# Role-Based Access Control (RBAC)

Prof. Ravi Sandhu
Executive Director and Endowed Chair

Lecture 4

ravi.utsa@gmail.com
www.profsandhu.com

# Access Control

**Fixed policy**

**Discretionary Access Control (DAC), 1970**

**Mandatory Access Control (MAC), 1970**

**Role Based Access Control (RBAC), 1995**

**Attribute Based Access Control (ABAC), ????**

**Flexible policy**

# Access Control

Fixed policy

Ownership gives discretion

**Discretionary Access Control (DAC), 1970**

One-directional information flow

**Mandatory Access Control (MAC), 1970**

Policy neutral

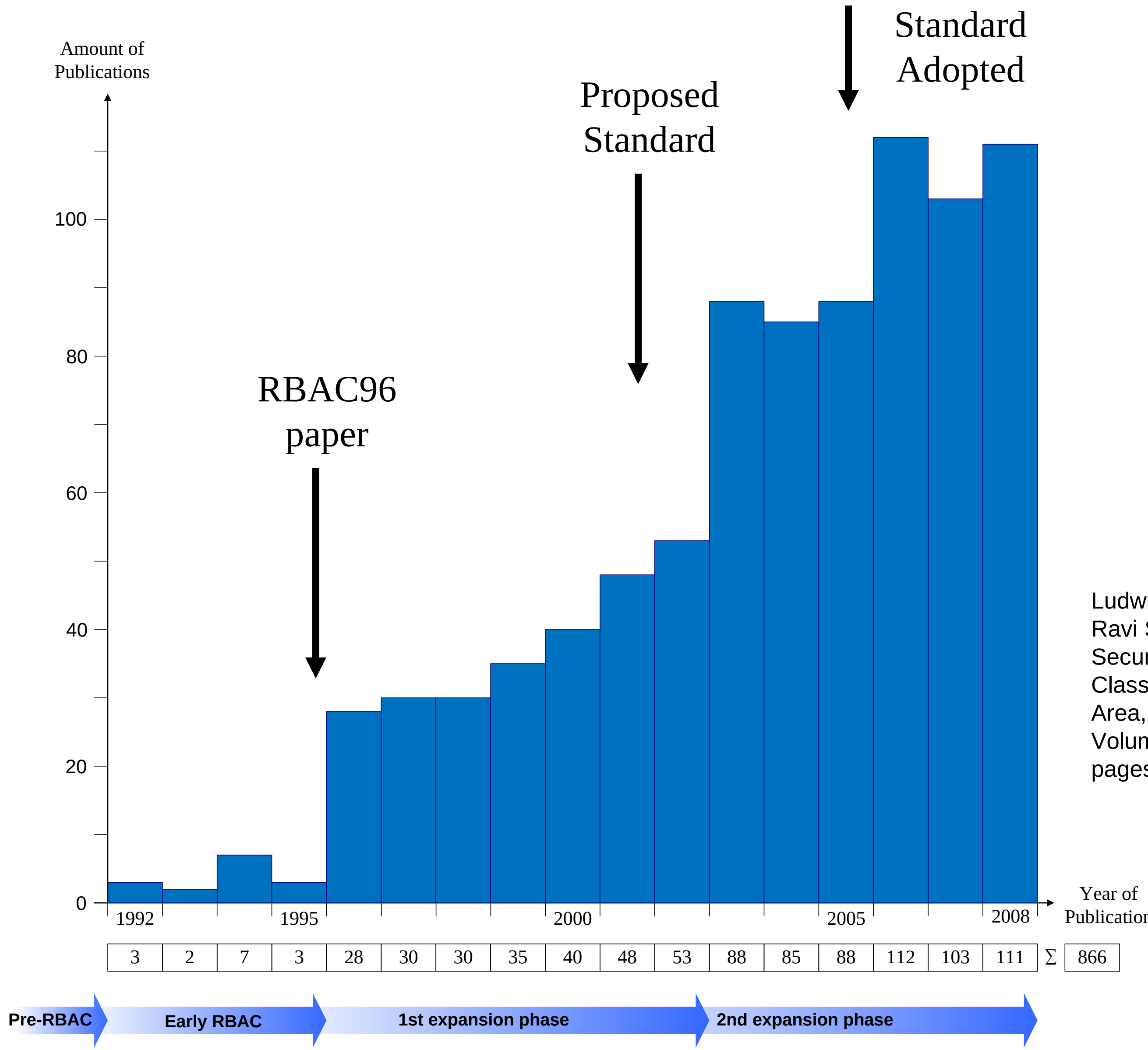
**Role Based Access Control (RBAC), 1995**

**Attribute Based Access Control (ABAC), ????**

Flexible policy

# The RBAC Story






Ludwig Fuchs, Gunther Pernul and Ravi Sandhu, Roles in Information Security-A Survey and Classification of the Research Area, Computers & Security, Volume 30, Number 8, Nov. 2011, pages 748-76

# RBAC: Role-Based Access Control

- Access is determined by roles
- A user's roles are assigned by security administrators
- A role's permissions are assigned by security administrators

First emerged: mid 1970s
First models: mid 1990s

Is RBAC MAC or DAC or neither?

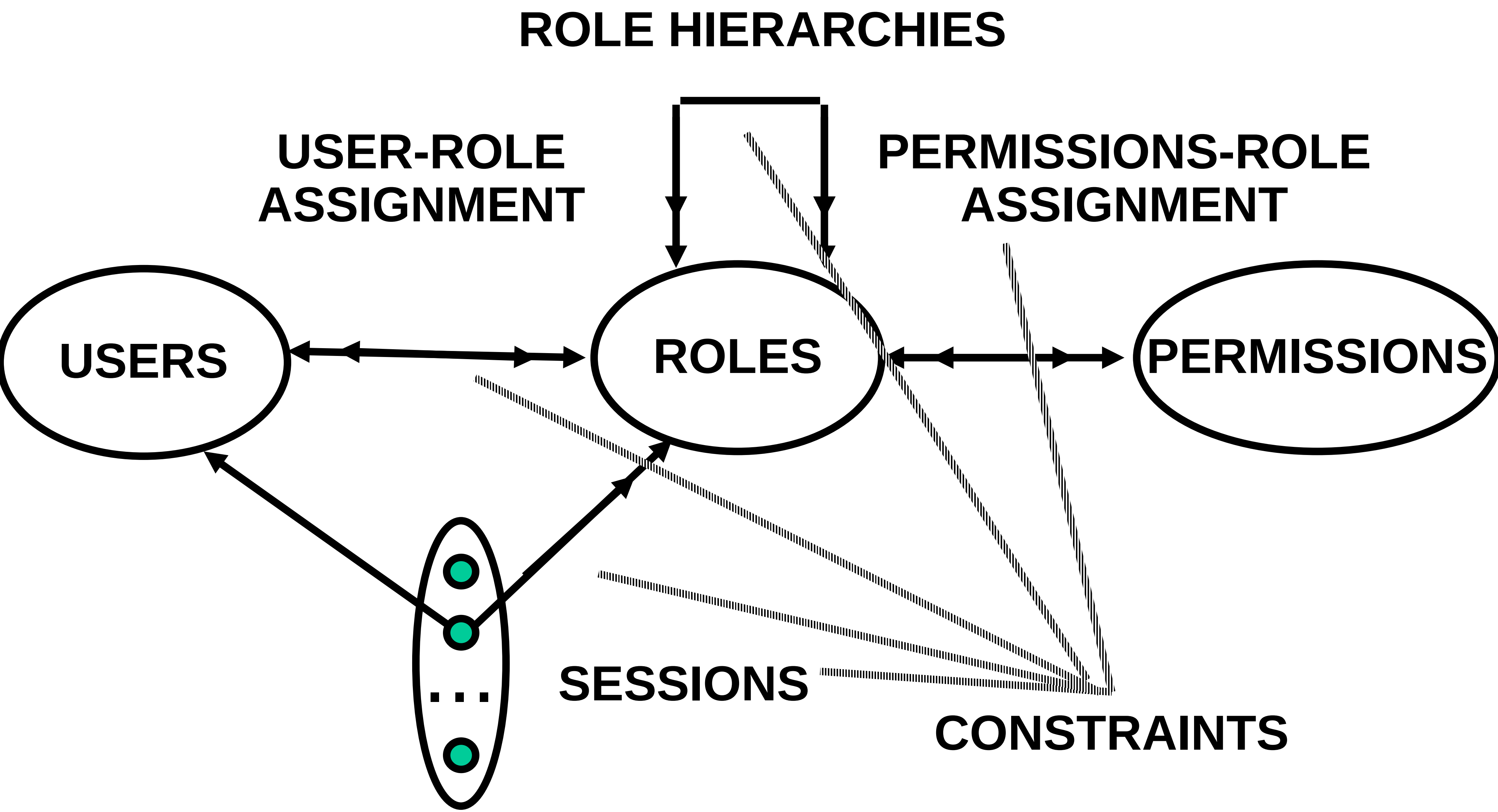
- RBAC can be configured to do MAC
- RBAC can be configured to do DAC
- RBAC is policy neutral

RBAC is neither MAC nor DAC!

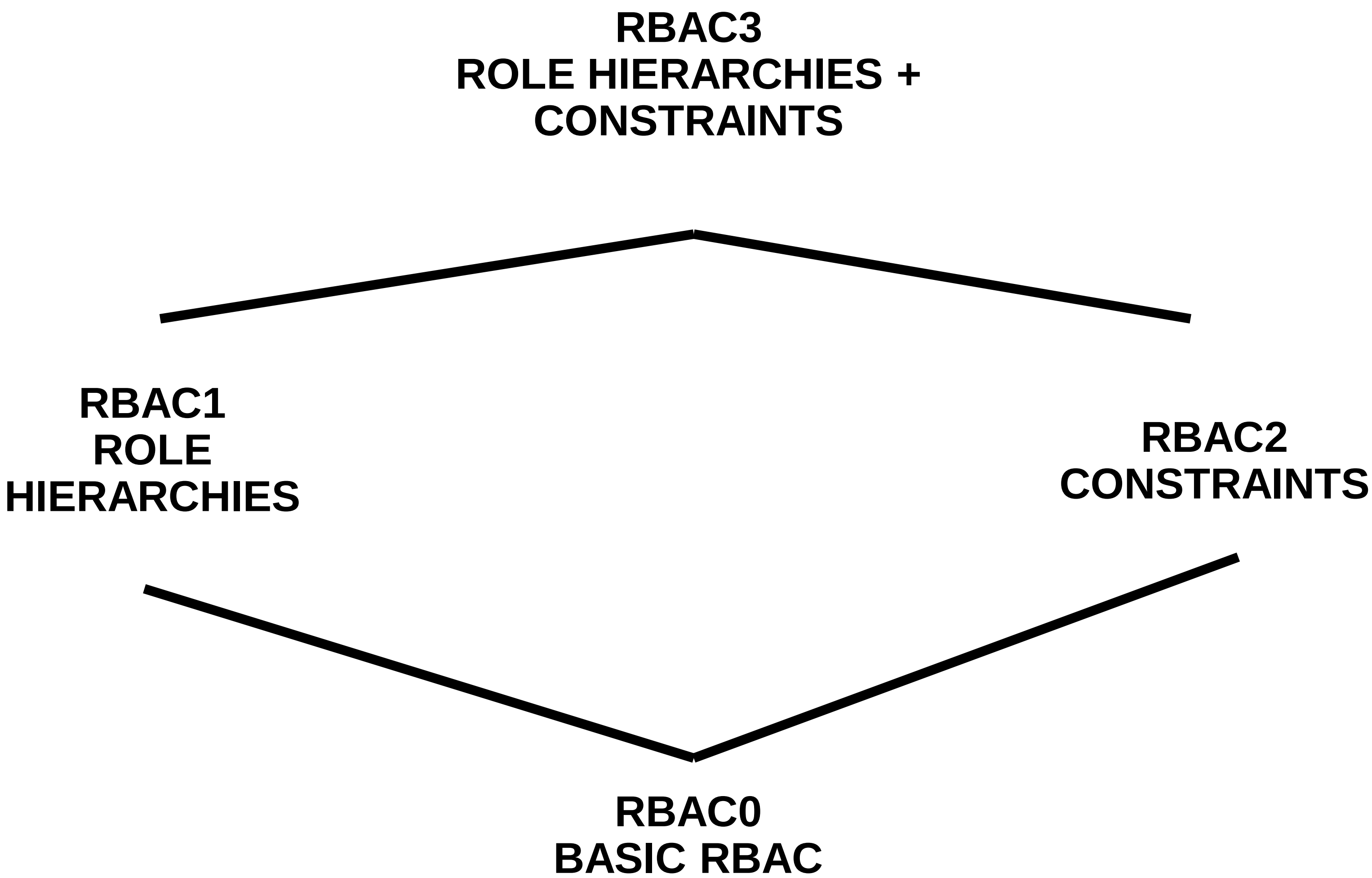
# RBAC96 Model

# RBAC96 Model Family





# RBAC96 Model Family



# Founding Principles of RBAC96

- **Abstraction** of Privileges
  - Credit is different from Debit even though both require read and write
- **Separation** of Administrative Functions
  - Separation of user-role assignment from role-permission assignment
- **Least Privilege**
  - Right-size the roles
  - Don’t activate all roles all the time
  - Limit roles of a user
  - Limit users in a role
- **Separation of Duty**
  - Static separation: purchasing manager versus accounts payable manager
  - Dynamic separation: cash-register clerk versus cash-register manager

# ROLES AS POLICY

- A role brings together
  - a collection of users and
  - a collection of permissions
- These collections will vary over time
  - A role has significance and meaning beyond the particular users and permissions brought together at any moment

