Mandatory Security Policies: BLP

MAC vs DAC

Oregon State University College of Engineering

- Discretionary Access Control (DAC)
  - Access control is at the discretion of the user
  - Normal users can change access control state directly assuming they have appropriate permissions
  - E.g.: Access control implemented in standard OS's, e.g., Unix, Linux, Windows
- Mandatory Access Control (MAC)
  - Access decisions cannot be changed by normal users
  - Generally enforced by system wide set of rules
  - E.g.: SELinux, Windows Vista Integrity Levels
- "Strong" system security requires MAC
  - Normal users cannot be trusted

Confidentiality Policy

- Goal: prevent the unauthorized disclosure of information
  - Deals with information flow
  - Integrity incidental
- Multi-level security models are best-known examples
  - Bell-LaPadula Model (BLP) basis for many, or most, of these

Bell-LaPadula: Basics

- Subject and objects are associated with a security level
- Security levels
  - Most basic example of security class
- The levels are completely ordered
  - Example: Top secret > secret > confidential > restricted > unclassified
- The subject's level is security clearance
- The object's level is security classification

Security Level Example

Oregon State University
College of Engineering

Security Level	Subject	Object
Top Secret	Alice	Design for next generation iPhone
Secret	Bob	New pricing levels
Confidential	Carol	Current Financial Earnings
Unclassified	Dave	Current products list

BLP: Simple Security Property

Oregon State University
College of Engineering

- No Read Up
- Subject can only read an object of less or equal security level.
- Level(0) <= Level(S)

BLP: *-Property

Oregon State University College of Engineering

- No Write Down

- A subject can only “append” (write-only) into an object of greater or equal security level.
  - Level(S) <= Level(O) for “append”

- A subject can only read+write into an object of same security level
  - Level(S) = Level(O) for “read+write”

BLP: ds-Property

Oregon State University
College of Engineering

- A MAC system may also include a traditional discretionary access control check
  - DAC in MAC

- If *-property and simple security property checks pass, then also check the discretionary access rules

More Advanced Security Classes

- Simple linear ordering not adequate for larger systems

- Add set of categories to the security level to create a security label
  - E.g., top secret:{project1, project2}.
  - As clearance, subject is cleared to top secret only for project 1 and project 2 not project 3.

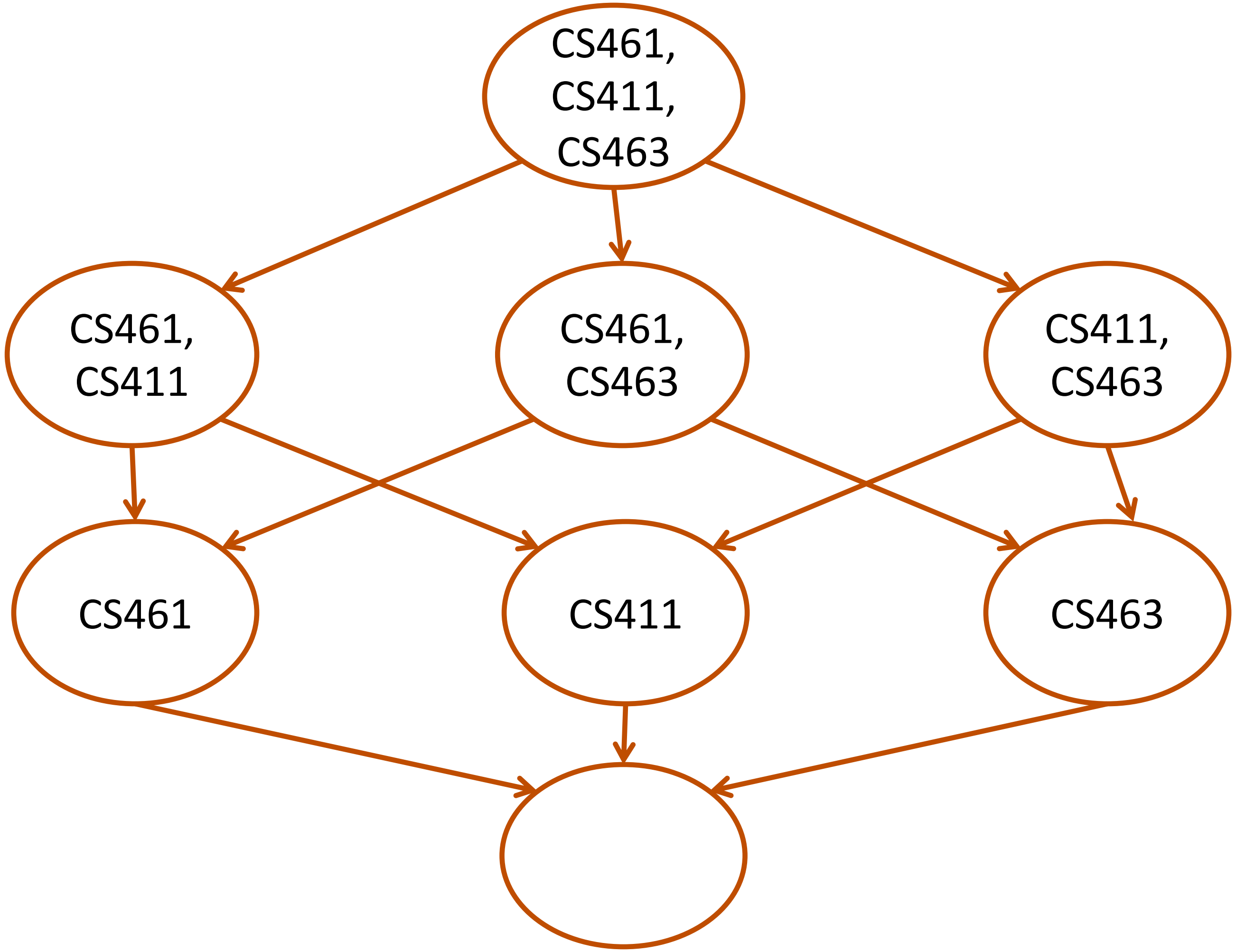
- Set of security labels forms a partial ordering or a lattice

Comparing Security Labels

- (A1, C1) "dominates" (A2, C2) iff A2 <= A1 and C2 subset of C1

- Replace "=>" with "dominates" and simple security condition and *-property hold
  - Simple Security Property: Subject s can only read an object o if Label(s) dominates Label (o)
  - *-Property: A subject s can append into an object o only if Label (o) "dominates" Label (s); A subject s can read+write into an object o only if Label (o) is same as Label (s)

