INFS 767
Secure Electronic Commerce
Fall 1999

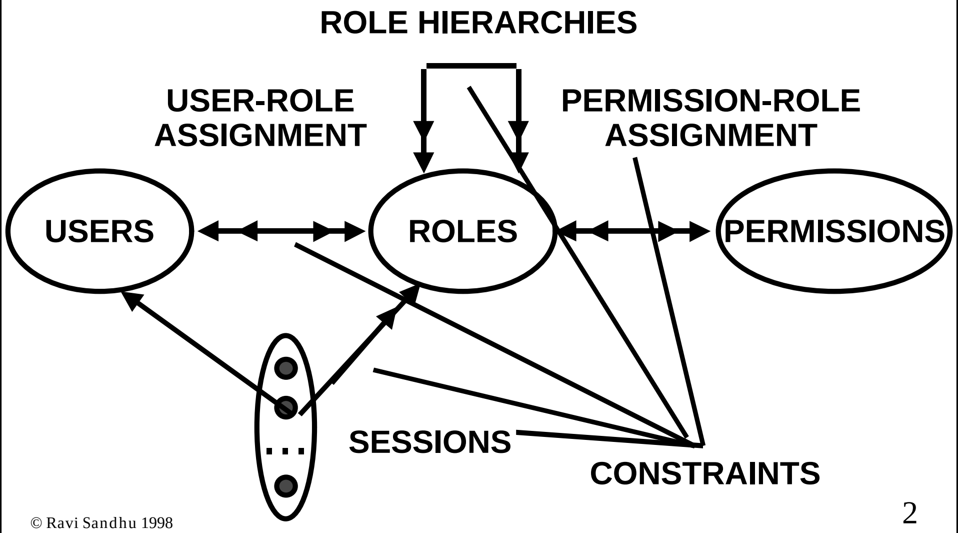
Lecture 5

DAC in RBAC

Role Hierarchies

Prof. Ravi Sandhu

RBAC96



2

DAC in RBAC

- RBAC is policy neutral and highly expressive
- Can RBAC enforce MAC and DAC policies?
- Simulation of MAC in RBAC is demonstrated earlier.
- Variety of DAC policies are simulated.

3

Owner Centric DAC

- The creator of an object becomes its owner.
- There is only one owner of an object
  - Ownership remain fixed.
  - It can be transferred to another user.
- Destruction of object can only be done by its owner.

4

Variations of DAC

- **Strict DAC**
- **Liberal DAC**

5

Strict DAC

- **Ownership cannot be transferred.**
- **Only owner has a discretionary authority to grant access to an object.**
- **Example:**
  - **Alice has created an object (he is owner) and grants access to Bob. Now Bob cannot grant propagate the access to another user.**

6

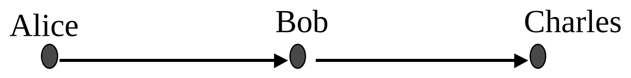
Liberal DAC

- **Owner can delegate discretionary authority for granting access to other users.**
  - **One Level grant**
  - **Two Level Grant**
  - **Multilevel Grant**

7

One Level Grant

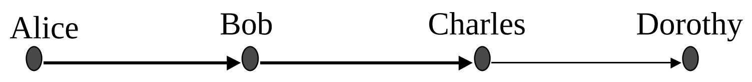
- **Owner can delegate authority to another user but they cannot further delegate this power.**



8

Two Level Grant

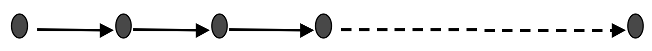
- In addition a one level grant the owner can allow some users to delegate grant authority to other users.



© Ravi Sandhu 1998 9

Multilevel DAC

- In this case the power to delegate the power to grant implies that this authority can itself be delegated.