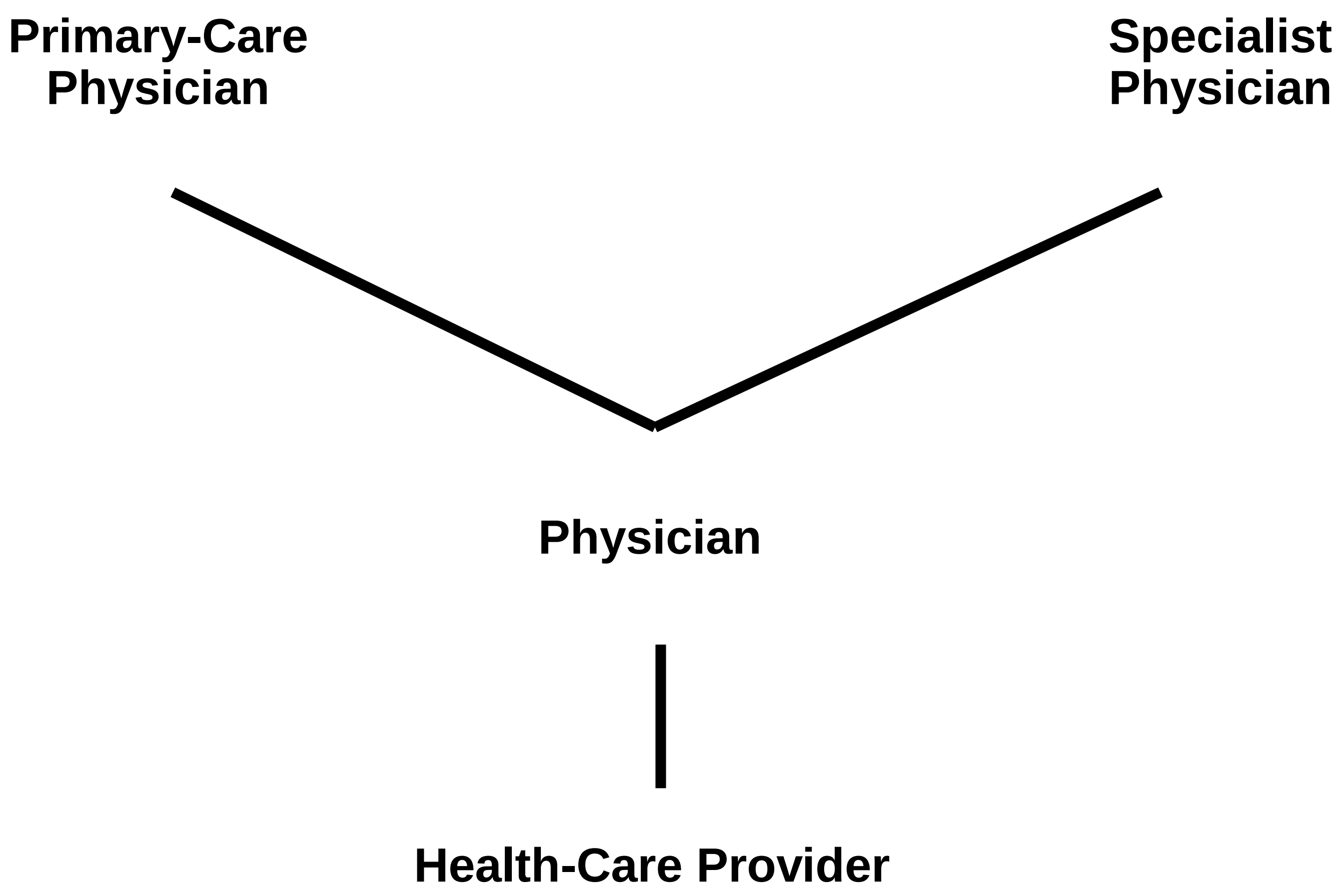
# ROLES VERSUS GROUPS



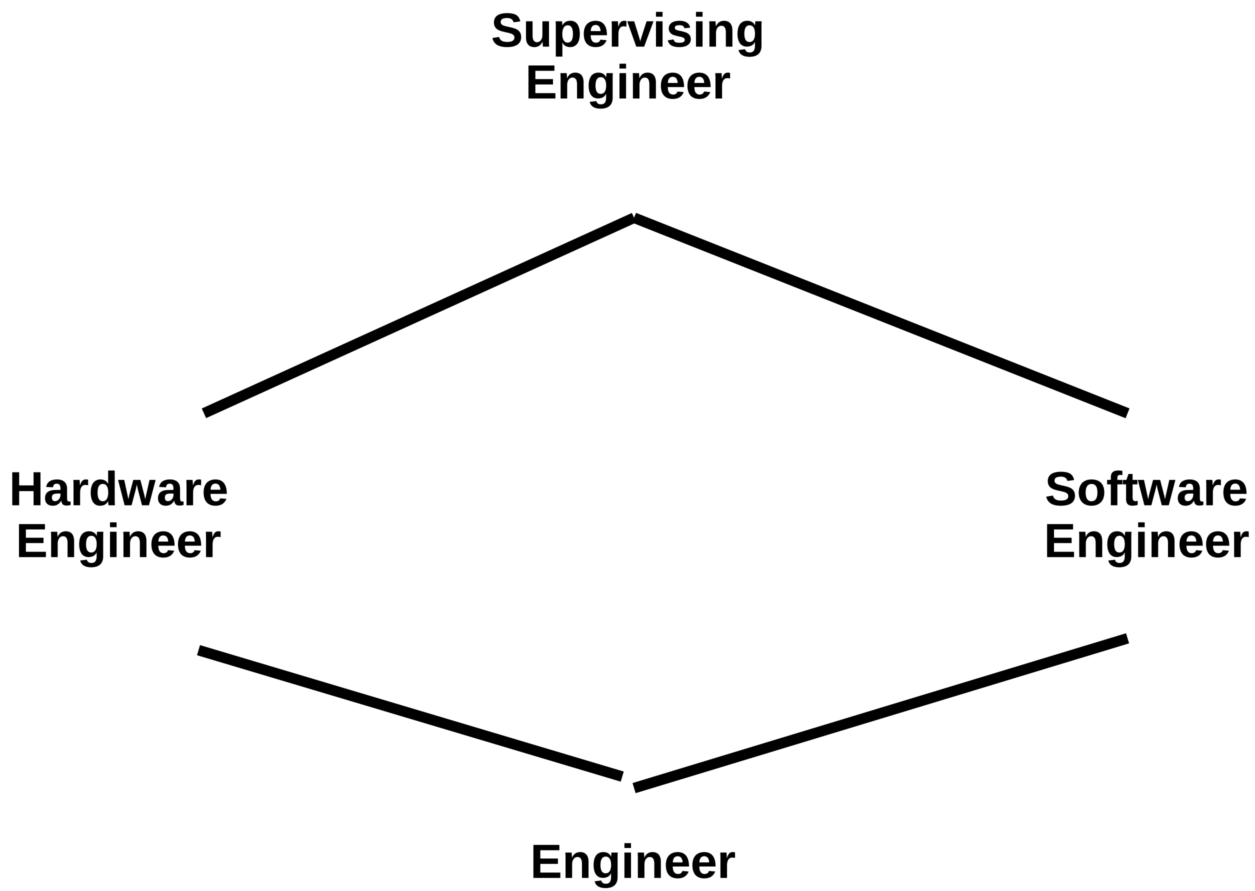
- Groups are often defined as
  - a collection of users
- A role is
  - a collection of users and
  - a collection of permissions
- Some authors define role as
  - a collection of permissions

- Most Operating Systems support groups
  - BUT do not support selective activation of groups
- Selective activation conflicts with negative groups (or roles)

