역할(mutually exclusive roles)들이 수행할 수 있는 권한들간의 충돌(conflict) 여부에 따라 이 역할들이 동일 사용자에게 할당하지 않음으로써 무결성 침해의 가능성을 최소화한다. 그러나, 상호배타적 역할의 지정은 응용 프로그램의 문맥에 따라 다르므로 각 시스템의 시스템 보안관리자에 의해 지정되며, 상호배태적 역할 지정에 적용되는 일반적인 규칙은 존재하지 않는다[2]. 대표적인 임무분리 성질들로는 상호배타적 역할들이 동일 사용자에게 배정되는 점검시점에 따른 정적, 동적 임무분리 성질이 있으며, 동일 응용 프로그램내에서 정보의 무결성에 직접적으로 영향을 미치는 모든 연산들이 한 사용자에 의해 수행가능한 지 점검하는 연산 임무분리 성질이 있다[1].

2.3.1 정적 임무분리

정적 임무분리(static SOD)는 사용자 배정 단계에서

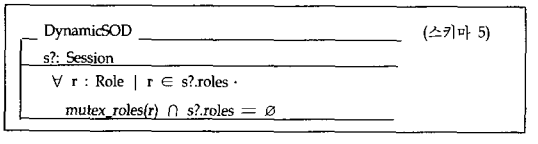
동일 사용자를 상호배타적 역할에 배정하지 않음으로써 정보 시스템의 무결성을 유지하도록 한다. 이 방법은 매우 엄격한 무결성 유지기능을 제공하는 반면 각 역할에 배정되는 많은 수의 사용자가 필요하게 되어 시스템 운용에 따른 부담이 증가하고, 시스템 운용이 유연하지 못한 단점을 가진다. (그림 3)은 정적 임무분리 특성에 대한 Z 스키마이다.



(그림 3) 정적 임무분리 특성

2.3.2 동적 임무분리

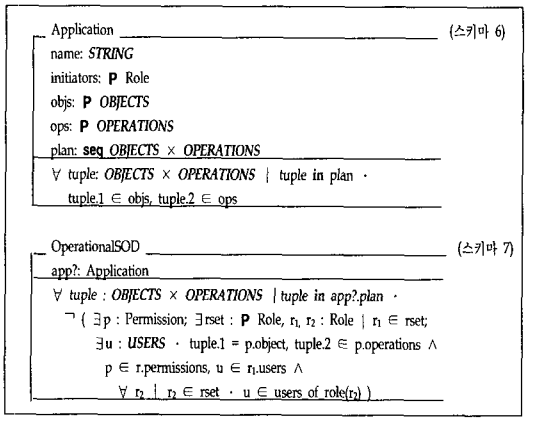
동적 임무분리(dynamic SOD)는 사용자가 비록 사용자 배정 단계에서 상호배타적 역할에 배정되더라도, 실제 동일한 세션에 의해 활성화되는 역할들이 상호배타적인 관계에 있지 않도록 유지하는 방법이다. (그림 4)는 동적 임무분리 특성에 대한 Z 스키마이다. 이 방식은 정적 임무분리 방법에 비해 시스템의 구성이 쉬어지고 운용의 유연성이 증대되는 장점을 가지고 있으나, 응용 프로그램이 실행단계에서 각 사용자가 세션을 통해 활성화하는 역할들간의 관계를 확인할 수 있는 기능이 추가로 요구된다.



(그림 4) 동적 임무분리 특성

2.3.3 연산 임무분리

정적, 동적 임무분리 외의 주요 임무분리 속성이 연산 임무분리(operational SOD)이다. 이 성질은 정보 시스템 환경에서 수행되는 임의의 응용 프로그램에 대하여, 그 응용 프로그램에서 사용되는 모든 연산들이 한 사용자에 의해 수행되어서는 않되는 성질을 기술하고 있다. 연산 임무분리 성질의 기술에 필요한 응용 프로그램과 연산 임무분리 성질을 정형적으로 기술하면 (그림 5)와 같다. (그림 5)의 (스키마 6)에 정의된 응용 프로그램 스키마에서는 응용 프로그램을 정보저장 객체와 그 객체에 수행되는 연산들의 집합으로 정의하고 있으며, (스키마 7)은 (스키마 6)을 기반으로 한 연산 임무분리 성질을 표현하고 있다.



(그림 5) 응용 프로그램 스키마와 연산 임무분리 특성

지금까지 살펴본 역할기반 접근통제 모델은 모델 구성요소들에 대한 정보관리 기능과 정보에 대한 사용자 접근요청의 허가여부 판단기능의 제공을 위하여 설계되었으며, 응용 프로그램의 정의나 설계, 실행을 위한 기능을 포함하고 있지 않는 문제점을 가지고 있다. 따라서, 현재의 역할기반 접근통제 모델을 응용 프로그램을 중심으로 운용되는 자동화된 정보시스템 환경에 적용하는데 문제점이 있으며, 이를 보완하기 위한 연구가 요구된다[20]. 3장에서는 이러한 문제점을 해결하기 위하여 역할기반 접근통제 모델의 구성 요소중 세션을 중심으로 한 응용 프로그램의 임무분리 특성, 그리고 응용 프로그램 기술문법 및 실행환경에 대하여 기술한다.