© Ravi Sandhu 1998 10

DAC with change of Ownership

- **This variation allows a user to transfer ownership of an object to another user.**
- **Can be combined with strict or liberal DAC with all variations.**

11

Revocation

- **Grant-Independent Revocation.**
- **Grant-Dependent Revocation.**

12

Common Aspects

- Creation of an object in the system requires the simultaneous creation of
  - three administrative roles
    - OWN_O, PARENT_O, PARENTwithGRANT_O
  - One regular role
    - READ_O

$$\begin{tikzcd}
\text{OWN\_O} \arrow[r] & \text{PARENTwithGRANT\_O} \arrow[r] & \text{PARENT\_O} \arrow[r] & \text{READ\_O}
\end{tikzcd}$$

Administration of roles associated with object O

$$\begin{tikzcd}
\text{OWN\_O} \arrow[d, dash] \\
\text{PARENTwithGRANT\_O} \arrow[d, dash] \\
\text{PARENT\_O}
\end{tikzcd}$$

Administrative role hierarchy

Common Aspects II

- **We require simultaneous creation of Eight Permissions**
  - **canRead_O**
  - **destroyObjet_O**
  - **addReadUser_O, deleteReadUser_O**
  - **addParent_O, deleteParent_O**
  - **addParentWithGrant_O, deleteParentWithGrant_O**

Roles and associated Permissions

- **OWN_O**
  - **destroyObject_O, addParentWithGrant_O, deleteParentWithgrant_O**
- **PARENTwithGRANT_O**
  - **addParent_O, deleteParent_O**
- **PARENT_O**
  - **addReadUser_O, deleteReadUser_O**
- **READ_O**
  - **canRead_O**

Common Aspects III

- Destroying an object O requires deletion of four roles and eight permissions in addition of destroying the object O.

17

Strict DAC in RBAC96

- Cardinality constraints as:
  - Role OWN_O = 1
  - Role PARENTwithGARNT_O = 0
  - Role PARENT_O = 0

18

One level DAC in RBAC96

- Cardinality constraints as:
  - Role OWN_O = 1
  - Role PARENTwithGARNT_O = 0

Two Level DAC in RBAC96

- Cardinality constraints as:
  - Role OWN_O = 1

Multilevel DAC in RBAC96

- At the time of creation of an object the association of permissions to roles are redefined as:
- OWN_O
  - destroyObject_O
- PARENTwithGRANT_O
  - addParent_O, deleteParent_O, addParentWithGrant_O, deleteParentWithgrant_O
- PARENT_O and READ_O are same
- Cardinality constraints as of Two level DAC.

Multiple Ownership

- Can be accommodated by allowing users to be added to OWN_O
- All members of OWN_O has identical powers, including the ability to revoke other owners
- grant-independent revoke

Grant-Dependent Revoke

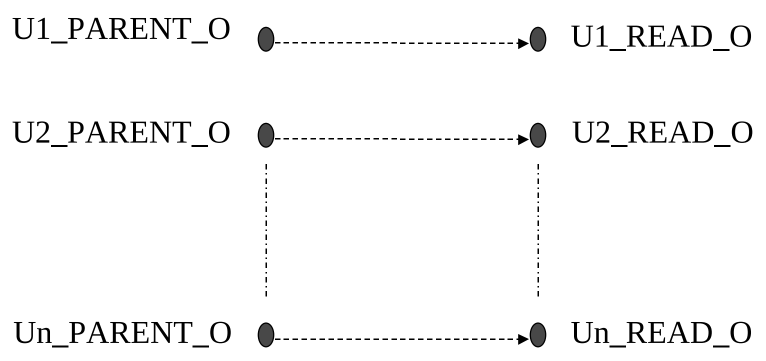
- **Only granter can revoke access**
- **When owner authorizes user U, the roles U_PARENT_O and U_READ_O and permissions addU_ReadUser_O and deleteU_ReadUser_O are automatically created.**
- **New permissions are assigned to U_PARENT_O role.**

Grant-Dependent Revoke cont..