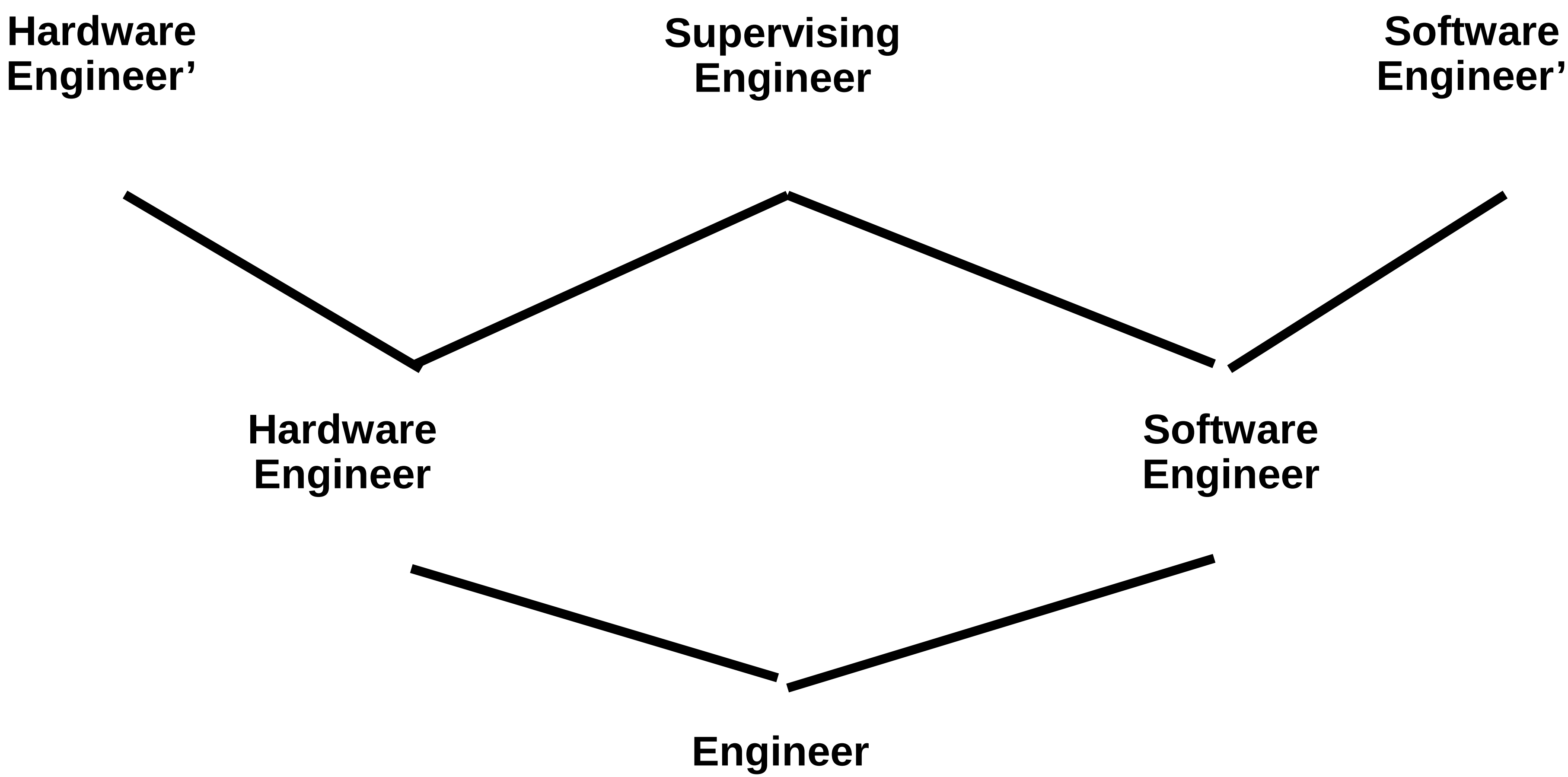
# HIERARCHICAL ROLES



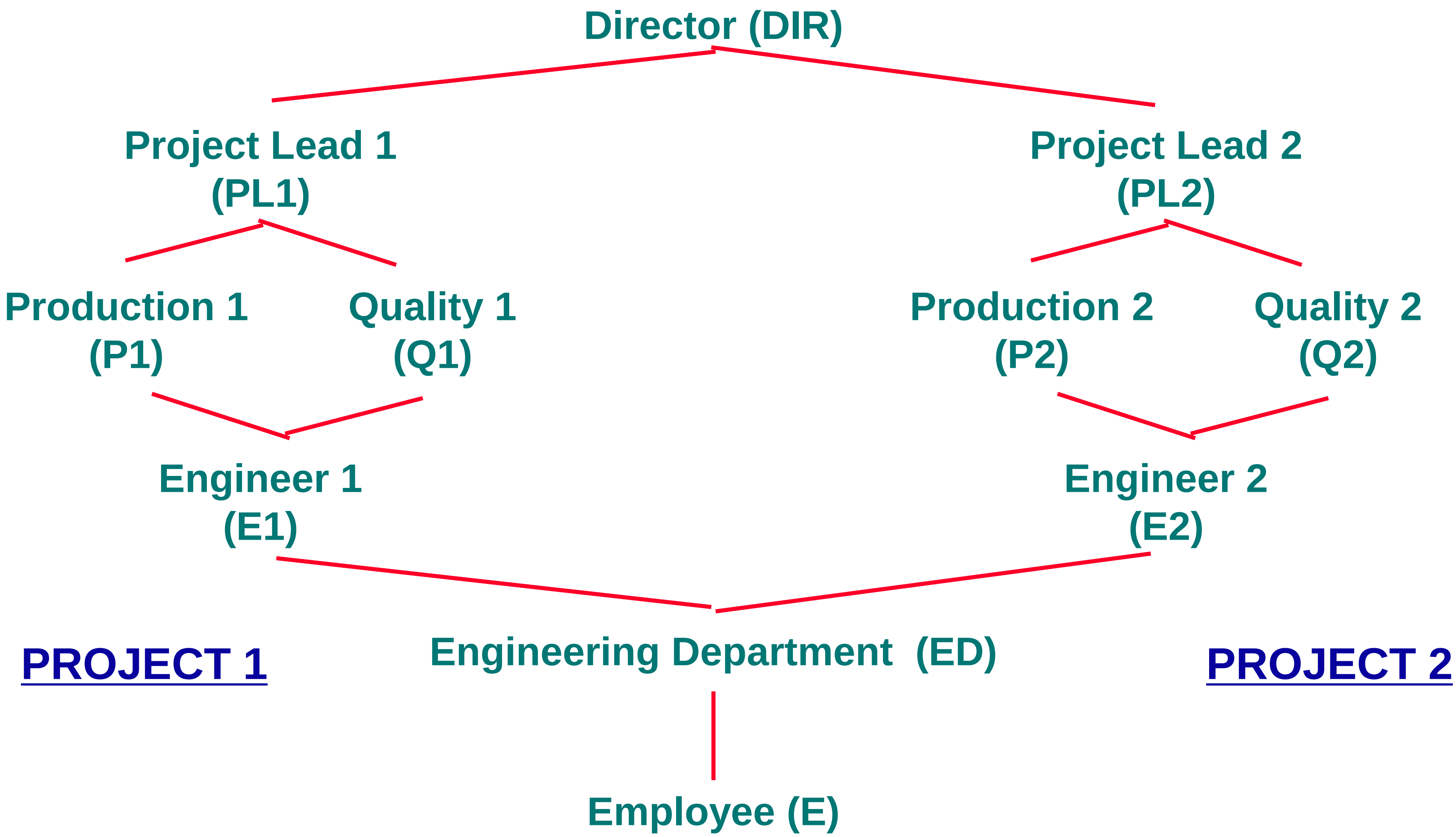
# HIERARCHICAL ROLES



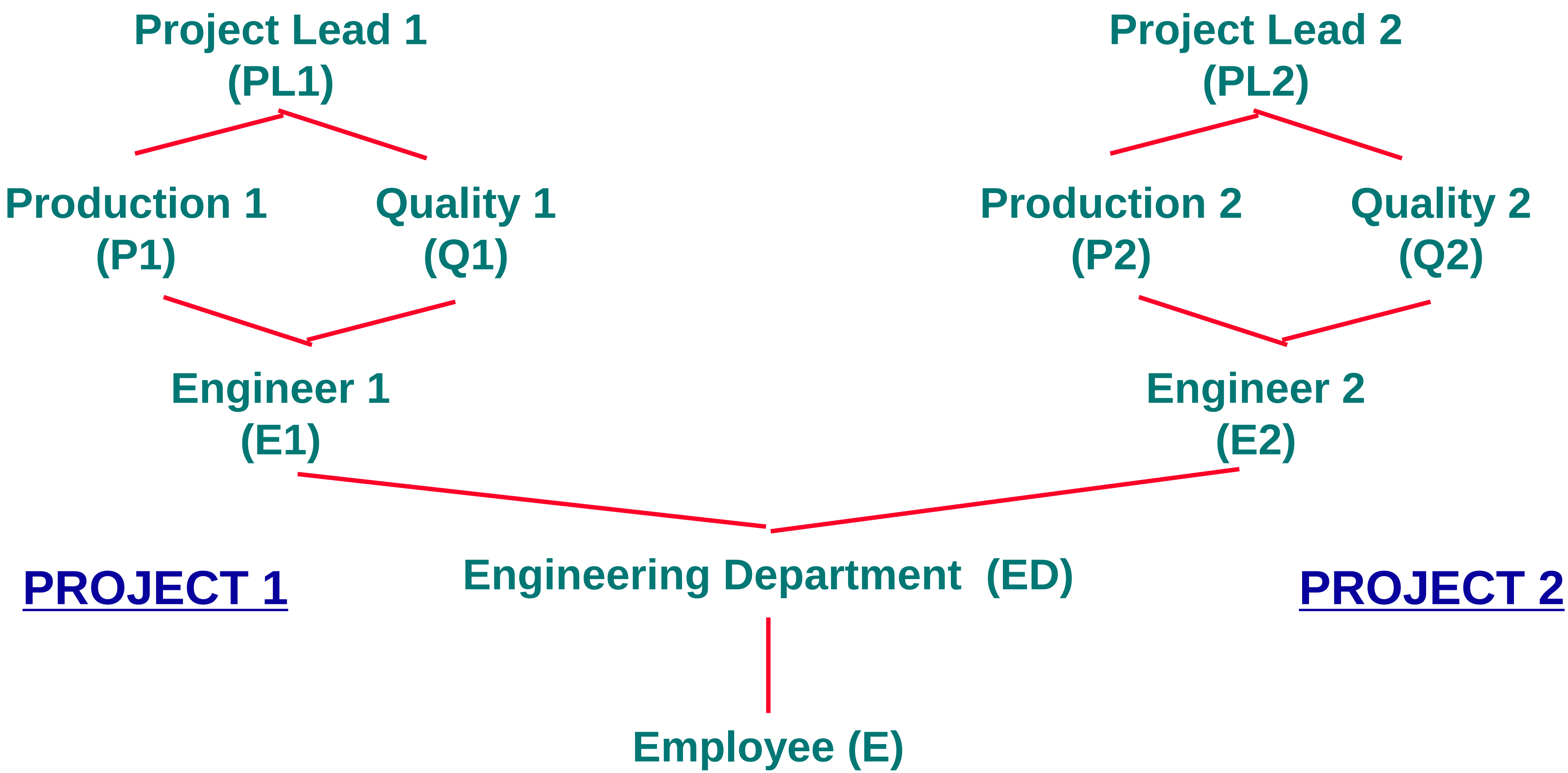
# PRIVATE ROLES



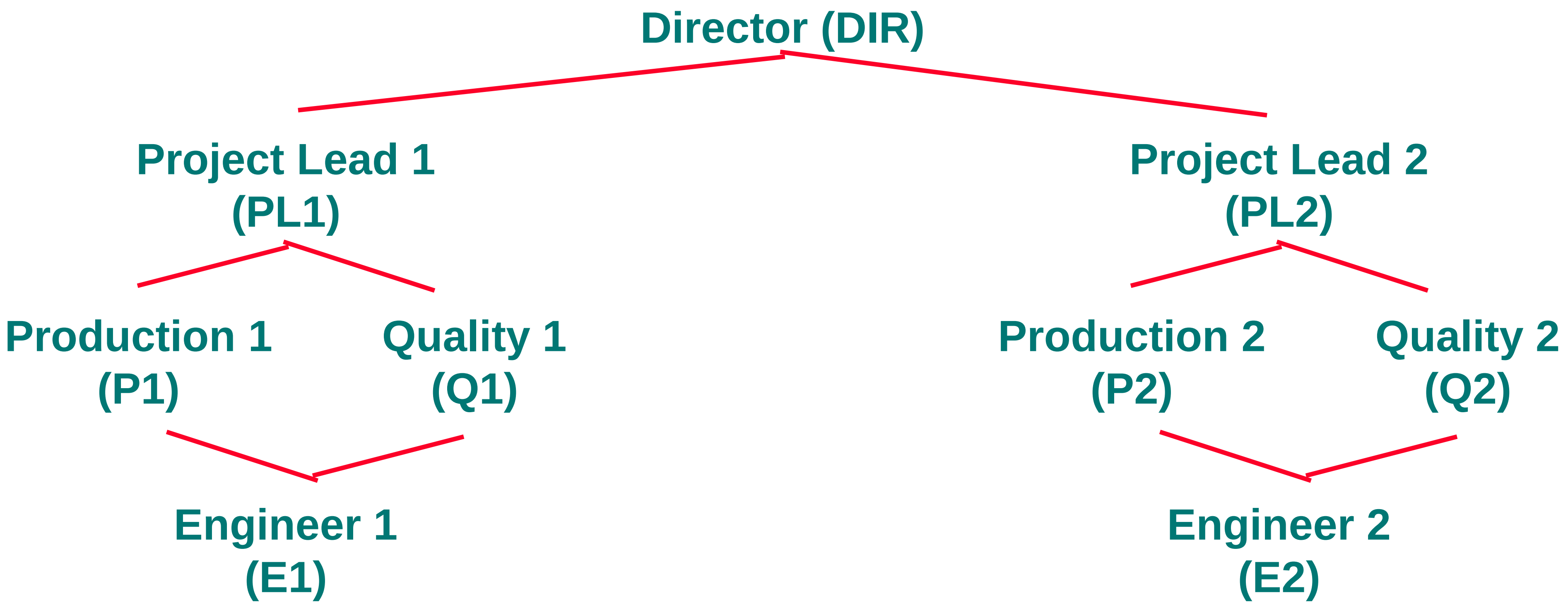
# EXAMPLE ROLE HIERARCHY



# EXAMPLE ROLE HIERARCHY



# EXAMPLE ROLE HIERARCHY




**PROJECT 1**

**PROJECT 2**

# EXAMPLE ROLE HIERARCHY

Project Lead 1
(PL1)

Production 1
(P1)

Quality 1
(Q1)

Engineer 1
(E1)

**PROJECT 1**

Project Lead 2
(PL2)

Production 2
(P2)

Quality 2
(Q2)

Engineer 2
(E2)

**PROJECT 2**

# CONSTRAINTS

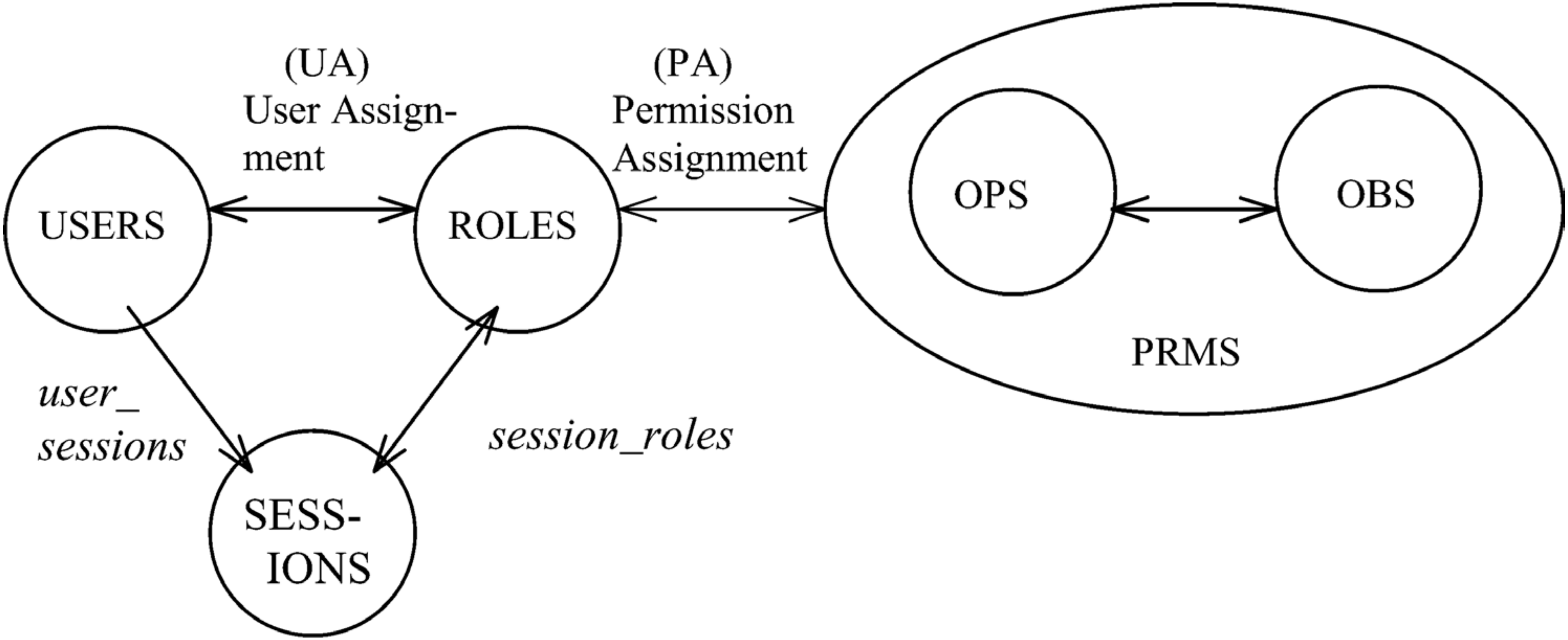


- Mutually Exclusive Roles
  - Static Exclusion: The same individual can never hold both roles
  - Dynamic Exclusion: The same individual can never hold both roles in the same context
- Mutually Exclusive Permissions
  - Static Exclusion: The same role should never be assigned both permissions
  - Dynamic Exclusion: The same role can never hold both permissions in the same context
- Cardinality Constraints on User-Role Assignment
  - At most k users can belong to the role
  - At least k users must belong to the role
  - Exactly k users must belong to the role
- Cardinality Constraints on Permissions-Role Assignment
  - At most k roles can get the permission
  - At least k roles must get the permission
  - Exactly k roles must get the permission

- Formalized in RCL2000 paper
  - Ahn, G. J., & Sandhu, R. (2000). Role-based authorization constraints specification. *ACM Transactions on Information and System Security (TISSEC)*, *3*(4), 207-226.