3. 세션 기반의 응용 프로그램 설계 및 실행 구조

응용 프로그램은 정보 시스템의 무결성을 유지시키면서 주어진 업무를 수행하는 단위로 정의될 수 있다. 지금까지의 진행된 연구 중, 역할기반 접근통제 모델에서 응용 프로그램에 대한 명확한 정의는 없으나, 정보를 저장하고 있는 객체들과 그 객체들에 대한 연산들의 집합으로 응용 프로그램을 정의하고 있는 경우도 있다[2]. 그러나 이와 같은 응용 프로그램에 대한 정의

는 정보시스템에서의 권한관리 기준을 연산 대신 역할로 간주하는 역할기반 접근통제 정책의 기본원칙에 부합되지 않는 점과 정보가 저장된 객체들에 대한 연산을 실행하는 주체 역시 세션임을 고려할 때 부적절하다고 판단된다[20]. 따라서, 본 논문에서는 역할기반 접근통제 정책의 기본원리에 부합되고, 응용 프로그램의 설계 및 운용에 적합하도록 세션을 기본으로 하는 응용 프로그램을 정의한다. 역할기반 접근통제 모델의 구성요소 중 세션은 2.1에 기술된 바와 같이 사용자에게 정보에 대한 실제적인 접근기능을 수행하는 동적인 기능을 수행하고 있지만, 지금까지의 연구들은 세션을 사용자가 터미널 등을 통해 인증과정을 거친 후 실행되고 사용자에 의해 종료되는 수동적인 개념으로 기술하고 있다. 따라서, 역할기반 접근통제 모델이 워크플로우 관리시스템이나 자동화된 정보시스템 등과 같은 동적이며 자동화된 환경에 적합한 보안모델로 활용되기 위하여 능동적인 주체로서 세션의 특성과 세션을 기반으로 한 응용 프로그램 설계 및 실행환경에 대한 연구가 필수적이다.

따라서 본 논문에서는 응용 프로그램을 연관된 여러 개의 세션들과 세션들간의 제약조건들의 집합으로 정의한다. 또한 응용 프로그램은 세션들에게 주어진 제약조건을 참조하여 각 세션의 시작과 종료, 세션에 배정되는 사용자 결정 등의 기능을 수행하는 제어기능 등을 포함한다. 응용 프로그램의 구성요소인 제약조건으로는 상호배타적 관계를 가진 세션들, 수행순서, 세션의 수행시간 또는 세션들간의 시간제약 사항 등이 포함될 수 있다. 이러한 응용 프로그램의 설계과정은 정보시스템의 구성과 응용 프로그램의 기능적 특성을 잘 이해하고 있는 사용자인 응용 프로그램 설계자에 의해 수행된다.

3.1 세션의 정의 및 특성

세션은 (그림 1)에서와 같이 사용자가 세션에 결합된 역할들을 통하여 객체에 연산을 실행하는 기본단위로 정의된다((그림 2)의 (스키마 3)). 그러나, 본 논문의 세션은 사용자에 의해 수동적으로 수행, 종료되는 개념이 아니라, 세션들간의 제약사항을 참조로 응용 프로그램의 제어에 따라 수행되는 응용 프로그램 기능의 설계 및 실행 단위로 정의된다. 그 이유는 세션이 하나의 사용자와 그 사용자가 실행할 수 있는 하나 이상의 역할들로 구성되어, 정보시스템의 무결성 보장을 위하여 하나 이상의 사용자에 의해 수행되어야 하는 응용 프로그램의 설계 및 실행 요구사항을 만족시키기 때문이다. 이러한 세션의 정의는 워크플로우 시스템과 같은 자동화된 정보시스템이나 분산환경에서 응용 프로그램의 실행환경 구축 및 운용의 기본구조로서 활용될 수 있다. 한편, 세션은 여러 개의 응용 프로그램에 의해 공통적으로 호출되는 함수 또는 객체로서 구현될 수 있으며 다음과 같은 특성들을 가진다

① 재사용이 가능한 논리적 단위기능
세션은 한 사용자에 의해 수행될 수 있는 역할들의 집합으로, 응용 프로그램의 구성단위이며 여러 응용 프로그램에 의해 공유가 가능하고 재사용될 수 있는 단위기능을 제공한다.

② 최소특권 원칙의 수행단위
세션과 결합된 활성 역할들은 주어진 시스템의 사용자 배정 과정에서 한 사용자가 수행할 수 있도록 배정된 역할(authorized role)들의 부분집합이다. 따라서, 사용자가 특정 응용 프로그램을 구성하는 세션을 수행할 때 해당 기능만을 수행할 수 있는 활성 역할들만 사용할 수 있으므로 최소특권 원칙을 준수하도록 한다.

$$\forall u : \mathit{USERS};\ \forall s : \mathrm{Session} \mid \mathrm{user\_of\_session}(s) = u \cdot \mathrm{roles\_of\_session}(s) \subseteq \mathrm{roles\_of\_user}(u)$$

(그림 6) 세션의 최소특권 원칙

③ 임무분리 원칙의 수행단위
정보시스템의 무결성에 관련된 응용 프로그램은 정보시스템의 무결성 성질을 보장하기 위하여 악의적 사용자에 의해 무결성을 해칠 수 있는 가능성을 최소화하여야 한다. 세션을 기반으로 한 응용 프로그램은 무결성을 해칠 수 있는 역할들을 상호배타적인 별도의 세션으로 분리하고, 각 세션을 서로 다른 사용자가 수행하도록 배정하는 기능을 제공함으로써 정보의 무결성 보장기능을 제공한다.