Example Lattice of Categories



Security Label Comparisons

- Instructor Label = Secret:{461, 498}
- TA Label = Secret:{461}
- Student label = Confidential:{461}
- Instructor writes exam for CS461
  - What label should it have, so TA can help write?
    - A. Secret: {461}
    - B. Confidential:{461}
    - C. Public:{461}
    - D. Top Secret:{498}

Top secret > secret > confidential > restricted > unclassified

Security Label Comparisons

- Instructor Label = Secret:{461, 498}
- TA Label = Secret:{461}
- Student label = Confidential:{461}
- Instructor writes exam for 461
  - What label should it have for student to read exam?
    A. Secret:{461}
    B. Confidential:{461}
    C. Top Secret:{461}
    D. None of the above

Top secret > secret > confidential > restricted > unclassified

Adding Security Clearance Flexibility

Oregon State University
College of Engineering

- Define maximum and current level for subjects
  - maxlevel(s) dominates curlevel(s)
  - In some systems, the min level is also defined

- How does this ease the previous example?

Principle and Types of Tranquility

Oregon State University
College of Engineering

- Strong tranquility
  - The clearances of subjects, and the classification of objects, do not change during the lifetime of the system

- Weak tranquility
  - The clearances of subjects and the classifications of the objects change in accordance with a specified policy.

Principle of Tranquility

Oregon State University
College of Engineering

- Raising object's security level
  - Information once available to some subjects is no longer available
  - Usually assume information has already been accessed, so questionable protection
- Lowering object's security level
  - The declassification problem
  - Essentially, a write down, violates *-property

Summary

- Mandatory Security Policies
  - Designed by security expert
  - Can be awkward for users

- Different models address different goals
  - E.g., BLP is a confidentiality model

BIBA Integrity Model

Intuition for Integrity Levels

- The higher the level, the more confidence
  - E.g. - that a program will execute correctly
  - E.g. - that data is accurate and/or reliable
- Note relationship between integrity and trustworthiness
- Important point: *integrity levels are **not** security levels*
- Integrity models finding use in modern Operating Systems

Integrity Level Example

Oregon State University
College of Engineering

Integrity Level	Subject	Object
Highly Trusted	Alice	System Software
Trusted	Bob	Software Signed by the OS Provider or other Trusted provider
Untrusted	Dave	Software downloaded from Internet

Strict Integrity Policy – Biba Model

- Simple Integrity Property
  - $s \in S$ can write to $o \in O$ iff $i(o) \leq i(s)$
  - No write up

- Integrity Confinement Property
  - $s \in S$ can read $o \in O$ iff $i(s) \leq i(o)$
  - No read down

- Invocation Property
  - $s_1 \in S$ can invoke $s_2 \in S$ iff $i(s_2) \leq i(s_1)$

- Dual of Bell-LaPadula model
  - Add compartments and discretionary controls to get full dual of Bell-LaPadula model

More Advanced Integrity Labels

Oregon State University
College of Engineering

- Add set of categories to the integrity level to create integrity label
  - E.g., Trusted:{project1, project2}.
  - The information in object is trusted only for project 1 and project 2.
- Set of integrity labels forms a partial ordering or a lattice

Comparing Integrity Labels

- (A1, C1) “dominates” (A2, C2) iff A2 <= A1 and C2 subset of C1
- Replace “=>” with “dominates” in simple integrity, integrity confinement and invocation properties
  - Simple Integrity Property: Subject s can **write** to an object o only if Label(s) “**dominates”** Label (o)
  - Integrity Confinement Property: A subject s can **read** an object o only if Label (o) **“dominates”** Label (s)
  - Invocation Property: A subject s_1 can **invoke** a subject s_2 only if Label (s_1) **“dominates”** Label (s_2)

Integrity Models in Practice

Oregon State University
College of Engineering

- Integrity Levels introduced in Windows Vista as Mandatory Integrity Control (MIC)

- Windows has 4 integrity levels:
  - Low < medium < high < system
  - Standard users – medium integrity level
  - System services – system integrity level
  - Object with no level associated are treated as “medium integrity level”

- Anything downloaded from the Internet is defaulted to “low” so that such software cannot corrupt medium, high and system files.
  - Note such software may still be able to read your data!

Summary

Oregon State University
College of Engineering

- BIBA is a mandatory access control policy model focused on integrity
- It is a dual of Bell-LaPadula (BLP) that focuses on confidentiality
- Integrity levels are being used in modern operating systems to prevent corruption of system files

Chinese Wall Model

Chinese Wall Model

Oregon State University
College of Engineering

- Addresses enforcing **conflict of interest** policies

- Example
  - Consultant knows sensitive information about Company A
  - Consultant's company also works with Company B, a competitor to Company A
  - Consultant shouldn't access information about Company B

- Other Examples:
  - between corporate advisors and brokers in investment banks
  - between editorial/news and advertising divisions
  - between lawyers representing the defendant and plaintiff from the same company/firm