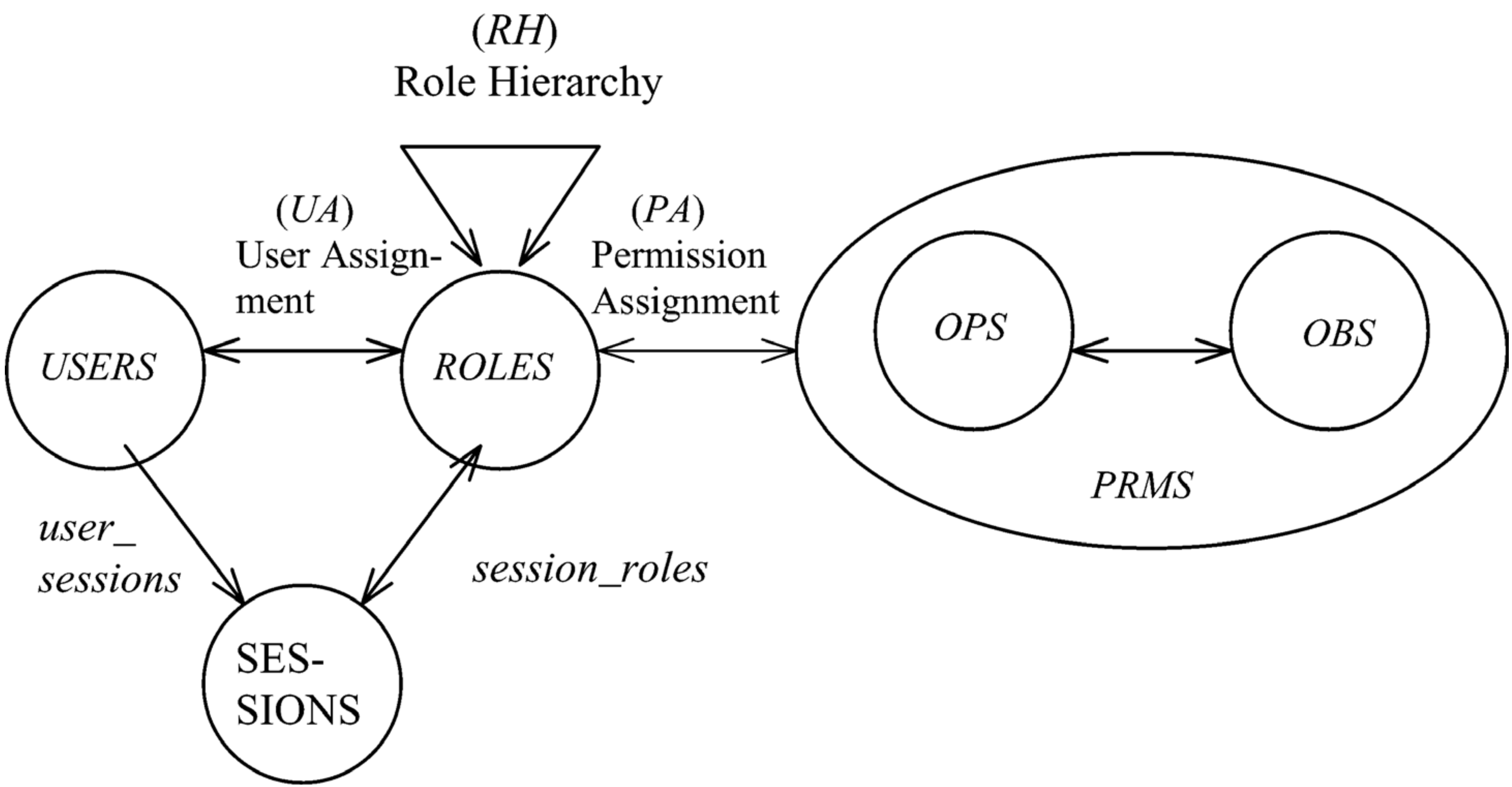
# NIST RBAC Model

# NIST MODEL: CORE RBAC



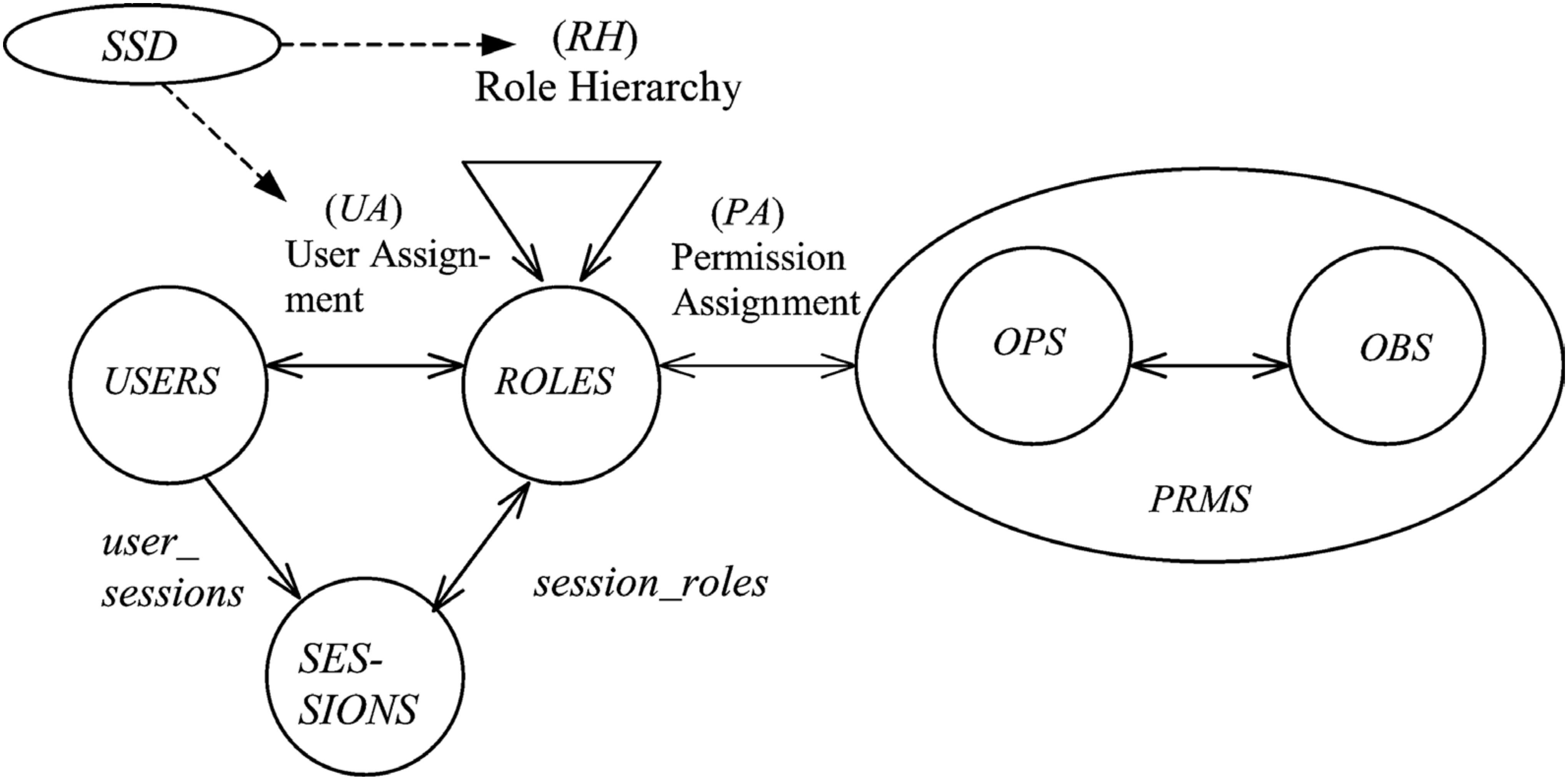
# NIST MODEL: HIERARCHICAL RBAC





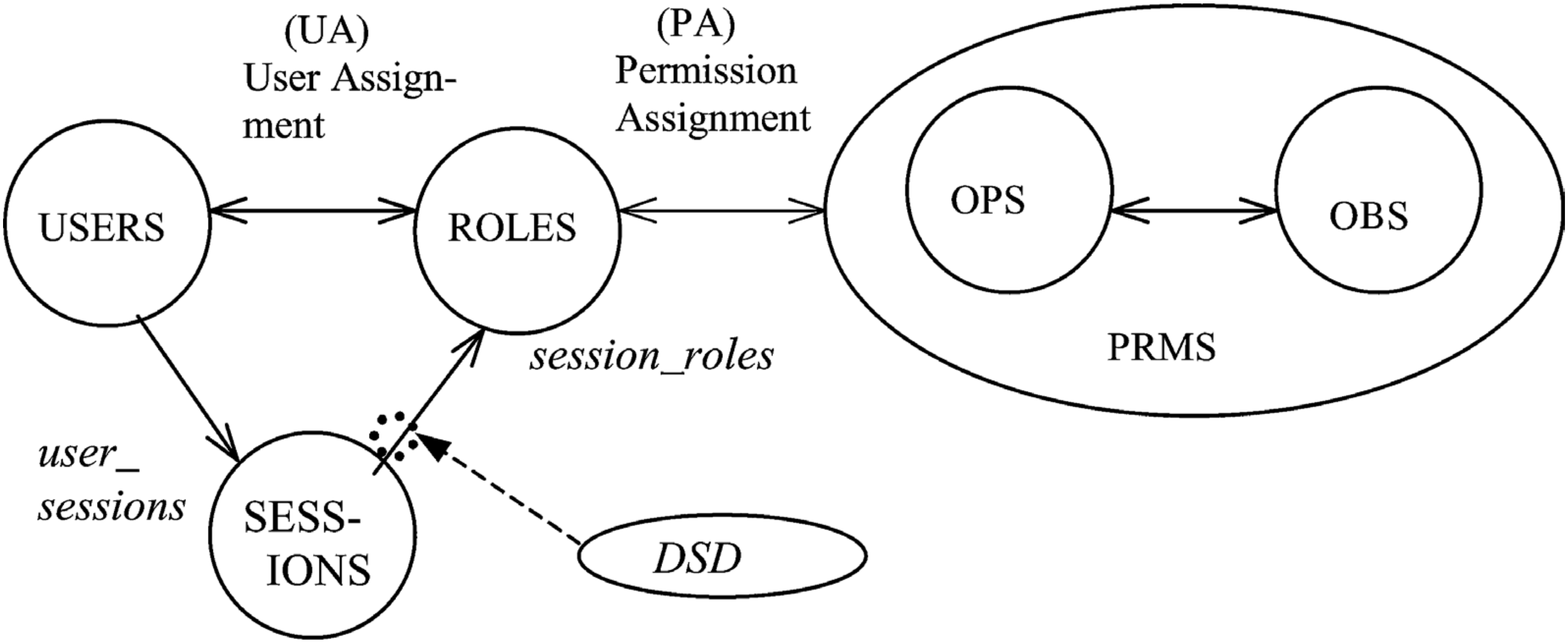
# SSD IN HIERARCHICAL RBAC



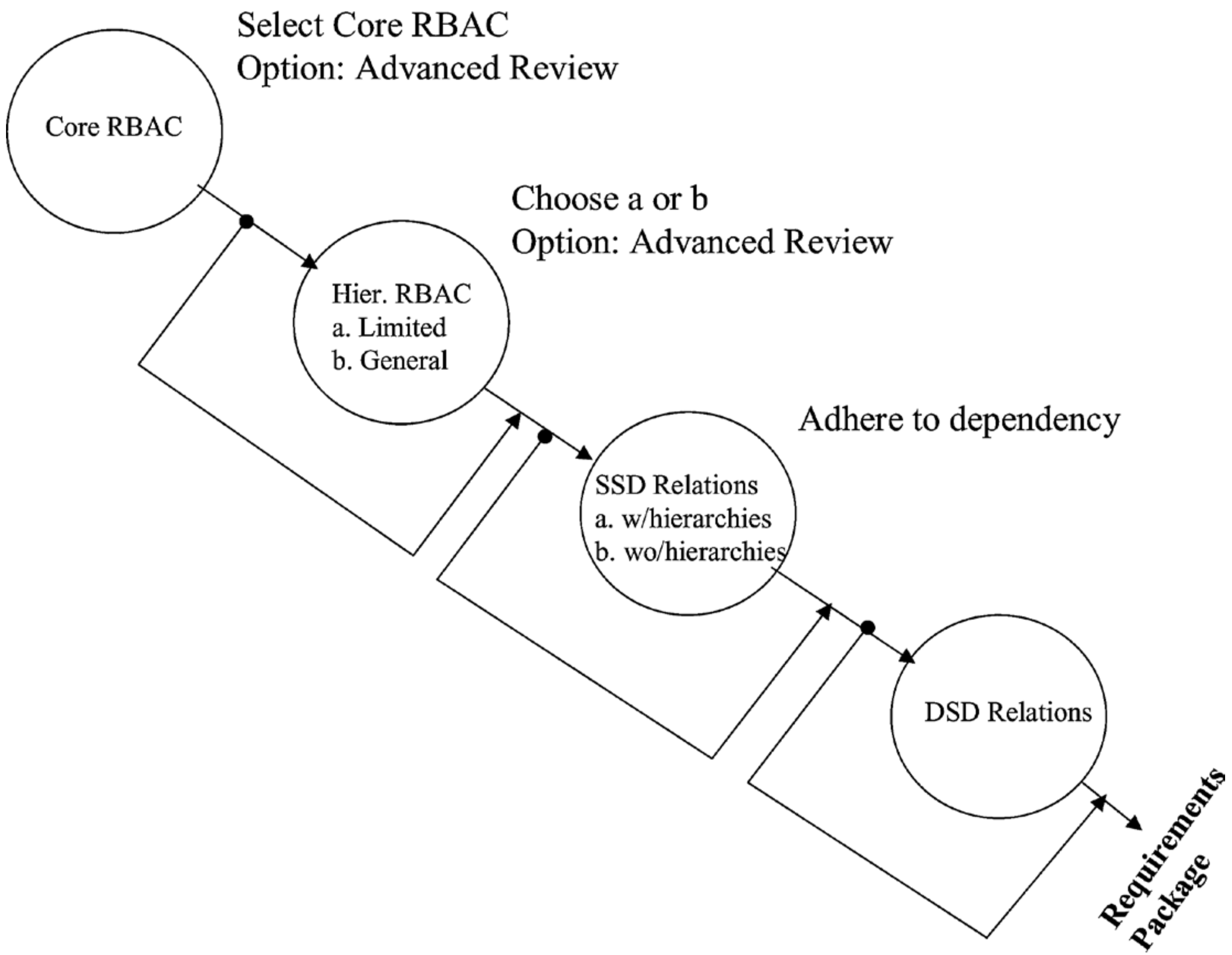


# DSD IN HIERARCHICAL RBAC

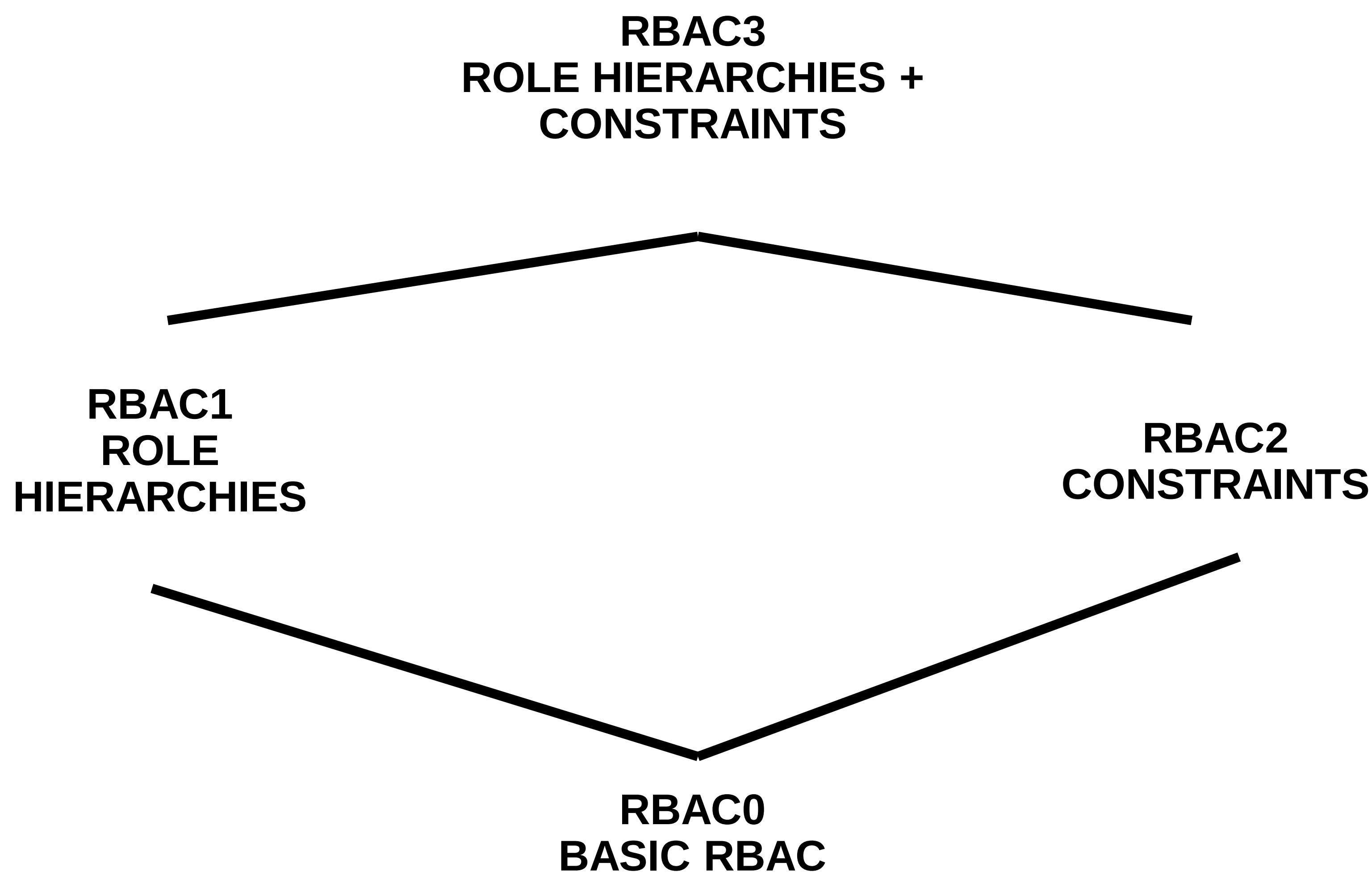




# NIST MODEL FAMILY



# Compare RBAC96 Model Family






*World-Leading Research with Real-World Impact!*

# RBAC Administration

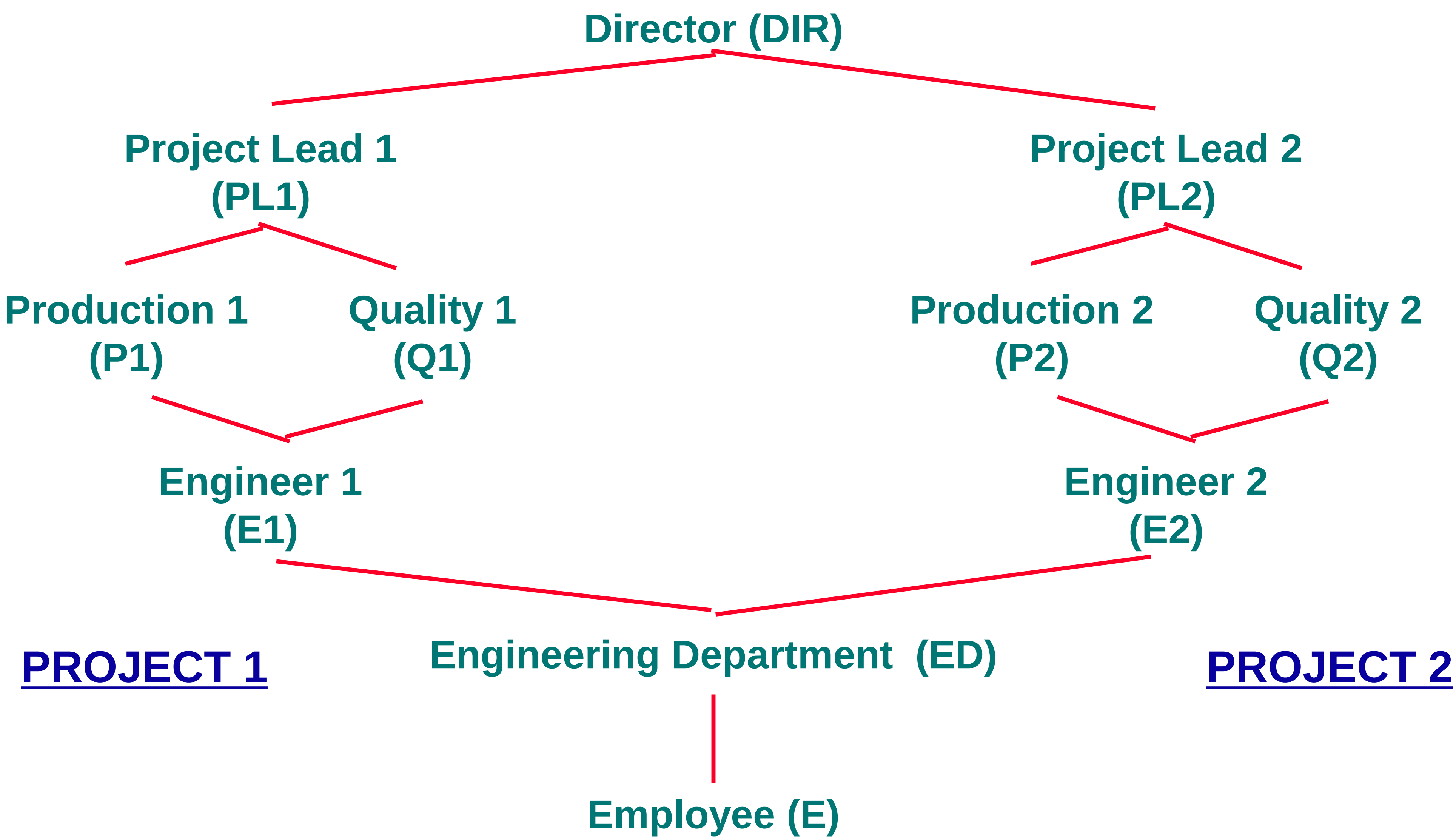
# ARBAC97 Model

- Separation of regular roles and administrative roles

- Formalized in ARBAC97 paper
  - Sandhu, R., Bhamidipati, V., & Munawer, Q. (1999). The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security (TISSEC)*, *2*(1), 105-135.