④ 고수준의 응용 프로그램 정의기능 제공
응용 프로그램에 대한 기존의 정의는 정보저장 객체들과 그 객체에 대한 연산들의 집합으로 규정하였으나[2], 세션을 기반으로 한 응용 프로그램에서는 서로 연관된 정보객체들과 연산들을 역할을 통

하여 세션으로 통합함으로써 응용 프로그램의 설계자와 임무분리 성질 등을 설정하는 시스템 보안관리자가 보다 개념적으로 응용 프로그램을 설계하고 시스템을 운용, 관리할 수 있다.

3.2 세션기반의 응용 프로그램 정의

세션은 3.1에서 기술된 바와 같이 응용 프로그램을 구성하는 기본단위로서 사용자의 접근권한을 실행단계에서 결정하는 기능을 제공한다. 또한 세션은 정보객체와 연산 단위의 대신 역할 단위의 접근통제를 관리하므로 임무분리 성질의 정의와 응용 프로그램들이 정의된 임무분리 속성을 준수하는 지의 판단이 용이한 장점을 가진다. (그림 7)은 (그림 5)의 (스키마 6)에 정의된 응용 프로그램을 세션의 특성을 기반으로 다시 정의한 세션 기반의 응용 프로그램(SoApplication : Session-oriented Application)에 대한 스키마를 나타내고 있다.

SoApplication (스키마 6')
name: *STRING*
initiators: **P** Role
sessions: **P** Session
constraint: *CONSTRAINTS*

(그림 7) 세션 기반의 응용 프로그램 정의

3.3 세션 임무분리 성질

연산 임무분리 성질은 주어진 응용 프로그램에서 사용되는 모든 연산을 수행할 수 있는 사용자에게 응용 프로그램의 수행권한을 부여하지 않음으로써, 시스템의 무결성을 유지하는 데 그 목적이 있다. 그러나 어떤 경우에는 비록 연산 임무분리 특성을 만족하더라도 시스템의 무결성이 침해받는 경우가 있어, 보다 엄격한 임무분리 특성의 정의가 필요하게 된다. 예를 들어 (그림 8)과 같은 3개의 세션으로 구성된 응용 프로그램을 가정한다.

s_1: 물품구매 요청
s_2: 구매 요청한 물품 검수
s_3: 구매된 물품에 대한 지불 허가

(그림 8) 세션으로 구성된 응용 프로그램 예

만일 사용자 u1이 세션 s1, s2를 수행하는 사용자로, 사용자 u2가 세션 s3를 수행하는 사용자로 결정되었다고 가정할 때, 비록 사용자 u1이 응용 프로그램에서 사용되는 모든 권한을 수행할 권한이 없어도 물품구매 과정의 무결성이 침해됨을 알 수 있다. 그 이유는 연산 임무분리 성질이 응용 프로그램에서 사용되는 모든 연산들이 한 사용자에 의해 수행되는 것만을 허용되지 않도록 규정하고 있을 뿐, 정보의 무결성에 영향을 주는 연산들의 대한 의미적 연관성은 고려하지 않기 때문이다.

따라서, 본 논문에서는 연산 임무분리 성질의 문제점을 보완하고, 무결성 보장의 단위를 응용 프로그램을 구성하는 전체 연산의 집합이 아닌 세션간의 의미적 연관성을 기준으로 한 세션 임무분리 성질(SoSOD: Session-oriented SOD)을 정의한다. 일반적으로, 모든 응용 프로그램들에 대하여 일반적으로 적용되는 세션 기반의 임무분리 특성에 대한 정의는 불가능하기 때문에, 시스템의 특성에 따라 시스템 보안관리자와 응용 프로그램 설계자에 의해 특정 응용 프로그램에서 한 사용자에게 동시에 배정되어서는 안되는 상호배타적인 세션이 결정된다. (그림 9)의 ***MUTEX_SESSIONS*** 자료형은 주어진 시스템에 정의된 상호배타적 세션들을 정의하기 위한 자료형으로, 상호배타적 세션은 각 세션이 이용되는 응용 프로그램에서의 의미적 관련성에 의해 결정되므로 응용 프로그램과 함께 정의된다.

MUTEX_SESSIONS == **P** Application × Session × Session

SoSOD (스키마 8)
app?: Application
mutex_sessions: ***MUTEX_SESSIONS***
$\forall s_i, s_j : Session \mid s_i \neq s_j, s_i, s_j \in app?.sessions \cdot$
((app?, s_i, s_j) ∈ mutex_sessions ∨
(app?, s_j, s_i) ∈ mutex_sessions) ⇒
user_of_session(s_i) ≠ user_of_session(s_j))

(그림 9) 상호배타적 세션 자료형과 세션 임무분리 특성

3.4 세션-사용자 배정, 세션-역할 배정

(그림 7)의 (스키마 6')에서 정의된 바와 같이 세션은 응용 프로그램 수행의 구성 단위로서 한 명의 사용자와 그 세션의 수행기간동안 사용자가 수행할 수 있는 역할들로 구성된다. (그림 1)의 역할기반 접근통제

모델의 구성요소 중, 사용자 배정과 권한 배정은 시스템 보안관리자에 의해 시스템 보안관리 과정에서 결정된다. 일반적으로 역할기반 접근통제 모델 구성요소 정보 자체의 무결성 유지를 위하여 시스템 관리작업은 시스템의 수행을 정지시킨 상태에서 시스템 보안관리자에 의해 수행된다. 그리고 이때 각 세션은 시스템 보안관리자에 의해 하나의 사용자와 여러 개의 역할들에게 고정적으로 배정되었다. 그러나, 이러한 배정방식은 세션이 하나의 사용자에게만 배정되어야 하는 모델의 정의에 따라 동일한 세션을 실행할 수 있는 사용자가 하나 이상인 경우 각 사용자에 대해 세션을 중복하여 배정해야 하는 문제점을 가지고 있다. 따라서 본 논문에서는 이러한 문제점을 해결하고 세션을 기반으로 한 응용 프로그램의 설계 및 실행에 적합한 사용자와 세션, 그리고 사용자와 역할의 배정방법을 제시한다.