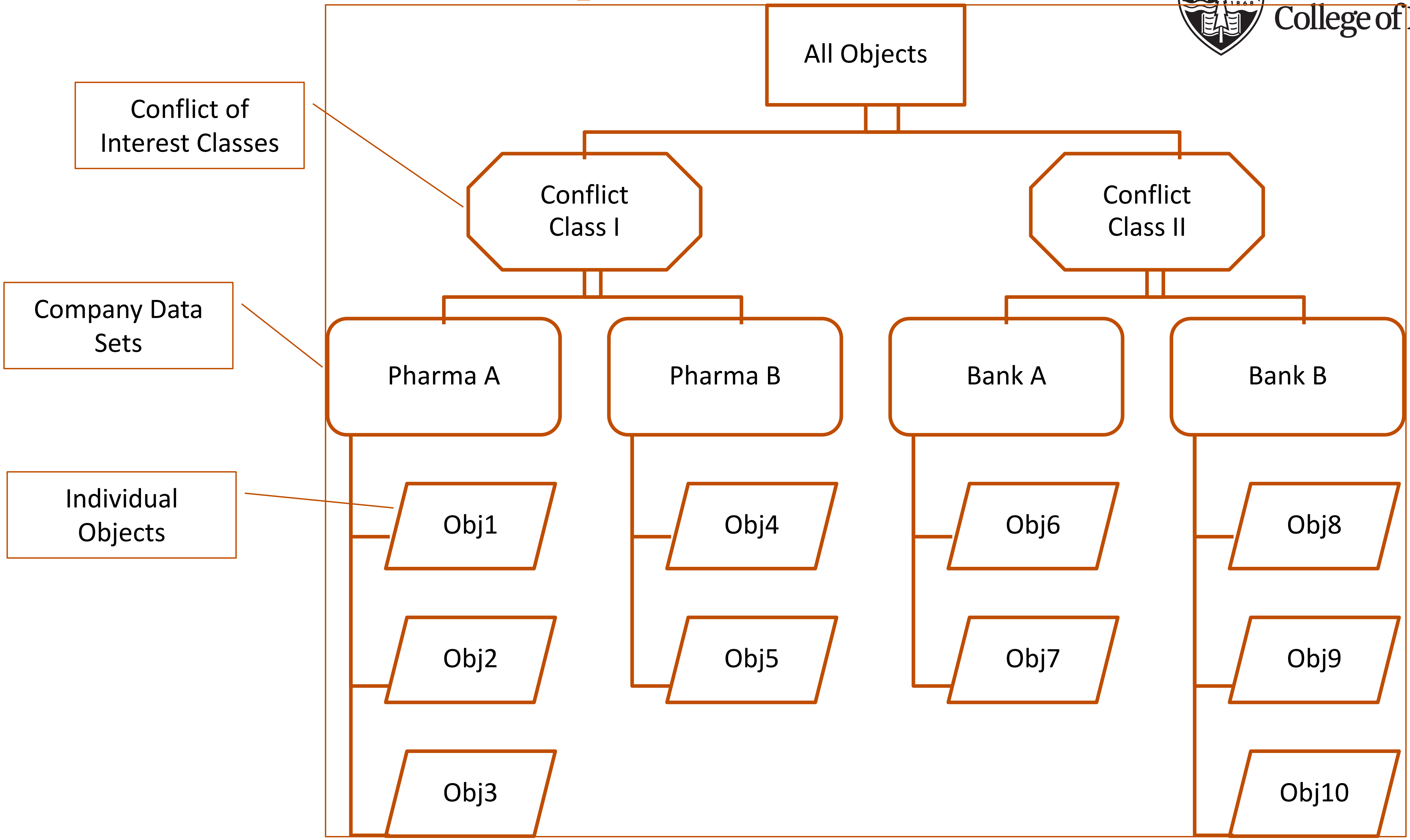
Definitions

- Subjects – Active entities

- Information
  - Objects – Individual items
  - Dataset (DS) – All objects that concern the same corporation
  - Conflict of interest (CI) class – All datasets whose corporations are in competition

- Access rules – rules for read and write access

Chinese Wall Example



Simple Security Rule

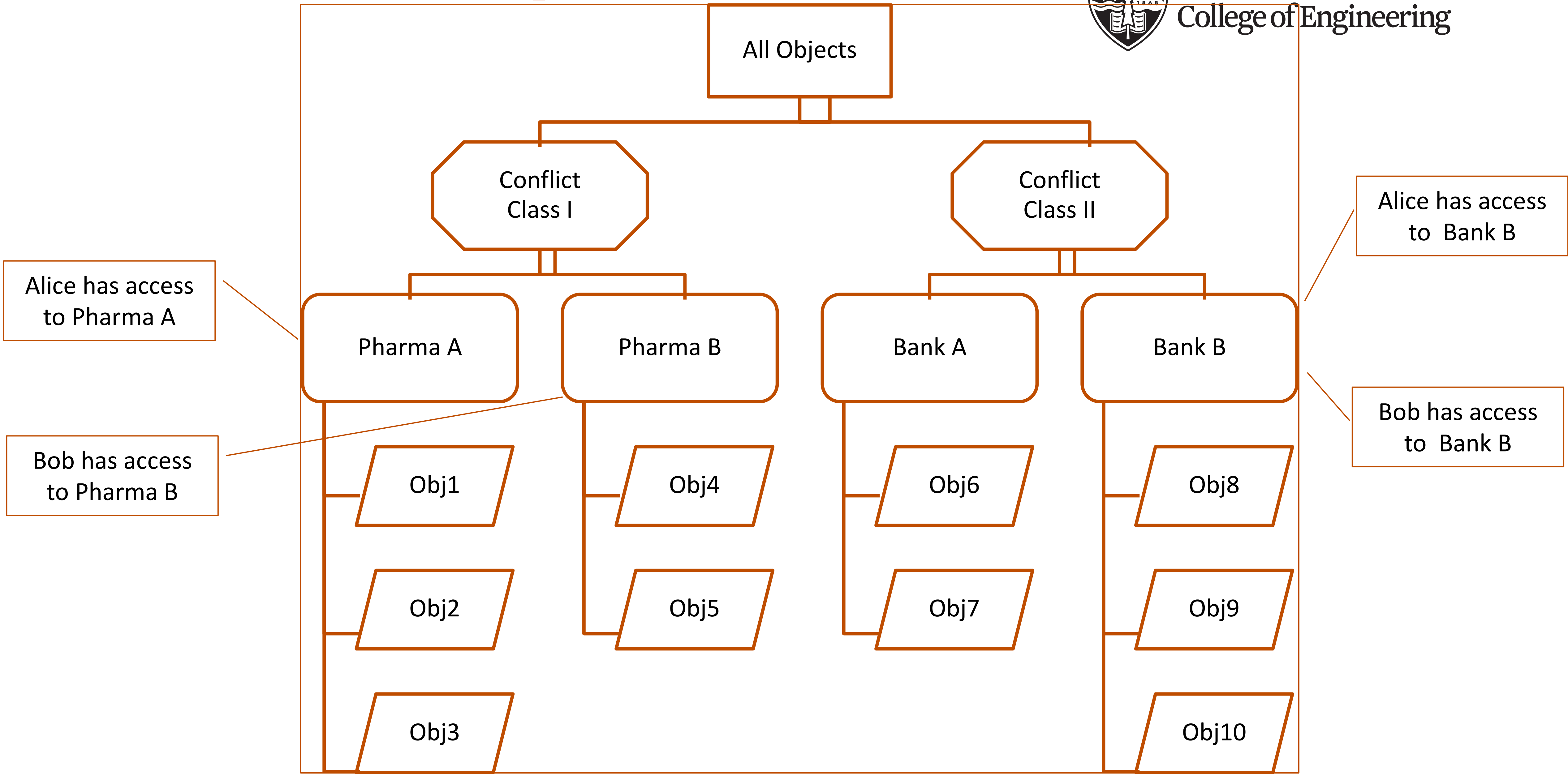
Oregon State University
College of Engineering