# EXAMPLE REGULAR ROLE HIERARCHY





# EXAMPLE ADMIN ROLE HIERARCHY

**Senior Security Officer (SSO)**

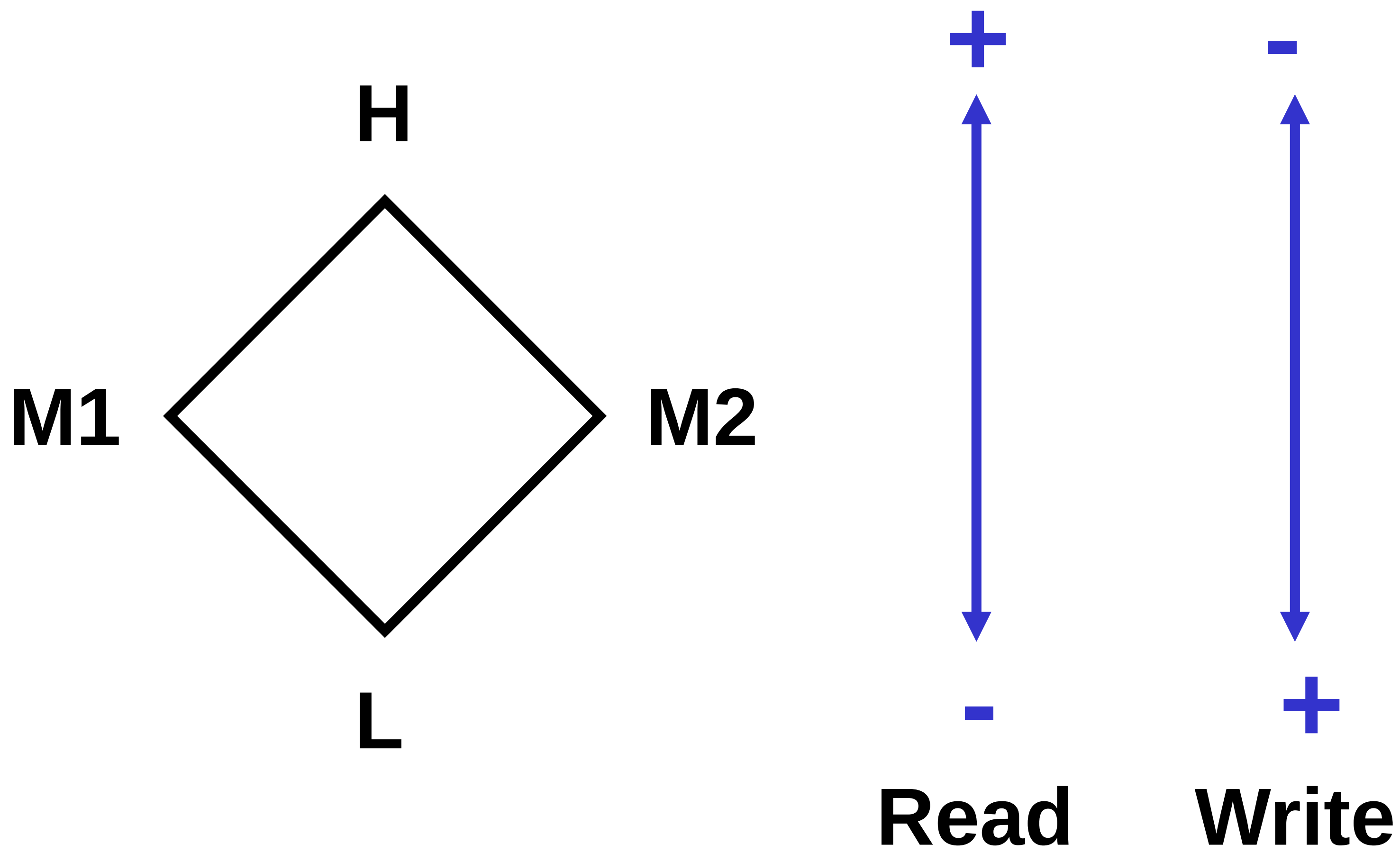
**Department Security Officer (DSO)**

**Project Security Officer 1 (PSO1)**

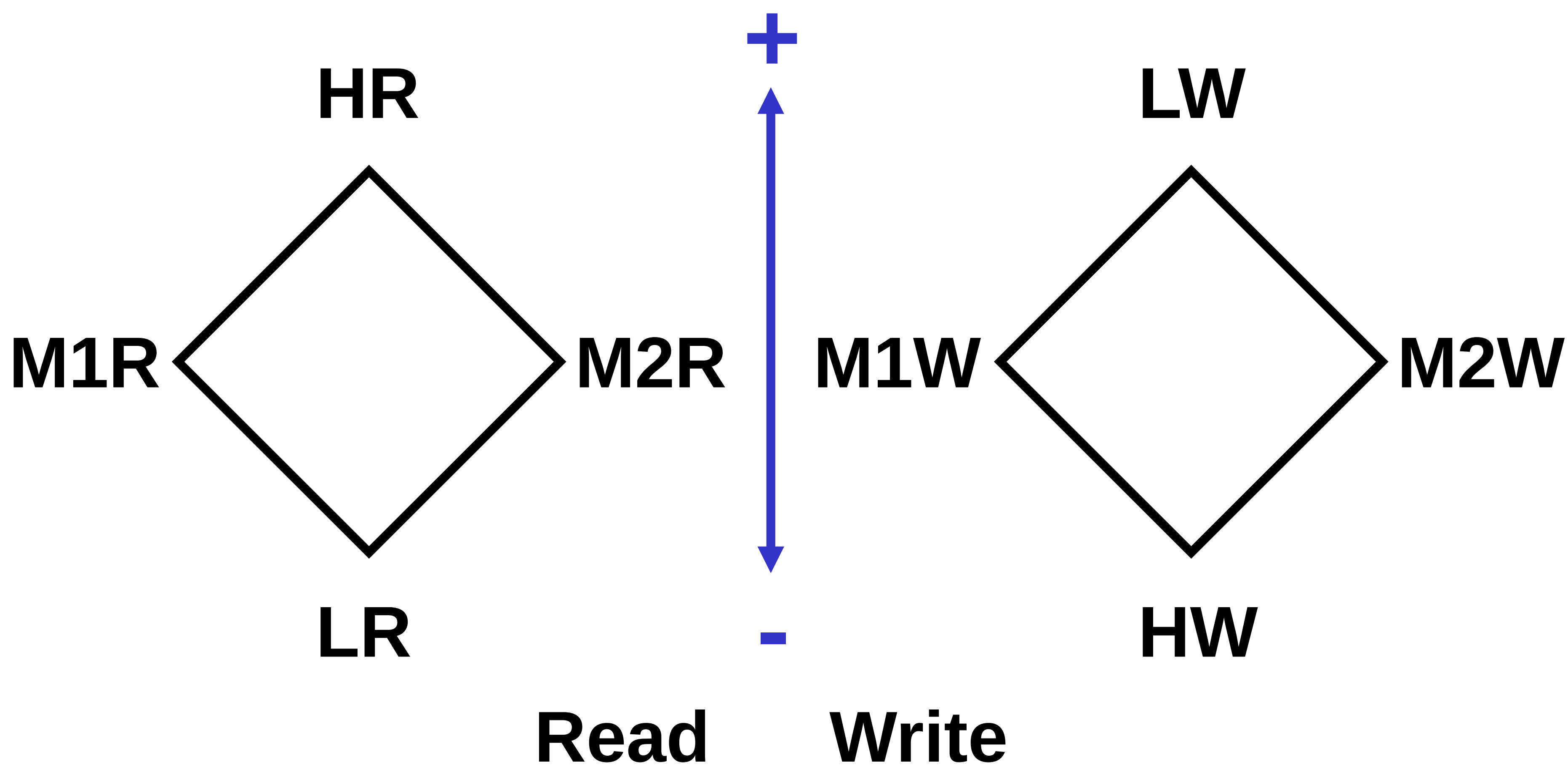
**Project Security Officer 2 (PSO2)**

# MAC in RBAC

# MAC Liberal ★-Property



# RBAC96 Liberal ★-Property



# RBAC96 Liberal ★-Property



- user $\in$ xR, user has clearance x
  user $\in$ LW, independent of clearance

- Constraints
  - session $\in$ xR iff session $\in$ xW
  - read can be assigned only to xR roles
  - write can be assigned only to xW roles
  - (O,read) assigned to xR iff
    (O,write) assigned to xW