주어진 세션에 대하여 사용자를 배정하는 **세션-사용자 배정**(SUA : Session-User Assignment)과 역할들을 배정하는 **세션-역할 배정**(SRA : Session-Role Assignment) 과정은 그 배정 시기에 따라 시스템의 유연성과 동작성능에 많은 영향을 미치게 된다. 만일 응용 프로그램의 설계자가 SUA, SRA 과정을 응용 프로그램 설계 과정때 결정한다면, 시스템의 동작 성능은 개선될 수 있으나 많은 수의 사용자가 필요하게 되어 시스템의 유연성을 감소하게 된다. 한편, SUA, SRA 과정을 실행시간에 결정한다면 시스템의 동작성능은 SUA, SRA 과정이 응용 프로그램 설계 과정때 결정되는 방법보다 저하될 수 있으나, 시스템 운용의 유연성은 증가하는 장점을 가진다. 따라서 SUA, SRA 두 가지 배정작업의 실행단계에 따라 시스템의 동작 성능과 유연성이 결정된다.

따라서, 본 논문에서는 SUA, SRA 과정의 실행시점을 SRA의 경우 응용 프로그램 설계과정때 응용 프로그램 설계자에 의해, SUA의 경우 응용 프로그램 실행때 응용 프로그램 실행모듈에 의해 결정하도록 한다. 그 이유는 응용 프로그램을 구성하는 각 세션이 실행할 수 있는 권한들이 응용 프로그램의 기능에 의해 결정되는 반면, 세션을 수행할 수 있는 사용자는 여러 명이고 응용 프로그램 실행환경의 조건에 따라 그들 중 한 명이 결정되는 방식으로 비교적 유동적이기 때문이다.

응용 프로그램 설계자는 응용 프로그램 설계과정에서 각 응용 프로그램의 기능에 맞는 세션들을 선택하여 세션들의 수행구조를 기술하고, 무결성 보장을 위한 세션 임무성질 등을 포함한 제약조건을 기술한다. 응용 프로그램 실행단계에서 응용 프로그램 실행 모듈은 응용 프로그램에 의해 지정된 세션 임무분리 성질과 실행 시점에서의 역할기반 접근통제 모델 구성요소 정보 등을 참조하여 각 세션을 수행할 사용자를 결정(세션-사용자 배정)한다.

세션-사용자, 세션-역할 배정은 세션이 한 사용자에 의해 동작되는 세션의 정의와 세션 임무분리 성질을 기반으로 동적, 연산 임무분리 성질들이 만족되도록 하는 기능을 제공한다. <표 1>은 세션-역할 배정, 세션-사용자 배정 단계에서 각각 수행되는 임무분리 성질들에 대하여 기술하고 있다. 응용 프로그램의 구성하는 세션들간의 의미적 연관성을 기초로 지정된 세션 임무분리 성질은 동적, 연산 임무분리 성질을 수행하는 기능을 제공한다. 한편, 정적 임무분리 성질은 시스템 보안관리자에 의해 수행되는 사용자 배정 단계에서 지정되므로 세션-역할 배정이나 세션-사용자 배정에 의해서는 직접 수행되지 않는다.

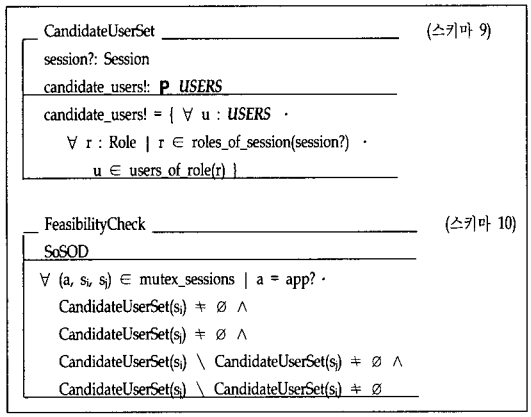
3.5 응용 프로그램 실행가능성 확인기능

응용 프로그램 설계자에 의하여 설계된 응용 프로그램이 세션 임무분리 특성을 만족시키면서 실행될 수 있는지 확인하는 과정은 실행과정에서 발견될 수 있는

〈표 1〉 세션-역할, 세션-사용자 배정단계에서 수행되는 임무분리 성질

세션 배정	임무분리 성질	비 고
세션-역할 배정	동적 임무분리	정보의 무결성과 관련된 역할들을 서로 다른 세션에 배정
	연산 임무분리	응용 프로그램을 하나 이상의 세션으로 분리하여 구성하고 분리된 세션들을 상호배타적 세션으로 지정
세션-사용자 배정	연산 임무분리	세션-역할 배정때 연산 임무분리 성질의 수행을 위해 지정된 상호배타적 세션들은 서로 다른 사용자에게 배정
	세션 임무분리	응용 프로그램에 의해 지정된 상호배타적 세션들은 서로 다른 사용자에게 배정