- **canRead_O permission is assigned to U_READ_O role at the time of creation.**
- **Ui_PARENT_O manages the membership assignments to Ui_READ_O.**
- **Cardinality constraint of Ui_PARENT_O is 1 and its membership cannot be changed.**
- **Owner can revoke every thing, thus OWN_O is senior to all Ui_PARENT_O.**

Grant-Dependent Revoke cont..

- Similar revocation can be simulated with respect to PARENT_O and PARENTwithGRANT_O roles.



READ_O role associated with members of PARENT_O

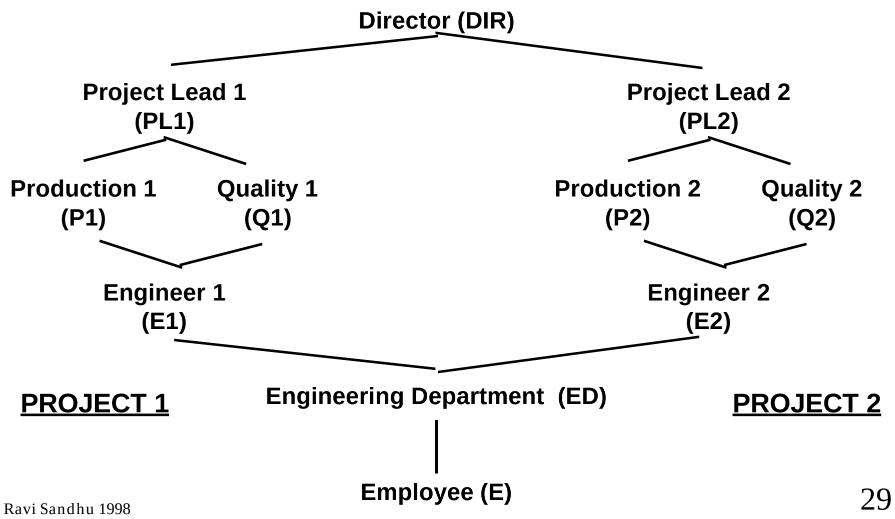
DAC in RBAC

- Many other variations can be simulated in the similar way.
- Informal constructions can be formalized using RBAC96.
- Objects can be grouped together to overcome the high number of roles.
- Results confirm that RBAC is policy neutral and can accommodate DAC and MAC.

ROLE HIERARCHIES

- Inheritance hierarchies
  - permission inheritance
  - user inheritance
- Activation hierarchies
  - role membership versus role activation

EXAMPLE ROLE HIERARCHY INTERPRETATIONS



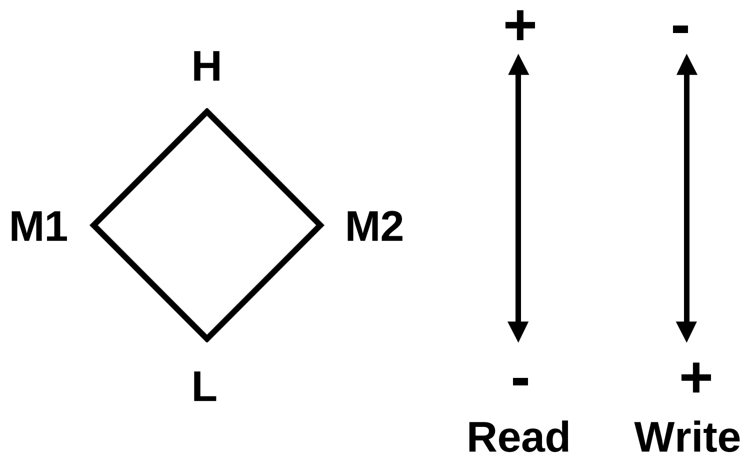
© Ravi Sandhu 1998 29

ALTERNATIVES

- separate inheritance and activation hierarchies
  - this paper
- single inheritance and activation hierarchy
  - most common approach, including RBAC96
- activation hierarchy only, no inheritance
  - alternative identified in NIST RBAC model
- inheritance hierarchy only, no activation hierarchy
  - does not seem to be useful