# RBAC96 Liberal ★-Property

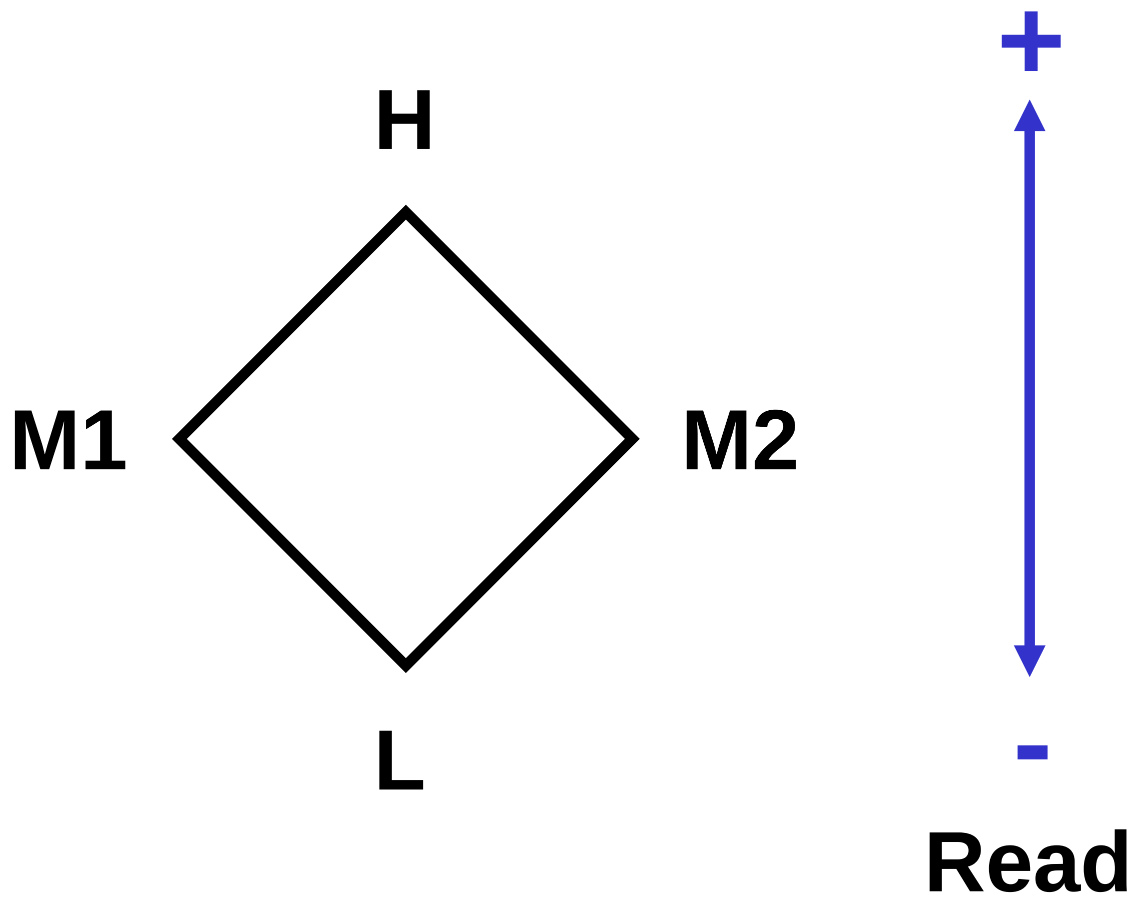


- user $\in$ xR, user has clearance x
  user $\in$ LW, independent of clearance

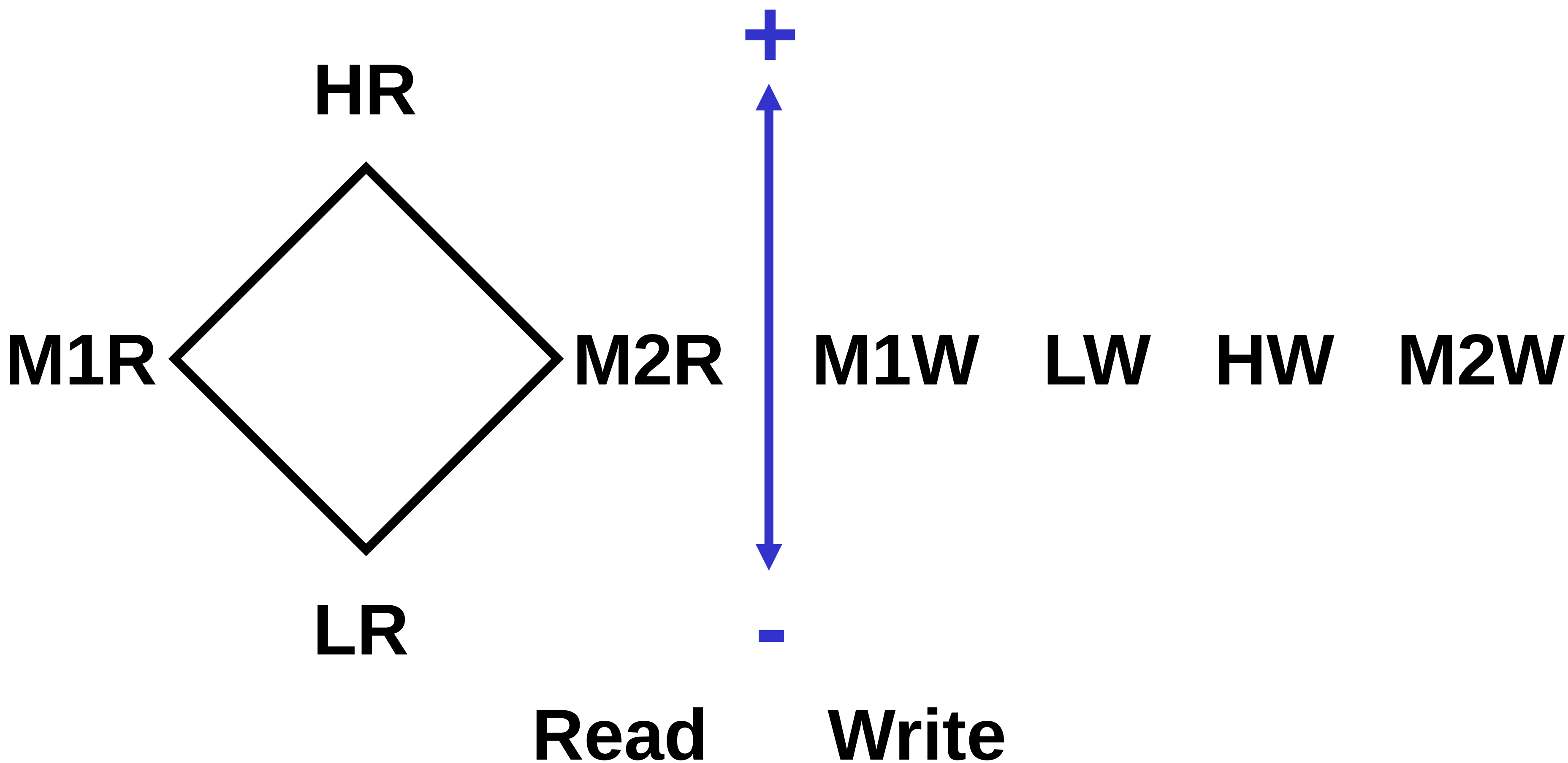
- Constraints
  - session $\in$ xR iff session $\in$ xW
  - read can be assigned only to xR roles
  - write can be assigned only to xW roles
  - (O,read) assigned to xR iff
    (O,write) assigned to xW

NIST Model cannot express these constraints

# MAC Strict ★-Property



# RBAC96 Strict ★-Property



# RBAC96 Strict ★-Property

- user $\in$ xR, user has clearance x
  user $\in$ {LW,HW,M1W,M2W}, independent of clearance

- Constraints
  - session $\in$ xR iff session $\in$ xW
  - read can be assigned only to xR roles
  - write can be assigned only to xW roles
  - (O,read) assigned to xR iff (O,write) assigned to xW

# DAC in RBAC

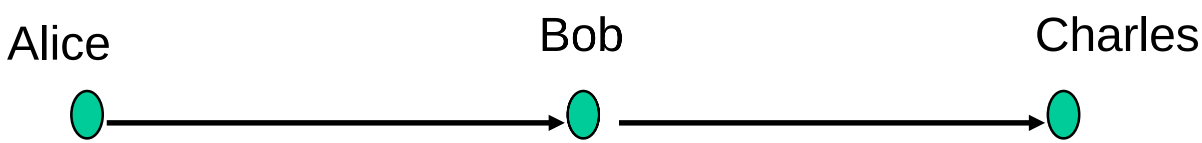
# Variations of Grant



- Strict DAC
  - Only owner has discretionary authority to grant access to an object.
  - Example:
    - Alice has created an object (she is owner) and grants access to Bob. Now Bob cannot grant propagate the access to another user.
- Liberal DAC
  - Owner can delegate discretionary authority for granting access to other users.
    - One Level grant
    - Two Level Grant
    - Multilevel Grant

# One-Level versus Two-Level-Grant

- Owner can delegate authority to another user but they cannot further delegate this power.




- In addition to a one level grant the owner can allow some users to delegate grant authority to other users.

# Variations of Revoke

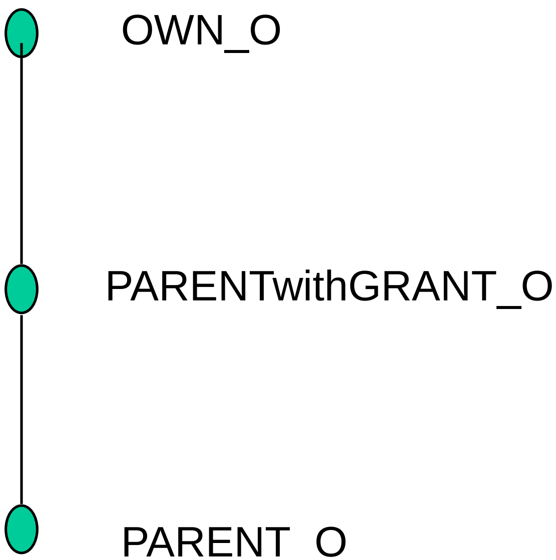
- Grant-Independent Revocation
  - Any authorized revoker can revoke
  - Easier to do in RBAC

- Grant-Dependent Revocation
  - Only original grantor can revoke
  - Need additional roles to accomplish in RBAC

# Common Aspects

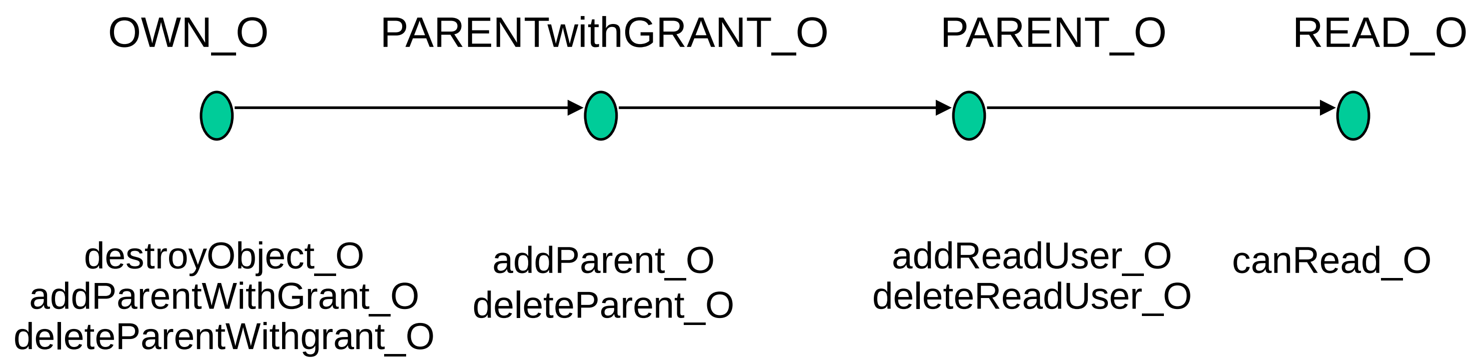
- Creation of an object O in the system requires the simultaneous creation of
  - 3 administrative roles
    - OWN_O, PARENT_O, PARENTwithGRANT_O
  - 1 regular role
    - READ_O
- Also simultaneous creation of 8 Permissions
  - canRead_O
  - destroyObject_O
  - addReadUser_O, deleteReadUser_O
  - addParent_O, deleteParent_O
  - addParentWithGrant_O, deleteParentWithGrant_O
- Destroying an object O requires deletion of 4 roles and 8 permissions in addition of destroying the object O