응용 프로그램의 구조적인 문제점을 미리 발견하여 수정할 수 있는 기능을 제공한다. 즉, 주어진 응용 프로그램의 세션 임무분리 성질을 만족시키는 역할기반 접근통제 모델의 구성요소가 존재하는 지를 확인하는 기능이다. 그러나, 실행가능성 확인 절차를 만족한 응용 프로그램도 실제 실행환경 시점의 구성요소 상태에 따라 실행이 불가능할 수 있다. (그림 10)의 (스키마 9)는 입력으로 주어지는 세션을 수행할 수 있는 사용자들의 집합을 계산하는 기능을 수행하며 수행결과로 주어지는 사용자 집합은 주어진 세션을 수행할 수 있는 역할들에 배정이 되어있음을 의미한다. (그림 10)의 (스키마 10)은 (스키마 9)와 세션 임무분리 성질을 이용한 설계된 응용 프로그램의 실행가능성 확인 스키마를 정의하고 있다.

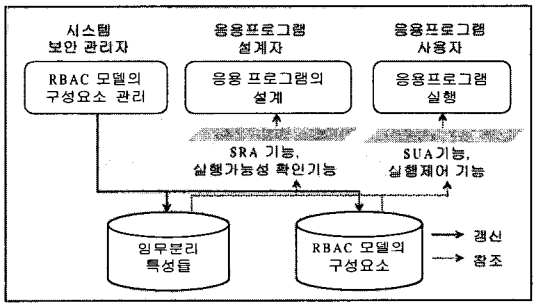


(그림 10) 응용 프로그램의 실행가능성 점검 스키마

3.6 응용 프로그램 설계 및 운용 구조

역할기반 접근통제 모델을 보안모델로 하여 구축된 정보시스템의 운용 구조는 시스템 보안관리자에 의해 수행되는 역할기반 접근통제 모델의 구성요소 관리단계, 응용 프로그램 설계자에 의한 응용 프로그램 설계 및 실행가능성 확인단계, 그리고 일반 사용자들에 의해 응용 프로그램이 실행되는 단계로 구성되며 (그림 11)은 전체적인 운용 구조를 나타내고 있다.

시스템 보안관리자는 보안정책에 의해 규정된 임무분리 특성들(정적, 동적, 연산, 세션)을 기반으로 역할기반 접근통제 모델의 구성요소의 변경기능을 수행하며, 응용 프로그램 설계자는 정보시스템의 무결성 보장을 위해 임부분리 특성들을 만족시키면서 시스템 보안관리자에 의해 관리되는 사용자, 역할, 권한, 사용자 배정, 권한배정 등의 정보를 참조하여 세션을 정의하고, 이들을 기초로 필요한 응용 프로그램을 설계한다. 이때 설계된 응용 프로그램은 사용자에 의해 실행되기 전 현재의 역할기반 접근통제 모델의 구성요소를 이용하여 실행될 수 있는지 확인하는 정적인 점검절차를 거치게 된다. 정보 시스템 운용 단계에서 각 응용 프로그램은 시스템에 정의된 임무분리 특성, 실행 시점에서의 역할기반 접근통제 모델의 구성요소 정보들을 참조하여 실행된다. 응용 프로그램 설계자는 시스템 보안관리자가 실행할 수 있는 임무분리 특성 정의기능을 응용 프로그램 설계단계에서 사용할 수 있는 특수한 시스템 보안관리자로 볼 수 있다.



(그림 11) 응용 프로그램 설계, 운용 및 관리 구조

3.6.1 응용 프로그램 기술언어

세션을 단위로 구성되는 응용 프로그램 구조의 명확한 기술은 응용 프로그램 구성의 체계적인 표현뿐 아니라 응용 프로그램 실행에 필수적인 정보로 활용된다. (그림 12)는 기본적인 응용 프로그램 기술 구조체와 BNF(Backus-Naur Form) 형식으로 표현된 응용 프로그램 기술문법을 각각 나타내고 있다. 세션을 기반으로 한 응용 프로그램 구조에 대한 문법적인 기술은 응용 프로그램에서 사용되는 정보객체와 연산, 역할, 임무분리 성질 등에 대한 정보제공 기능 외에도 이에 대한 자동처리를 통하여 응용 프로그램 실행 과정에서 세션-사용자 배정 알고리즘의 입력으로 사용된다.

현재는 순차, 병렬, 조건부, 반복 수행 및 종료 등의 기본적인 구조체들을 고려하였으며, 매우 복잡하고 응용 프로그램의 기술을 위하여 추가적인 기능을 가진 구조체들의 추가도 가능하다. 응용 프로그램 기술구조체 중 시간제약사항은 한 세션의 수행시작 가능시각,

```
application → 'APPLICATION' application_name
              'SESSION' application_sessions
              [ 'INITIATOR' role_lists ]
              [ 'SOD' sod_lists ]
              application_spec_lists
application_sessions ::= application_sessions ',' session_spec | session_spec
session_spec ::= '(' session ',' role_lists ')'
role_lists ::= role_lists ',' role | role
sod_list ::= sod_list ',' sod_element | sod_element
sod_element ::= '(' session ',' session ')' | temporal_spec
temporal_spec ::= '(' session ',' session ')' 'for' time
                | session 'between' '[' time ',' time ']'
application_spec_lists ::= application_spec_lists application_spec | application_spec
application_spec ::= sequential_spec | parallel_spec | conditional_spec
                   | loop_spec | abort_spec | '(' application_spec_lists ')'
sequential_spec ::= application_spec ';' application_spec
parallel_spec ::= application_spec '||' application_spec
conditional_spec ::= 'if' condition 'then' application_spec [ 'else' application_spec ]
loop_spec ::= 'while' '(' condition ')' 'do' application_spec
abort_spec ::= 'abort'
session ::= <session_identifier>
role ::= <role_identifier>
time ::= [0-9]+
condition ::= temporal_spec | logical_expr
logical_expr ::= relation { 'and' relation } | relation { 'or' relation }
relation ::= simple_expr [ rel_op simple_expr ]
simple_expr ::= [ unary_op ] term { add_op term }
term ::= factor { mul_op factor }
factor ::= <object_identifier> | '(' logical_expr ')'
rel_op ::= '=' | '<>' | '<' | '<=' | '>' | '>='
add_op ::= '+' | '-'
mul_op ::= '*' | '/'
unary_op ::= '+' | '-' | 'not'
```