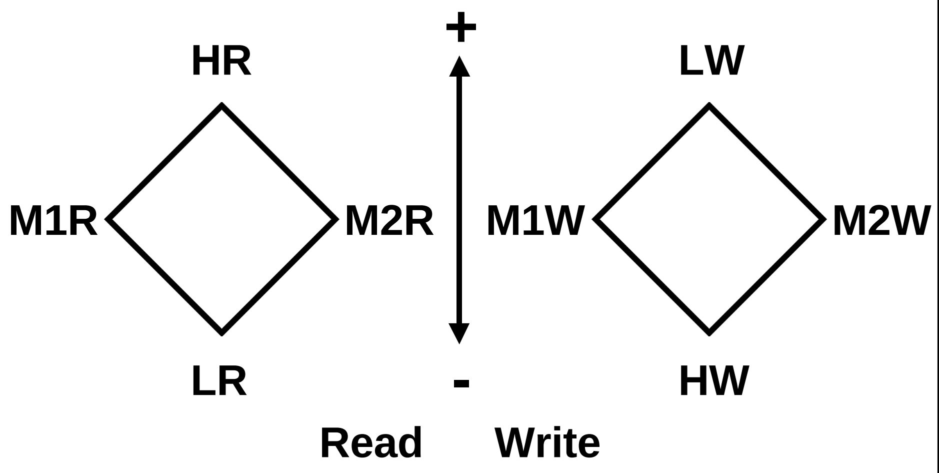
© Ravi Sandhu 1998 30

LBAC: LIBERAL *-PROPERTY

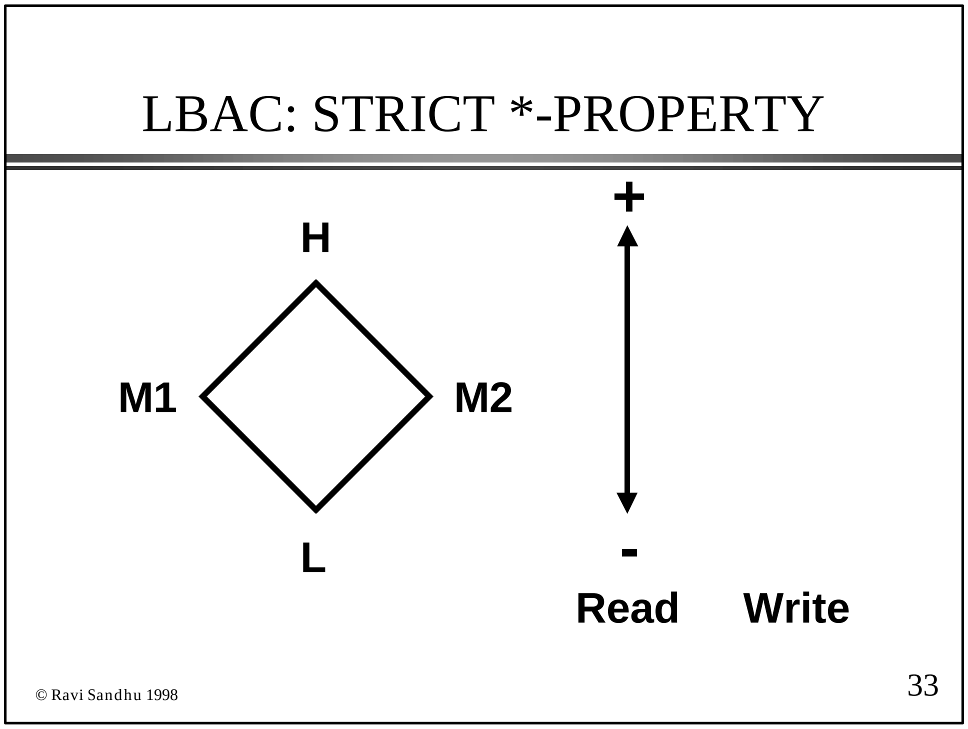


31