- Subject S can read an object O only if
  - O is in same DS as object already accessed by S or
  - O belongs to CI from which S has not yet accessed any information

*-Property Rule

- A subject S can write to object O only if
  - S can read O according to the simple security rule AND
  - All objects that S can read are in the same DS as O

Chinese Wall Example



Summary

Oregon State University
College of Engineering

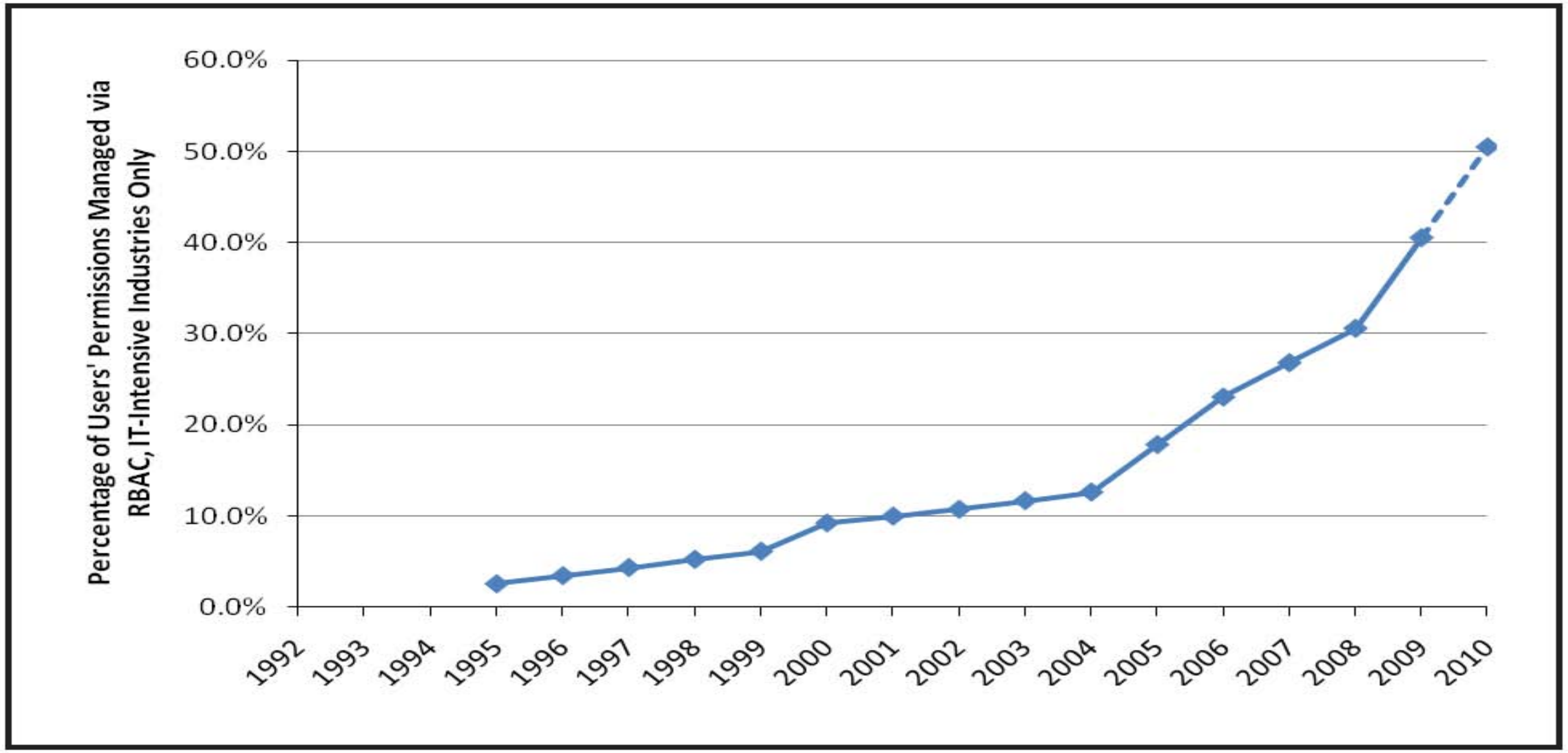
- Chinese Wall is a mandatory access control model that looks at multi-lateral security
  - focuses on conflict-of-interest
- It is used in law firms, consulting firms, accounting firms and in (semi) regulated industries like electricity generation and distribution

Role-Based Access Control (RBAC): Introduction

Role-Based Access Control (RBAC)