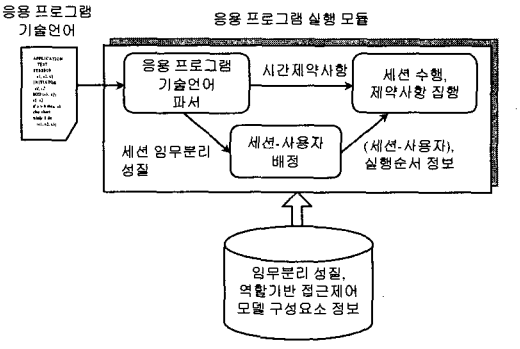
(그림 12) 세션 기반의 응용 프로그램 기술 문법

두 세션간의 수행간격 등을 지정한다. 응용 프로그램의 설계자는 응용 프로그램 수행도중 특별한 조건에서 응용 프로그램을 종료시킬 때, **abort**를 이용할 수 있다. (그림 12)에 기술된 문법의 <role_identifier>, <session_identifier>, <object_identifier>는 각각 역할과 세션, 정보객체를 유일하게 표시하기 위한 식별자를 의미하며, 조건부 수행과 반복 수행의 결정에는 여러 가지 조건이 사용될 수 있으나, 현재는 정보객체들간의 산술, 논리 관계와 시간 제약사항만을 포함하고 있다.

3.6.2 응용 프로그램 실행 모듈

응용 프로그램 실행 모듈은 응용 프로그램 설계자에 의해 기술된 연관된 세션과 임무분리 성질 등을 고려하여 응용 프로그램을 수행하며 그 구성은 (그림 13)과 같다.

응용 프로그램 기술언어 파서는 응용 프로그램 기술언어를 입력으로 구문분석 과정을 수행한 후, 기술언어 중 세션 임무분리 성질과 관련된 정보는 세션-사용자 배정 모듈에게, 시간제약사항은 세션수행 모듈로



(그림 13) 응용 프로그램 실행모듈의 구성

각각 전달한다. **세션-사용자 배정 서브모듈**은 파서로부터 주어진 정보를 참조하여 각 세션에 배정될 사용자를 결정하고, 수행순서 관련 정보를 세션수행 모듈로 전송한다. **세션수행 서브모듈**은 응용 프로그램의 각 세션에 배정된 사용자 정보, 세션의 수행순서, 세션 수행의 시간제약사항 등을 참조하여 각 세션을 동작 또는 종료시키며, 각 단계별 제약사항이 준수되도록 관리하는 역할을 수행한다. 응용 프로그램 실행 모듈의 서브모듈들은 각 동작단계마다 시스템 보안관리자에 의해 지정된 역할기반 접근통제 구성요소 정보를 참조하여 수행한다. (그림 14)는 세션-사용자 배정 알고리즘을 나타내고 있다. 알고리즘의 DoS(Degree of Separation) 함수 값은 임의의 응용 프로그램 내부에서 입력으로 주어진 세션과 상호배타적 관계에 있는 세션의 개수를 나타낸다. 상호배타적 관계에 있는 두 세션 중 DoS 함수 값이 큰 세션에 대한 사용자 배정작업을 먼저 수행하는 이유는 더 적은 수의 세션과 상호배타적 관계를 가진 세션의 사용자 배정작업이 먼저 이루어진 경우, 더 많은 수의 상호배타적 세션을 갖는 세션의 사용자 배정과정에서 먼저 수행한 사용자-세션 배정작업을 다시 수행해야 하는 경우가 발생할 수 있기 때문이다.

3.7 세션기반의 응용 프로그램 설계 및 실행과정

이 절에서는 지금까지 기술된 프레임워크를 기반으로 응용 프로그램 설계 및 실행과정을 예제를 통해 살펴본다. 먼저 (그림 15)와 같은 역할기반 접근통제 모델의 구성정보를 가진 가상의 기업환경을 설정하고, 응용 프로그램을 물품구매 응용 프로그램으로 한다. 편의상 (그림 15)에 나타낸 사용자, 역할, 권한, 역할계