LBAC: LIBERAL *-PROPERTY DUAL ROLE SIMULATION

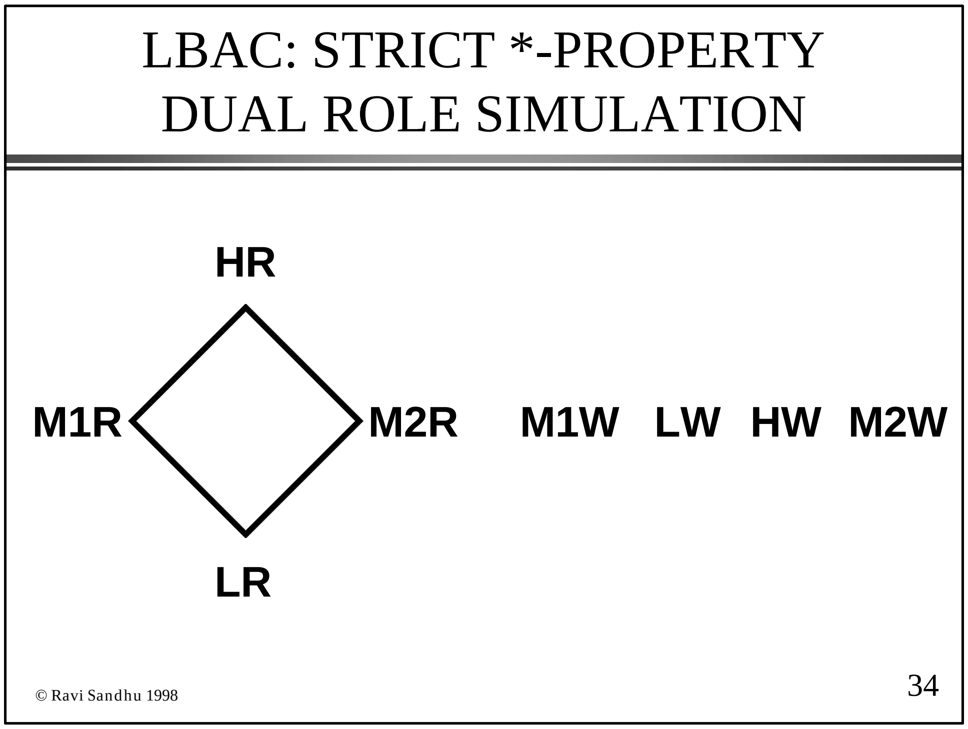


32



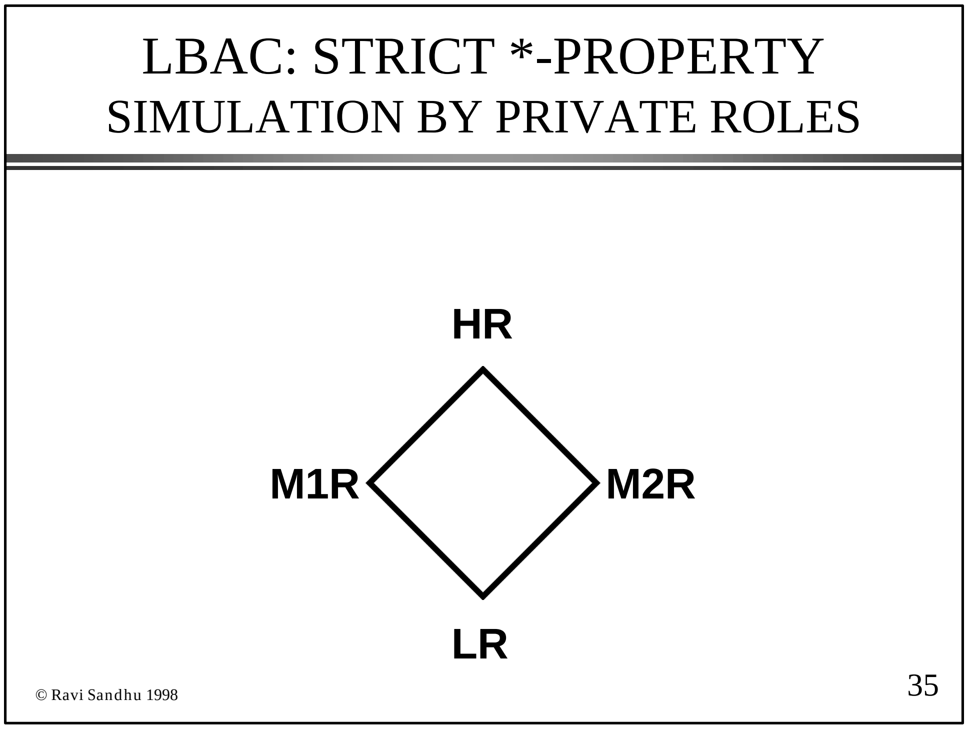