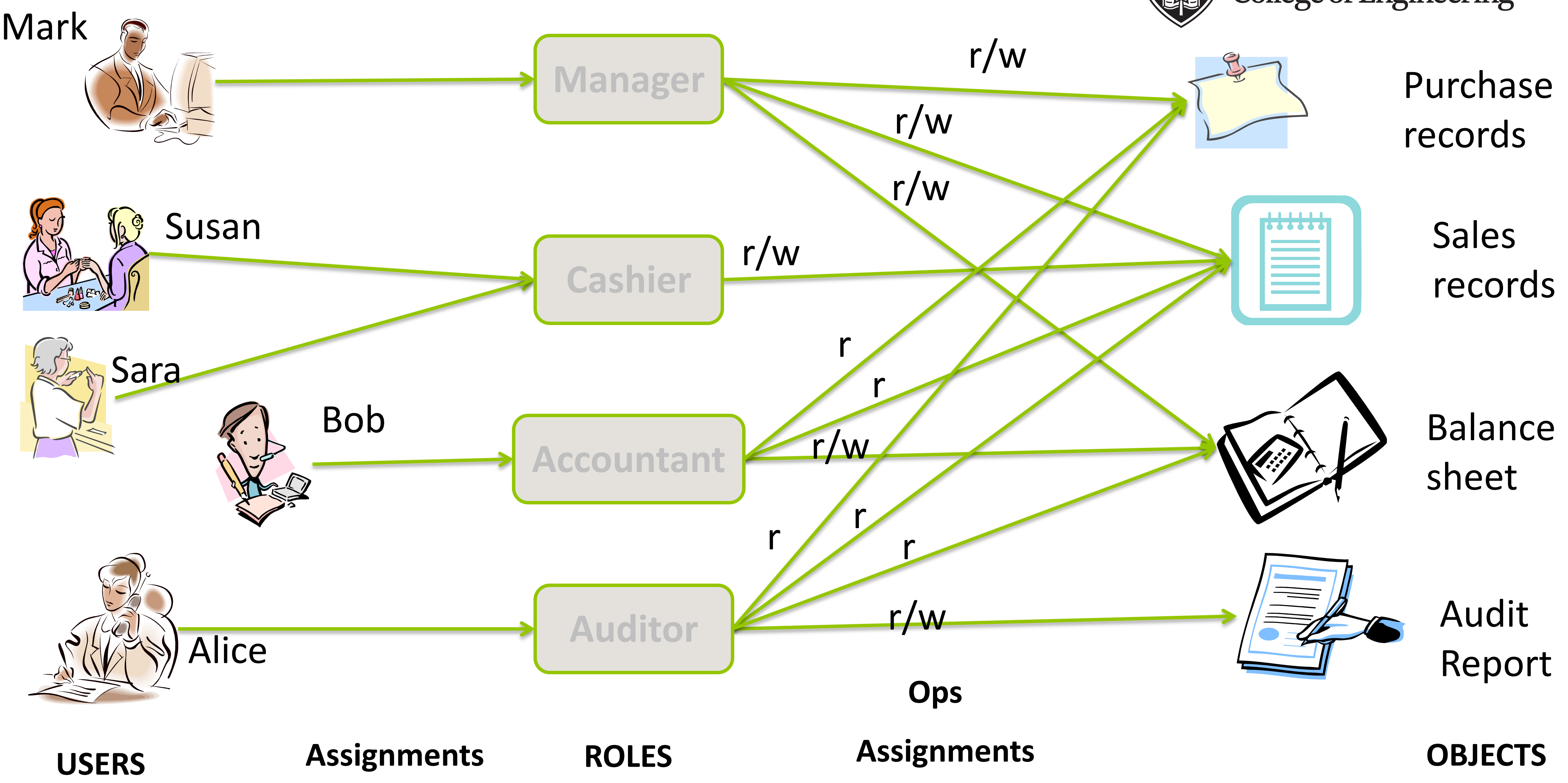
- RBAC is most widely used access control model in business world [NIST&RTI 2010]
  - introduced in early-nineties

Figure ES-2. RBAC Adoption, 1992–2010



A Mini Example of RBAC

Oregon State University
College of Engineering



Role-Based Access Control

- Each role is defined by a set of permissions
  - created for job functions in an org.
  - represents competency, authority and responsibility
  - can be granted new permissions for new apps.
  - permissions can be revoked from a role

- Each user is assigned with one or more roles
  - based on the user's responsibility and qualification
  - can be reassigned with different roles

- Why RBAC?
  - User-permission associations are transitory
  - Role-permission associations are stable

Access Matrix Representation

	R1	R2	R3
U_1	X	X	
U_2	X		X
U_3		X	X
U_4			X
U_5	X	X	
…			
U_n	X		X

User-to-Role Mapping

	Roles			Files		Processes		Disks	
	R1	R2	R3	F1	F2	P1	P2	D1	D2
R1	control	owner	owner control	Read*	Read owner	wakeup	wakeup	seek	owner
R2		control		Write*	execute			owner	Seek*
R3			control		write	stop			

Role-to-Permission Mapping

- Similar to DAC ACM
- Roles can be Objects

DAC vs RBAC

- DAC
  - Users, Groups → Permissions
- RBAC
  - Roles → Permissions; Users → Roles
- Difference between groups and roles?
  - Group: collection of users
  - Role: collection of permissions, and possibly other roles [S96]
- Difference between DAC and RBAC
  - Different perspectives:
    - RBAC is from perspective of organization
  - Different right management: [Ferraiolo&Kuhn1992]
    - DAC: allows a user to grant access to the objects he owns;
    - RBAC: typically users cannot pass their rights to others

Summary

- Role-based access control is introduced in early 90s to simplify permission management

- In RBAC
  - Roles are assigned permissions (role is a collection of permissions)
  - Users are assigned roles

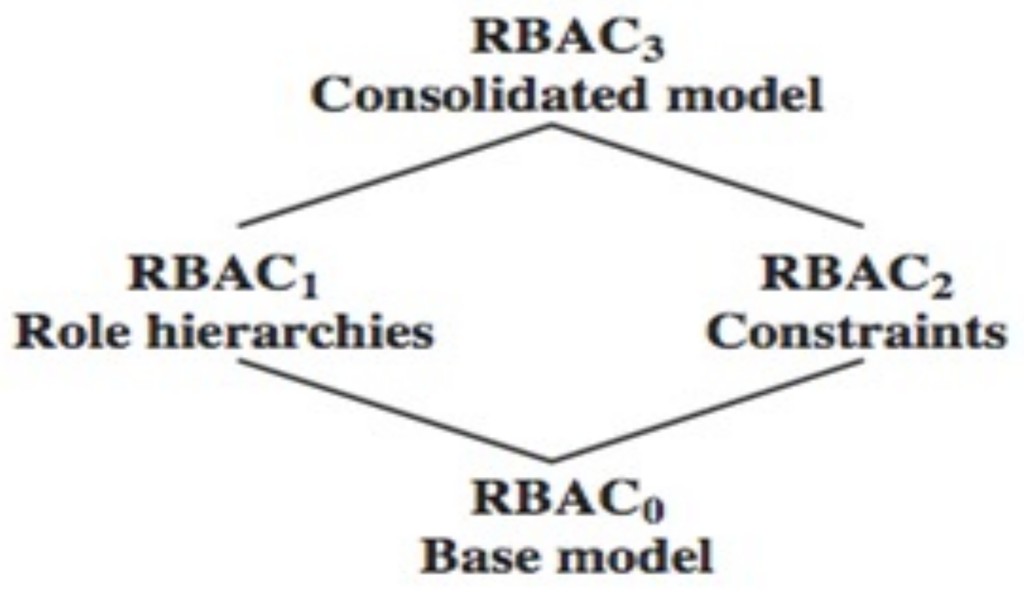
- Role-permission associations are relatively stable
  - Need to change less frequently

- User–role assignments change more often
  - Users change their role (promotions, lateral moves etc.)