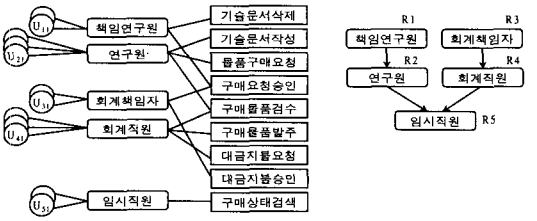
```
ALGORITHM Session-User Assignment
INPUT
  /* 응용 프로그램, 상호배타적 세션들의 집합 */
  app, mutex_sessions
  s1, s2, sk ∈ app.sessions
  /* 역할기반 접근통제 구성 정보 중 사용자 정보 */
  U = {u1, u2, ..., up}, p ≥ 1
BEGIN
  for each session si where (app, si, sj) ∈ mutex_sessions or (app, sj, si) ∈ mutex_sessions
  do
    if no user is assigned to session si then
      if DoS(si) ≥ DoS(sj) then
        for all mutually exclusive sessions (app, si, sk) where (app, si, sk) ∈ mutex_sessions or
            (app, sk, si) ∈ mutex_sessions
        do
          if candidate_user_set(si) ≠ Ø and candidate_user_set(sk) ≠ Ø and
            candidate_user_set(si) − candidate_user_set(sk) ≠ Ø and
            candidate_user_set(sk) − candidate_user_set(si) ≠ Ø then
              assign any user u ∈ candidate_user_set(si) − candidate_user_set(sk) to si
              if no user is assigned to sk then
                assign any user u ∈ candidate_user_set(sk) − candidate_user_set(si) to sk
              fi
          else
            continue /* 세션 si와 sk에 배정 가능한 다른 사용자 선택작업 수행 */
          fi
        od
      fi
    fi
  od
  /* 응용 프로그램에 포함된 모든 상호배타적 세션들에 대하여 사용자가 배정되었는 지 확인 */
  for each session si where (app, si, sj) ∈ mutex_sessions or (app, sj, si) ∈ mutex_sessions
  do
    if si is not assigned to user u ∈ U then
      exit /* 세션사용자 배정작업 실패 */
    fi
  od
  /* 응용 프로그램에 포함된 상호배타적 세션들 이외의 모든 세션들에 대하여 사용자 배정작업 수행 */
  for each session si where (app, si, sj) ∉ mutex_sessions and (app, sj, si) ∉ mutex_sessions
  do
    assign user u ∈ candidate_user_set(si) to si
  od
  /* 응용 프로그램에 포함된 상호배타적 세션들 이외의 모든 세션들에 대하여 사용자 가 배정되었는지 확
인 */
  for each session si where (app, si, sj) ∉ mutex_sessions and (app, sj, si) ∉ mutex_sessions
  do
    if si is not assigned to user u ∈ U then
      exit /* 세션사용자 배정작업 실패 */
    fi
  od
END
```

(그림 14) 세션-사용자 배정 알고리즘

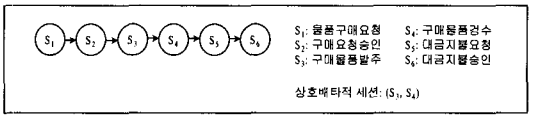
층 등의 구성정보는 물품구매 응용 프로그램과 관련된 주요 권한과 역할, 역할계층 등을 위주로 구성하였다.

3.7.1 응용 프로그램의 분석

현재의 역할기반 접근통제 모델에서는 응용 프로그램의 정의, 설계 및 실행에 대한 지원기능을 제공하지 않는데 비해, 3.1에서 기술된 바와 같이 본 논문에서 제시된 프레임워크에서는 세션을 응용 프로그램의 설계단위로 정의하고 있다. 그 이유는 세션이 하나의 사용자와 그 사용자가 실행할 수 있는 하나 이상의 역할들로 구성되어, 정보시스템의 무결성 보장을 위하여 하나 이상의 사용자에 의해 수행되어야 하는 응용 프로그램의 설계 및 실행 요구사항을 만족시키기 때문이다. 따라서, 본 논문에서 제시된 새로운 프레임워크에서는 응용 프로그램의 작성을 위하여 응용 프로그램



(그림 15) 가상기업의 역할기반 접근통제 모델 구성정보



(그림 16) 세션 단위로 분석된 응용 프로그램 구조

개발자는 먼저 응용 프로그램을 역할단위로 분석하는 작업을 수행한다. 역할단위의 분석작업이 끝나면 응용 프로그램을 구성하는 역할들 중 의미적으로 상호배타적 역할을 결정한 후 이들을 먼저 세션으로 지정한다. 그리고 응용 프로그램을 구성하는 나머지 모든 역할들에 대하여 연속되어 실행되는 두 역할이 한 사용자에 의해 실행이 가능하면 이 들을 묶어 하나의 세션으로 구성하는 작업을 반복한다. (그림 16)은 물품구매 응용 프로그램이 세션 단위로 분석된 결과를 나타내고 있다. 그리고, 응용 프로그램이 운영되는 정보시스템의 특성에 따라 상호배타적 세션의 집합이 함께 분석되어야 한다.

예를 들어, 응용 프로그램 설계자가 연구부서에서 사용될 물품구매 응용 프로그램을 작성하는 경우, 세션 S1에는 '연구원' 역할을, 세션 S2에는 '책임연구원' 역할을 배정해야 하며, 회계부서에서 사용될 물품구매 응용 프로그램을 작성하는 경우는 세션 S1, S2에 각각 '회계직원', '회계책임자' 역할을 배정해야 한다.