# Common Aspects




**Administrative role hierarchy**

**Administration of roles associated with object O**




**Role permissions**

# Grant Variations in RBAC96

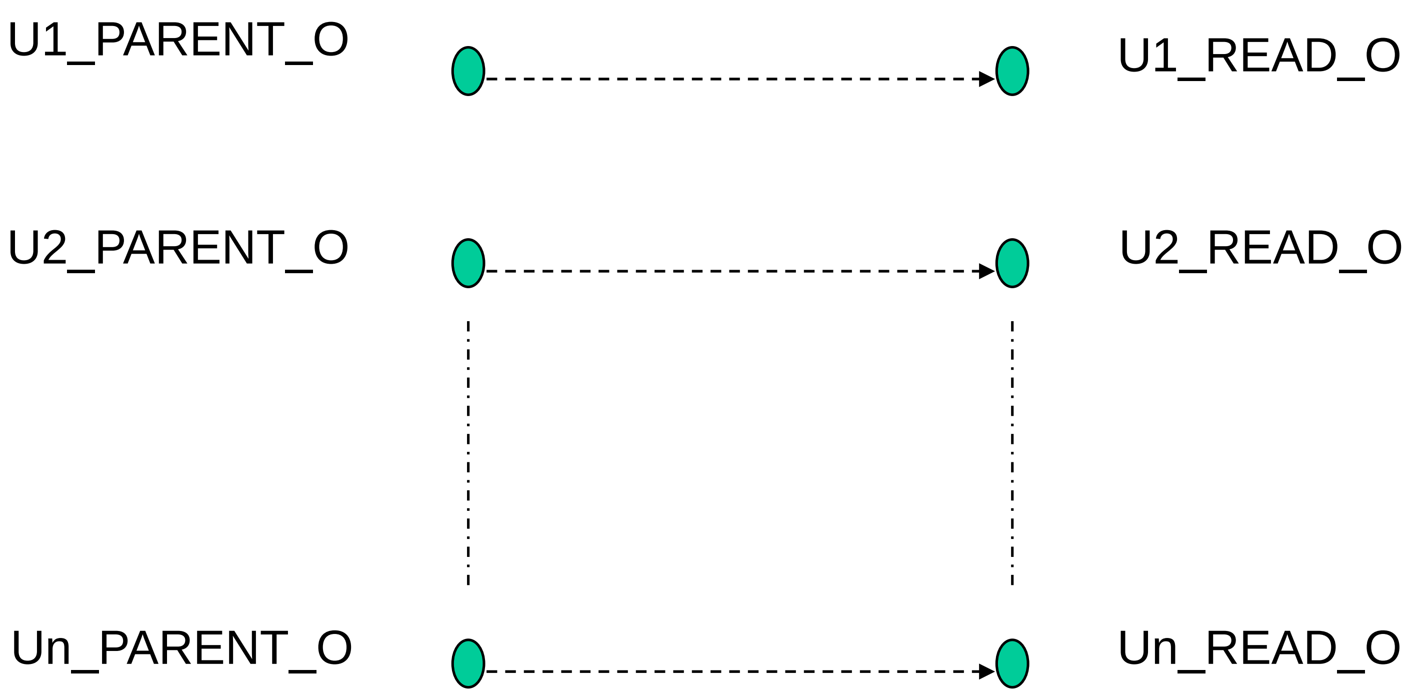
- Strict DAC cardinality constraints
  - Role OWN_O = 1
  - Role PARENTwithGRANT_O = 0
  - Role PARENT_O = 0
- One-level grant cardinality constraints
  - Role OWN_O = 1
  - Role PARENTwithGRANT_O = 0
- Two-level grant cardinality constraints
  - Role OWN_O = 1

# Grant Variations in RBAC96

- Strict DAC cardinality constraints
  - Role OWN_O = 1
  - Role PARENTwithGRANT_O = 0
  - Role PARENT_O = 0
- One-level grant cardinality constraints
  - Role OWN_O = 1
  - Role PARENTwithGRANT_O = 0
- Two-level grant cardinality constraints
  - Role OWN_O = 1

NIST Model cannot express these constraints

# Grant-Dependent Revoke in RBAC96

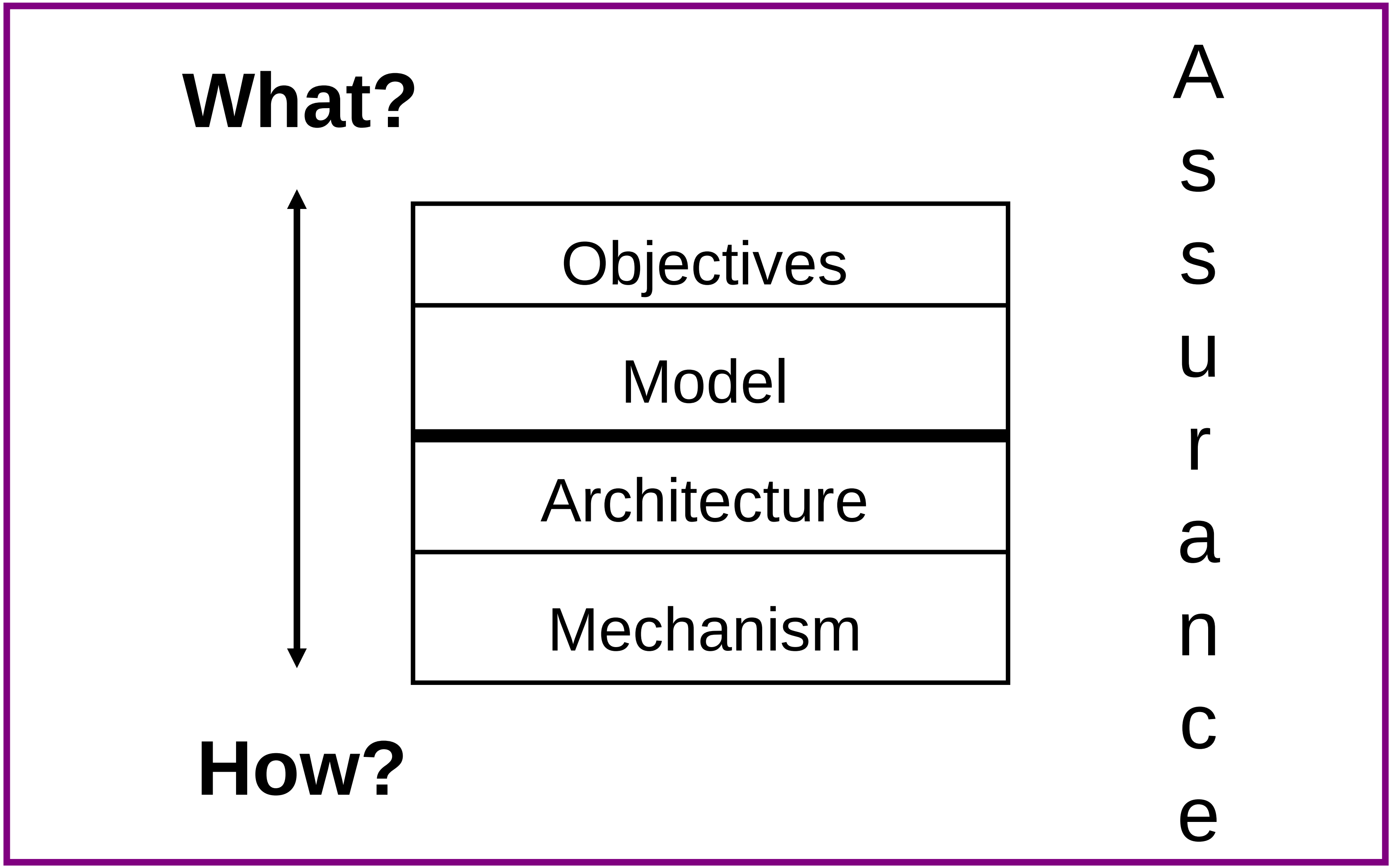



**READ_O role associated with members of PARENT_O**

# OM-AM and PEI

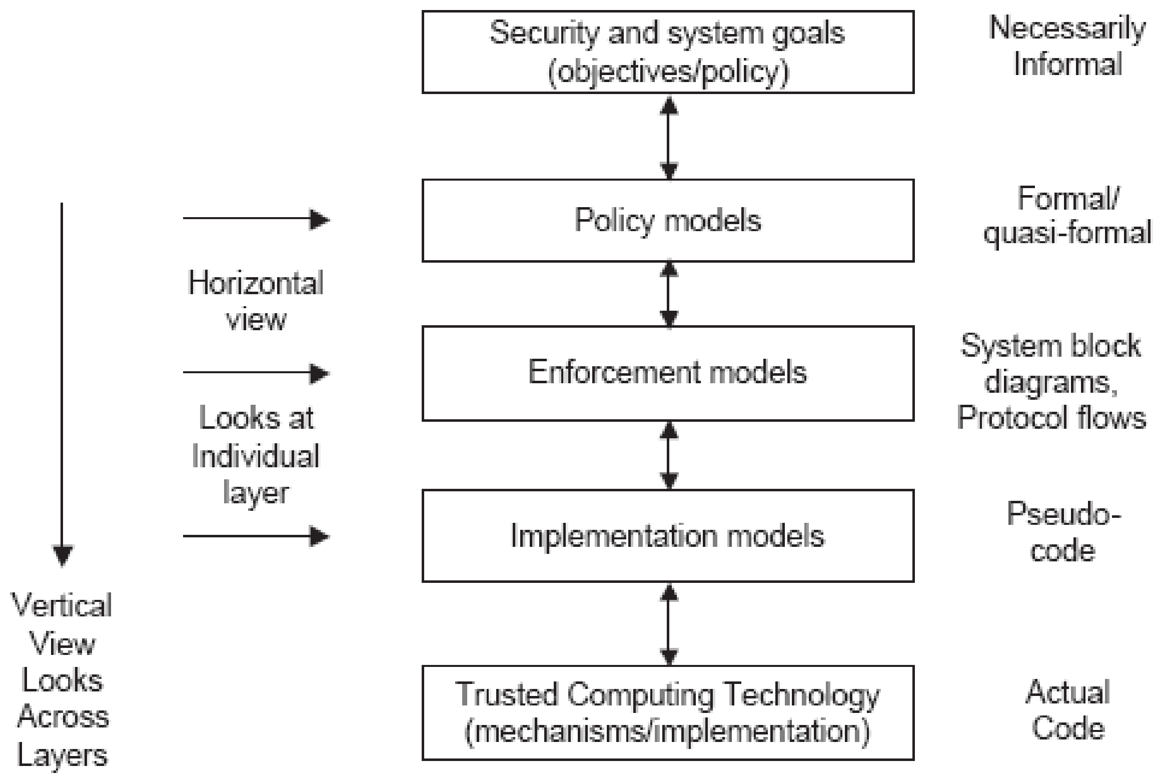
# OM-AM




What?
Objectives
Model
Architecture
Mechanism
How?
Assurance

# PEI Models





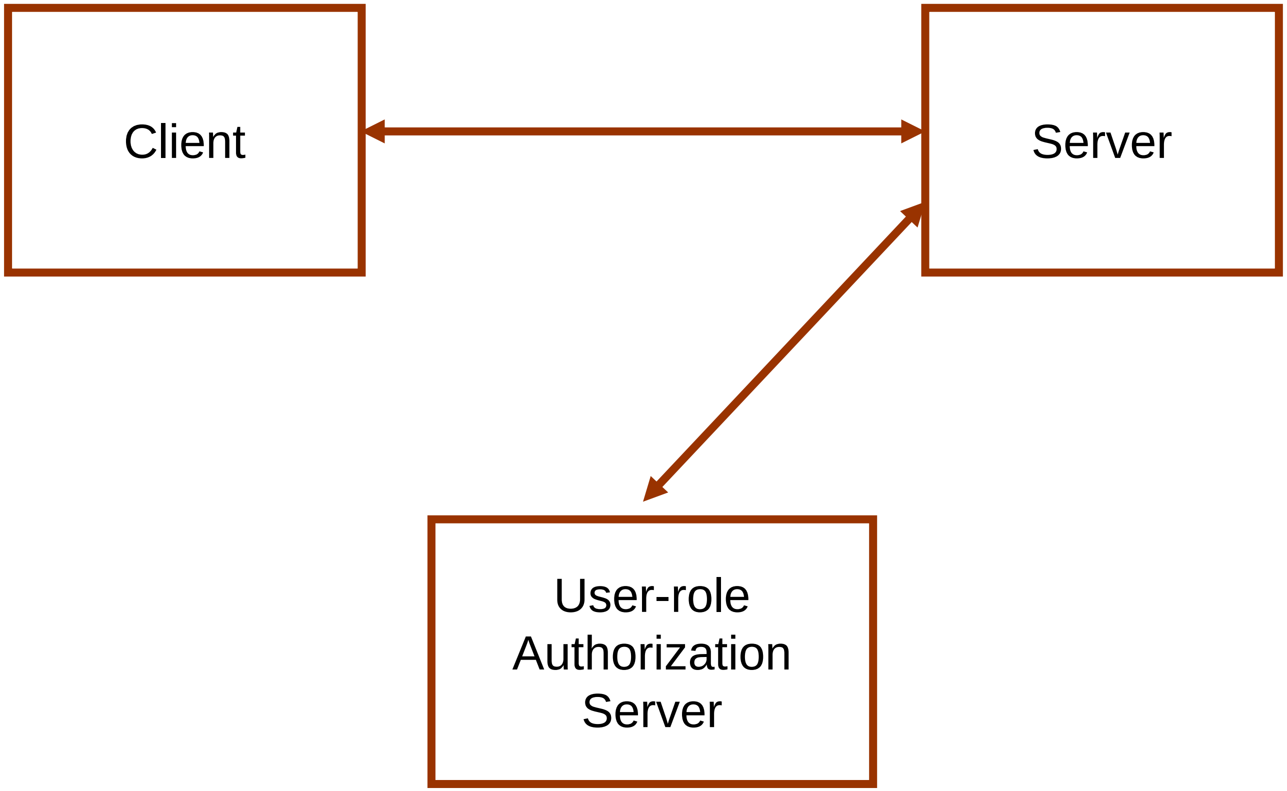

Idealized

Enforceable
(Approximate)