RBAC Models

RBAC Models

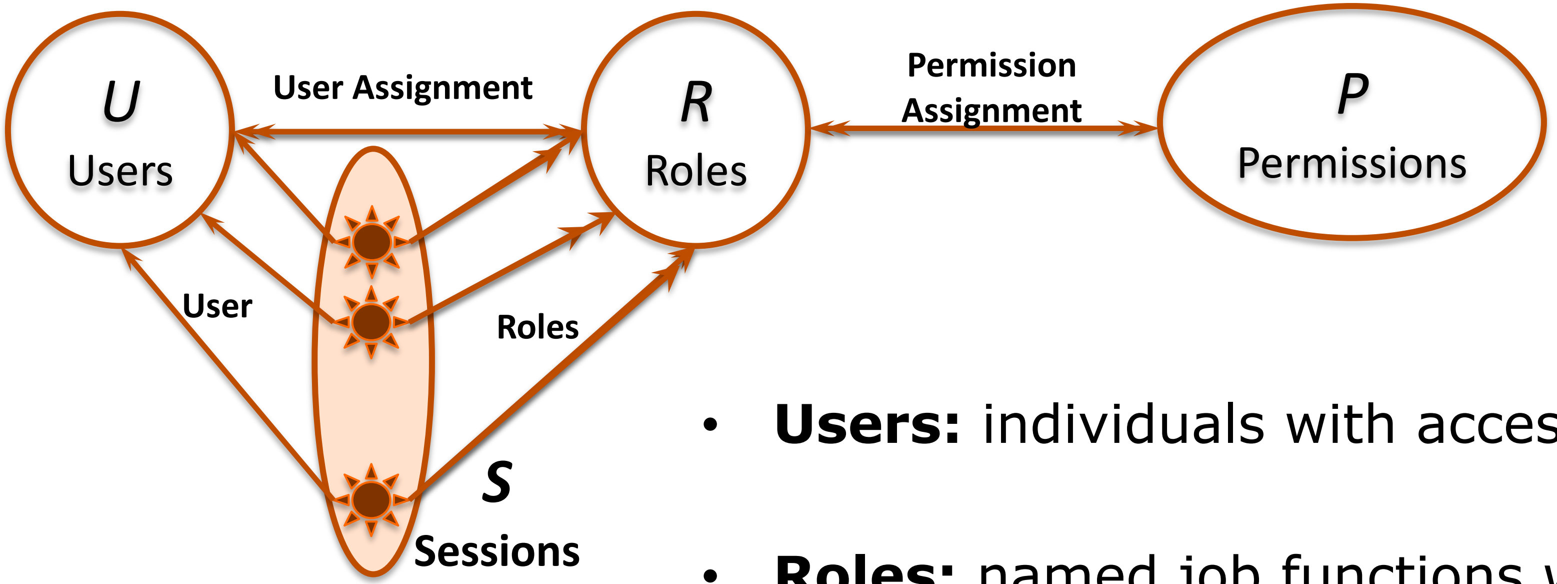
- $RBAC_0$
  - Minimum functionality
- $RBAC_1$
  - $RBAC_0$ + Role hierarchies
- $RBAC_2$
  - $RBAC_0$ + Constraints
- $RBAC_3$
  - $RBAC_0$ + $RBAC_1$ + $RBAC_2$



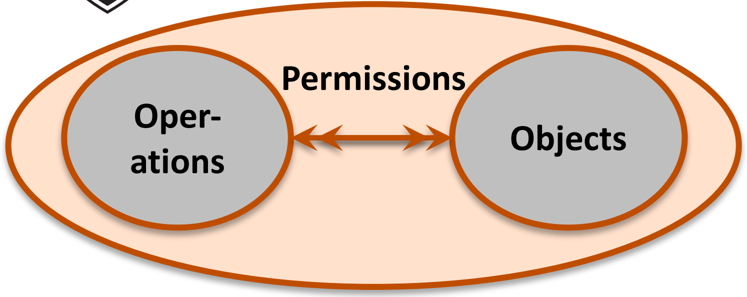
(a) Relationship among RBAC models

Figure Credit: Sandhu et al, Role-Based Access Control Models, IEEE Computer, Volume 29, Number 2, February 1996, pages 38-47.

$RBAC_0$ – Base



Oregon State University
College of Engineering



- **Users:** individuals with access to the system
- **Roles:** named job functions within the org
- **Permission:** approval to perform an *operation* on *object(s)* (a particular mode of access to objects)
  - Object: any resource
  - Operation: executable image of a program / action on object(s)
- **Session:** a mapping between a user and a subset of roles
  - allows selective activation and deactivation of roles assigned

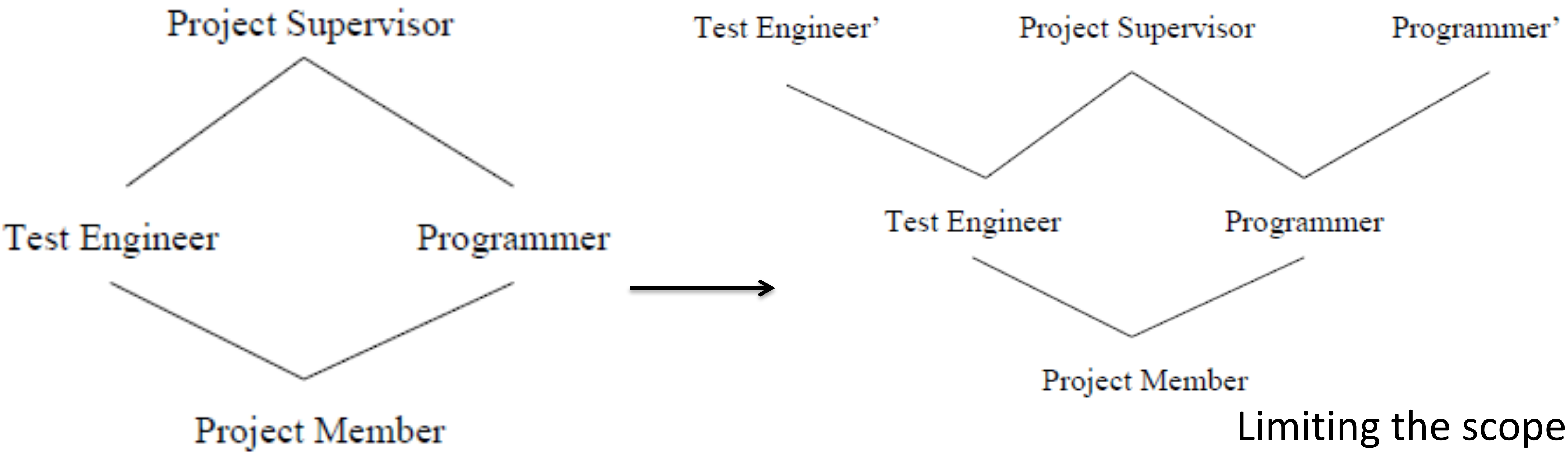
$RBAC_1$ – – $RBAC_0$ + Role Hierarchies