3.7.2 응용 프로그램 기술 및 실행가능성 확인

(그림 16)과 같이 분석된 응용 프로그램은 (그림 13)의 응용 프로그램 실행모듈에 의해 처리되기 위하여 응용 프로그램 기술문법에 따라 명세되어야 한다. (그림 17)은 (그림 12)의 응용 프로그램 기술문법에 따라 (그림 16)과 같이 분석된 물품구매 응용 프로그램을 연구부서에서 사용할 수 있는 형태로 기술한 내용이다. 이 응용 프로그램은 연구원 역할에 배정된 사용자

```
APPLICATION 물품구매
SESSION (S1, 연구원), (S2, 책임연구원), (S3, 회계직원)
        (S4, 회계직원), (S5, 회계직원), (S6, 회계책임자)
INITIATOR 연구원
SOD (S3, S4)
S1; S2; S3; S4; S5; S6
```

(그림 17) 물품구매 응용 프로그램의 명세

만이 실행시킬 수 있음을 알 수 있다.

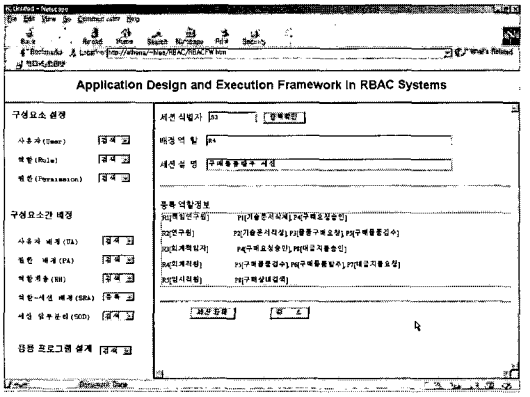
(그림 17)에 의해 명세된 응용 프로그램의 실행가능성 확인은 (그림 10)의 스키마에 따라 점검할 수 있다. 즉, 상호배타적인 세션 S3와 S4를 수행할 수 있는 사용자의 집합(CandidateUserSet)을 각각 계산하여 각 사용자 집합이 공집합이거나 상호배타적인 세션의 CandidateUserSet이 서로 다른 하나 이상의 사용자를 포함하고 있지 않으면 실제 운영환경에서 응용 프로그램의 실행이 불가능하다는 것을 알 수 있다. 예제의 경우는 '회계직원'의 역할에 배정된 사용자가 하나 이상이어야 물품구매 응용 프로그램이 실행될 수 있음을 알 수 있다.

3.7.3 응용 프로그램 실행

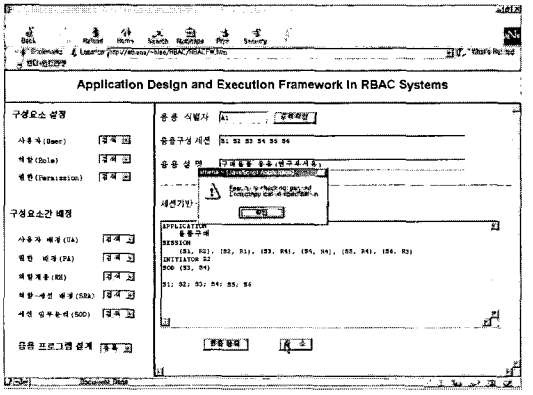
세션을 중심으로 분석, 설계되어 기술문법에 의해 명세된 응용 프로그램은 (그림 13)에 나타난 응용 프로그램 실행모듈에 의해 수행된다. 먼저 응용 프로그램 실행을 요청한 사용자에 대해 해당 사용자가 'INITIATOR'에 의해 지정된 역할을 실행할 수 있는 사용자인지 확인하는 과정을 거친다. 그리고 응용 프로그램 명세에 나타난 상호배타적 세션 정보와 (그림 14)에 기술된 세션-사용자 배정 알고리즘을 통해 각 세션을 수행할 사용자를 결정한다. 세션수행 서브모듈을 세션의 수행순서, 세션에 배정된 사용자 정보, 기타 시간제약사항 등에 따라 세션을 기동, 종료시킴으로써 응용 프로그램의 실행하게 된다.

3.8 프로토타입 구현

응용 프로그램 설계 및 실행지원 프레임워크 중 역할기반 접근통제 모델 구성요소 관리기능과 응용 프로그램 설계 명세처리 기능, 세션-사용자 배정 알고리즘에 대한 프로토타입이 구현된 상태이다. 시스템 보안관리자의 역할기반 접근통제 모델 구성요소에 대한 검색, 등록, 변경, 삭제 권한과 응용 프로그램 설계자의 세션, 세션 임무분리, 응용 프로그램 설계명세에 대한



(a) 세션 등록 화면



(b) 응용 프로그램 등록 화면

(그림 18) 응용 프로그램 설계 지원

검색, 수정, 등록, 삭제 권한을 웹 인터페이스를 이용하여 수행할 수 있도록 구현되었다. (그림 18)은 응용 프로그램 설계자에 의해 실행되는 세션 등록((그림 18)의 (a))과 응용 프로그램 등록((그림 18)의 (b)) 화면을 나타내고 있다.

역할기반 접근통제 모델 구성요소와 세션 정보, 응용 프로그램 정보는 Solaris 2.6 운영체제의 Sun 워크스테이션에서 동작하는 오라클 데이터베이스 버전 7.2.3에 의해 관리되며, 모델의 구성요소들은 각각의 DB 테이블로 구현되었다. 응용 프로그램 명세처리기는 lex와 yacc[21]을 이용하여 작성되었으며, 웹을 통한 시스템 보안관리자와 응용 프로그램 설계자의 관리기능 수행은 C 언어와 오라클 데이터베이스 접근을 위해 Pro*C 라이브러리로 작성된 CGI 프로그램과 자바스크립트를 사용하여 구현하였다.