Codeable

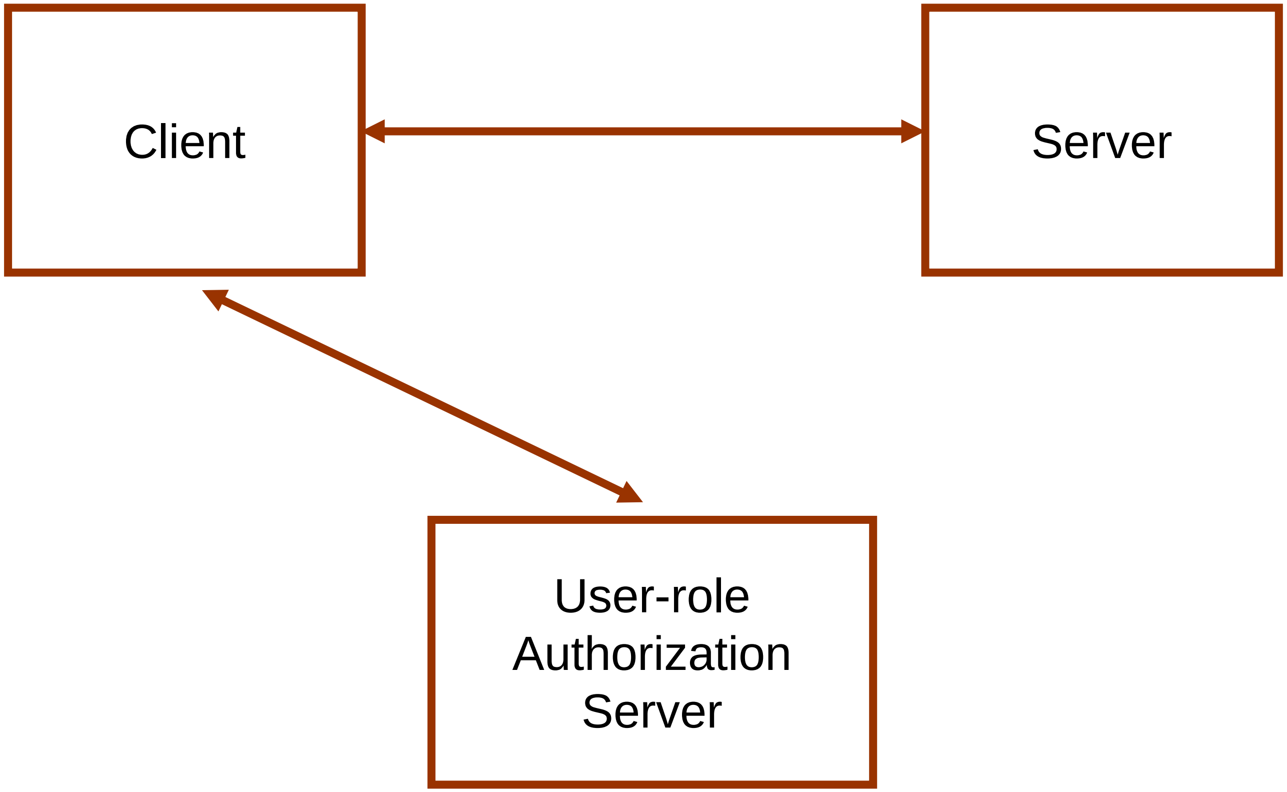
# RBAC: SERVER PULL

E model



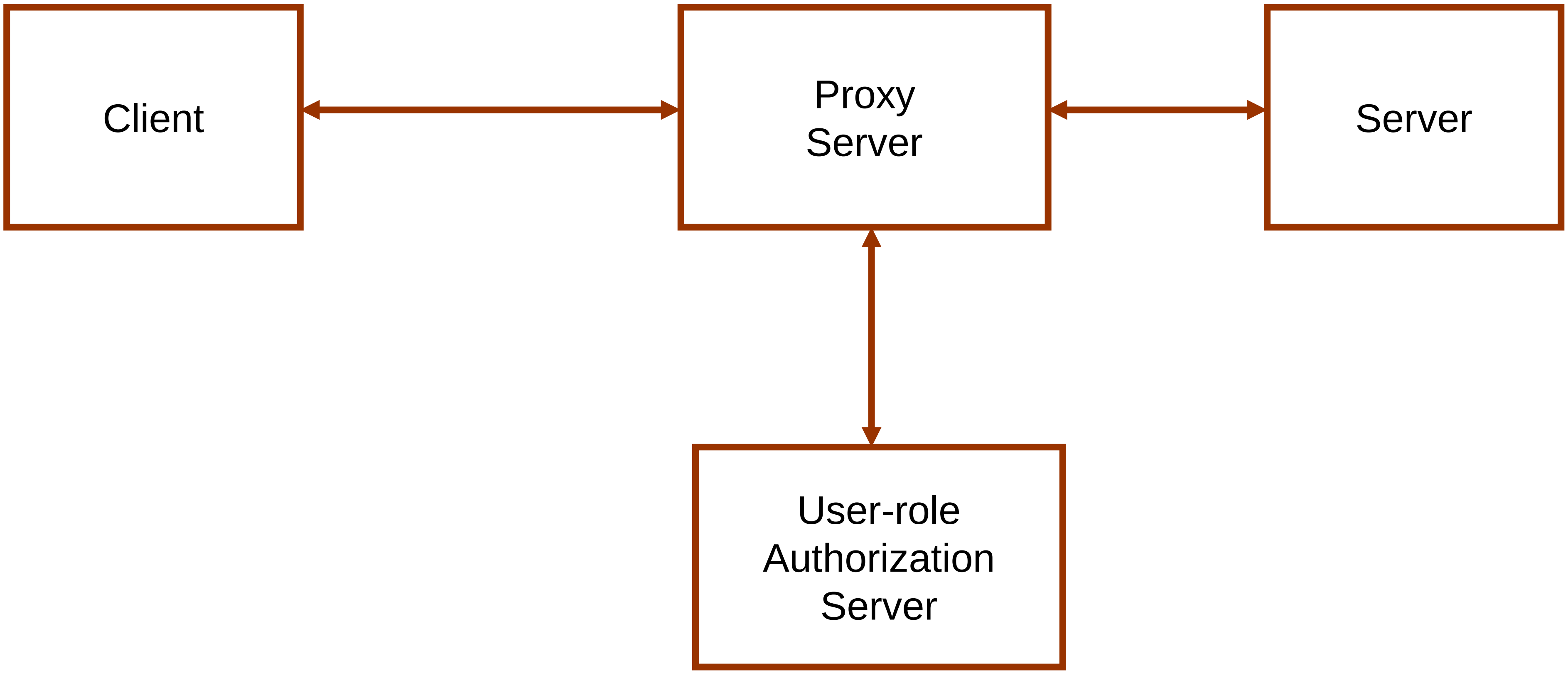
# RBAC: CLIENT PULL

E model



# RBAC: PROXY-BASED

E model



# MAC or LBAC or BLP (or Biba)

- BLP enforces one-directional information flow in a lattice of security labels

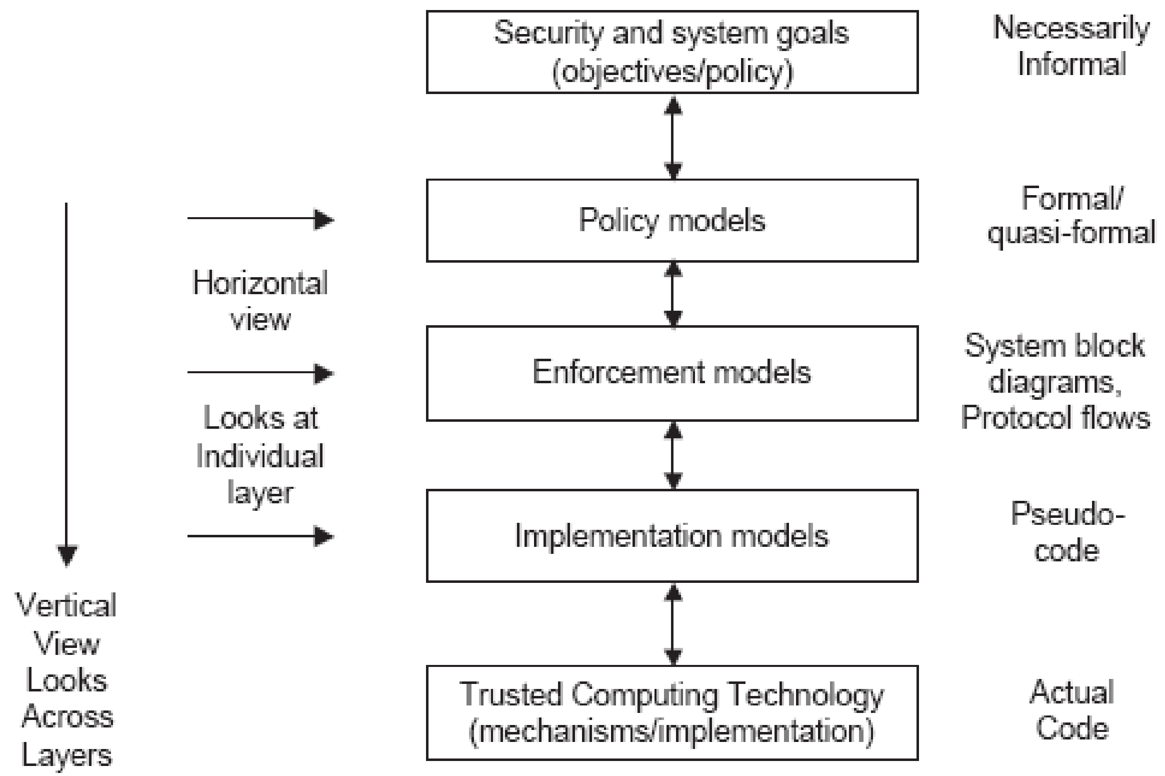
Policy

- BLP can enforce one-directional information flow policies for
  - Confidentiality
  - Integrity
  - Separation of duty
  - Combinations thereof

Objective

# MAC

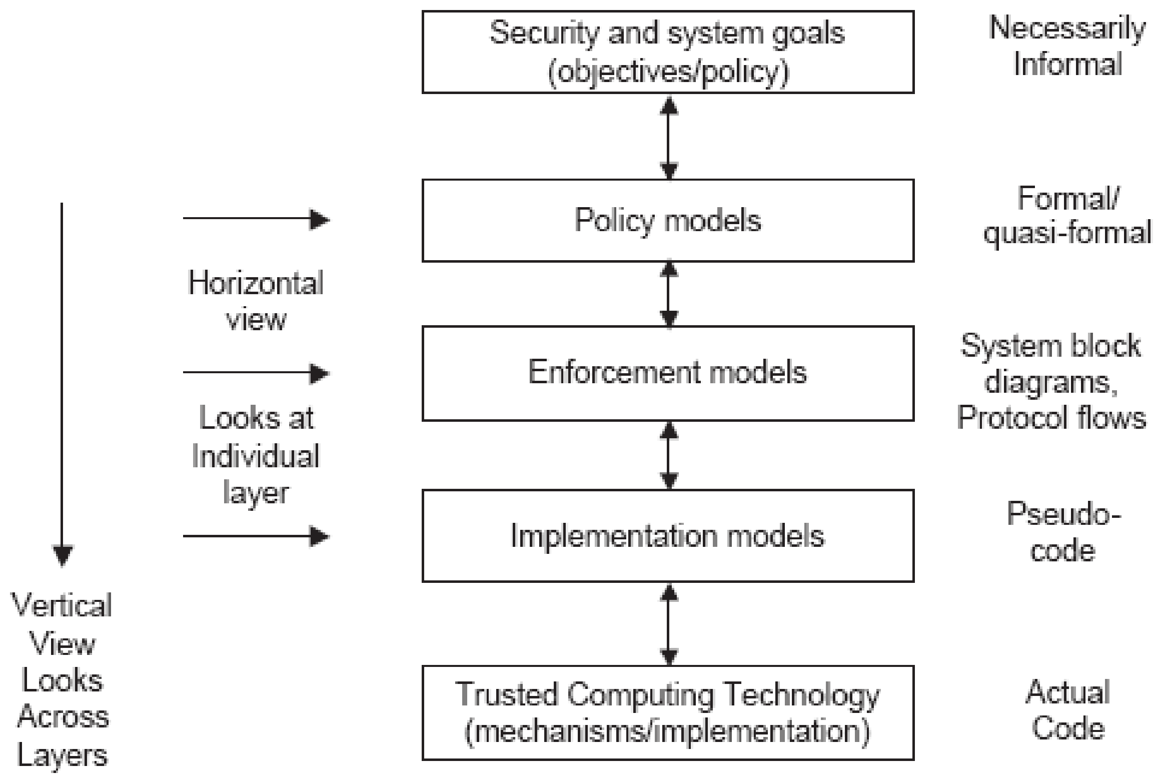





Confidentiality
Integrity
Separation

One-Direction
Information Flow

# DAC




Owner Discretion

Access Matrix

Capabilities,
Access Control Lists
Access Control Relations