- Reflect hierarchical structure of roles in org
- Mathematically, partial order (reflexive, transitive, anti-symmetric)
- Help reduce permission management further

$RBAC_1$ – – $RBAC_0$ + Role Hierarchies

Oregon State University College of Engineering

Higher -> More rights, line from lower to higher means inheritance of rights



Example of Role Hierarchy

Limiting the scope of inheritance:
Role Hierarchy with *private roles*

Figure Credit: Sandhu et al, Role-Based Access Control Models, IEEE Computer, Volume 29, Number 2, February 1996, pages 38-47.

$RBAC_2$ – $RBAC_0$+Constraints

Oregon State University College of Engineering

- Constraints: Reflect higher-level organizational policy
- Example constraint types
  - **Mutually exclusive roles** (U → R and R → P)
    - User to only one role in set, permission to only one role
    - Implication – users with different roles have no shared permissions
    - E.g.: A user cannot be both the Accountant and Auditor
  - **Cardinality** – maximum number users assigned to role, maximum number of roles permitted a user (static or dynamic), maximum number of permissions to a role
    - E.g.: Role of CEO of a company can only have one user assigned to it

$RBAC_2$ – $RBAC_0$+Constraints

Oregon State University
College of Engineering

- Example constraint types
  - **Prerequisite** – can assign role only if already assigned prerequisite role
    - Idea is to support least privilege…if role R1 inherits from R2 and R3, then if only R2 or R3 rights are needed, those roles can be used
    - Remember, no hierarchies in $RBAC_2$

Static Separation of Duty (SSD)

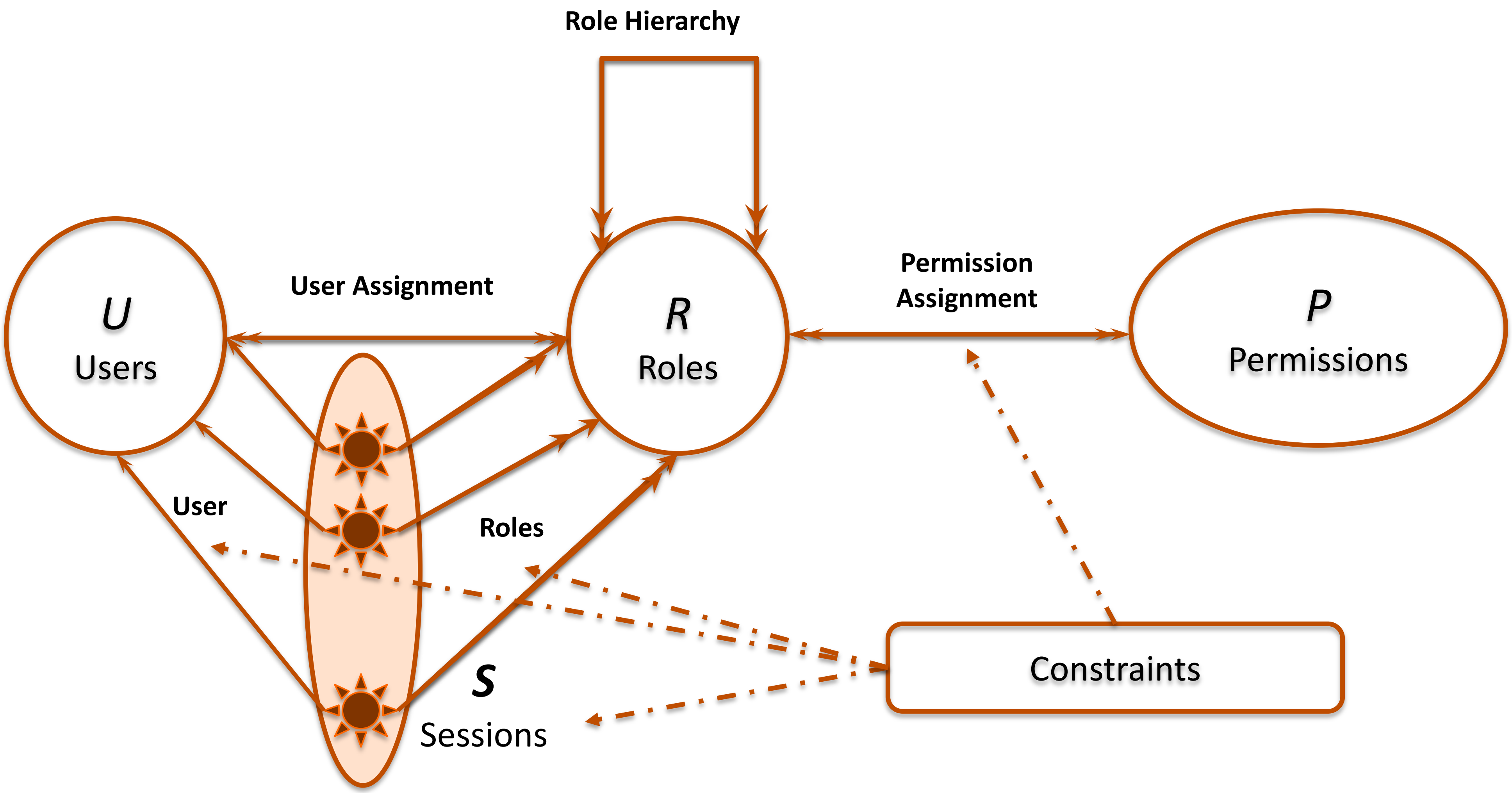
Oregon State University
College of Engineering

- Prevents conflict of interest
- Cardinality constraint on a set of roles
  - SSD := (*rs*, *n*) where no user is assigned to *n* or more roles from the *role set rs, i.e.*
- Mutual exclusive roles as a special case:
  - SSD := $(\{r_1, r_2\}, 2)$

Dynamic Separation of Duty (DSD)

- Similar to SSD, but activated within sessions
- Typically for temporal conflicts of interest
- Definition
  - DSD := (*role set*, *n*) ($n \geq 2$) no user session may activate $\geq n$ roles from *role set*
- Example: Author and PC member (conference)

RBAC$_3$ – Consolidated Model



RBAC and Security Principles

Oregon State University
College of Engineering

- RBAC supports well-known security principles:
  - Least Privilege
  - Separation of duties

- Least Privilege : RBAC can be configured so only those permissions required for a job function are assigned to a role representing that function.

- Separation of duties : by ensuring that mutually exclusive roles must be invoked to complete a sensitive task.

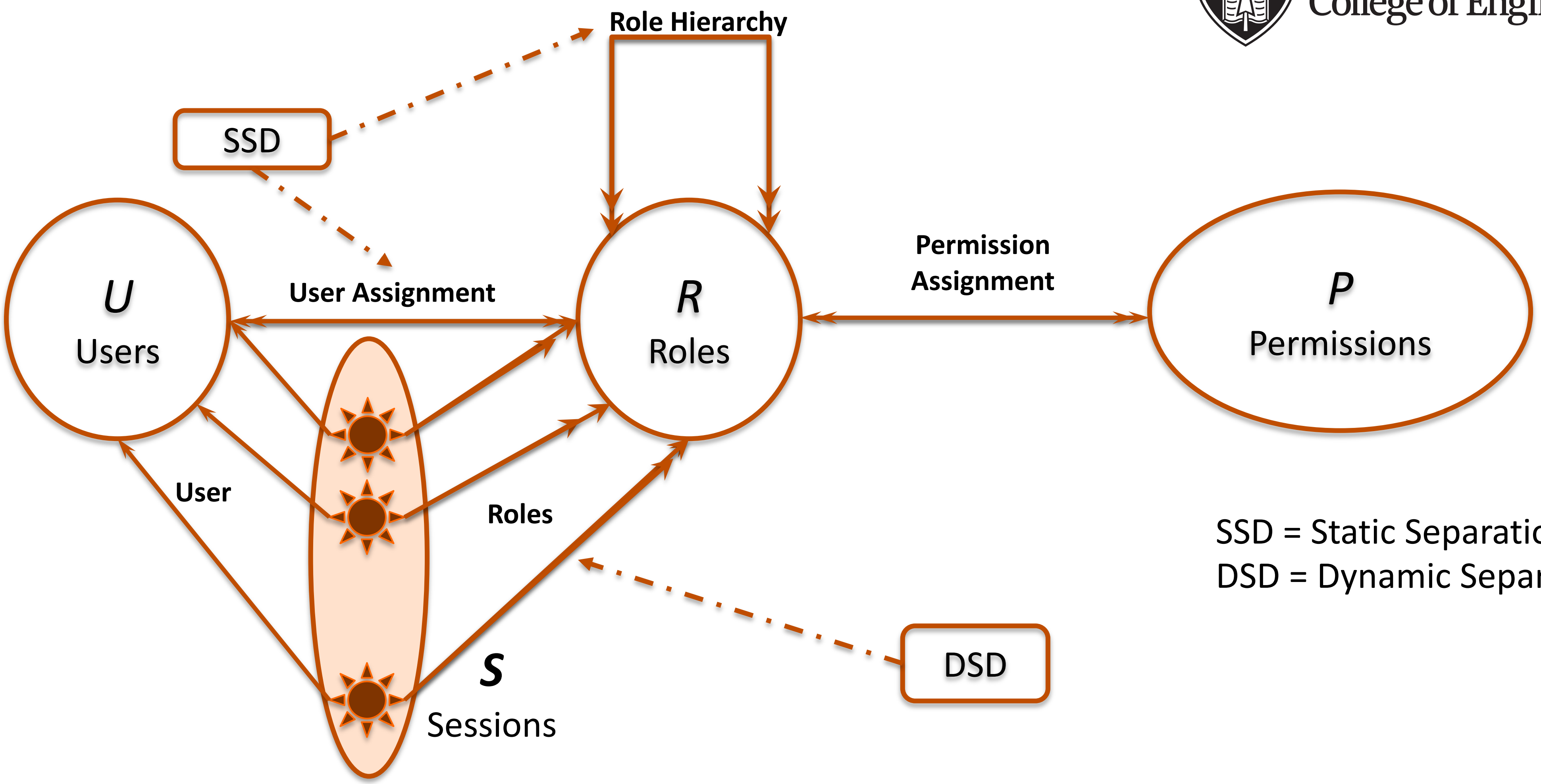
NIST RBAC Standard

- Standards
  - **ANSI INCITS 359-2012** replaces 359-2004.
  - **ANSI INCITS 494-2012** supplements 359-2012 with enhanced constraints, which can include attributes.
  - **ANSI INCITS 459-2011** provides guidance in RBAC system implementation and interoperability.

- RBAC Reference Models:
  - Core RBAC
  - Hierarchical RBAC
    - General Hierarchies
    - Limited Hierarchies
  - Constrained RBAC
    - RBAC with Static Separation of Duty Relations
      - With and Without Hierarchies
    - RBAC with Dynamic Separation of Duty Relations
    - **RBAC with general constraints on attributes of users/objects/environment** (**new** in **ANSI INCITS 359-2012**)

NIST Model

Oregon State University
College of Engineering



SSD = Static Separation of Duty
DSD = Dynamic Separation of Duty

NIST RBAC Model

Unspecified by NIST RBAC

- Scalability
- Authentication
- Negative permissions
- Nature of permissions
- Discretionary role activation
- Role engineering
- RBAC administration
- Role revocation

Summary

Oregon State University
College of Engineering

- Many variations of RBAC models exist
- First paper introduced 4 models
  - $RBAC_0$
  - $RBAC_1 = RBAC_0$ + Role Hierarchies
  - $RBAC_2 = RBAC_0$ + Constraints
  - $RBAC_3 = RBAC_0$ + Role Hierarchies + Constraints
- NIST Standardized variations of the original models
  - Core RBAC, Hierarchical RBAC and Constrained RBAC
- RBAC is suitable for supporting well-known security principles
  - Least Privilege, Separation-of-duty