LBAC: STRICT *-PROPERTY

33



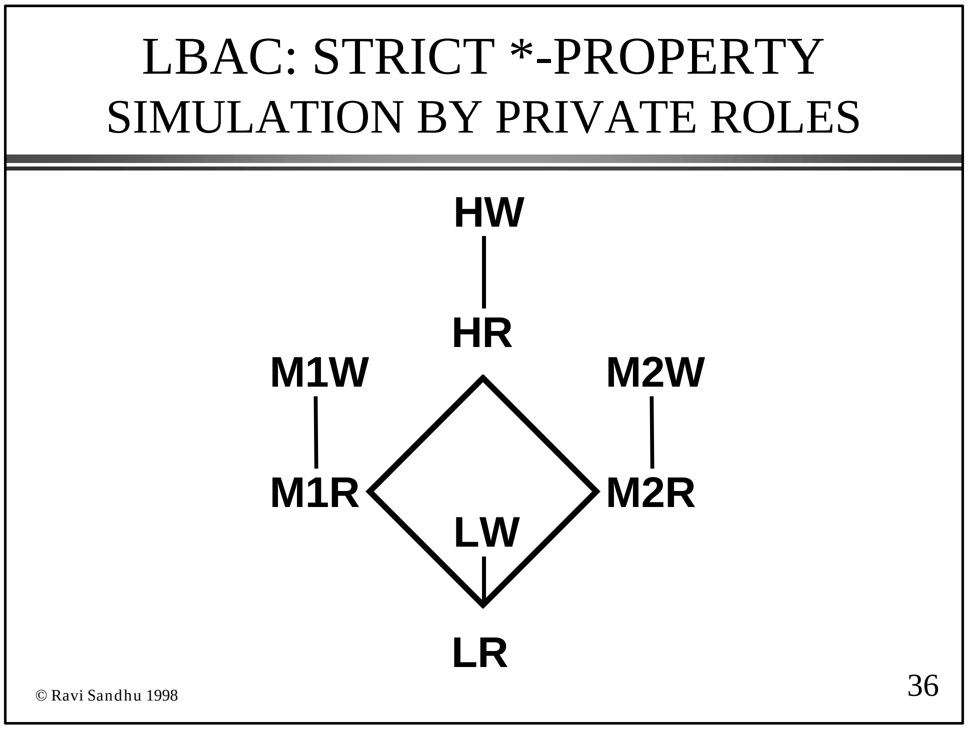
LBAC: STRICT *-PROPERTY DUAL ROLE SIMULATION

34



LBAC: STRICT *-PROPERTY SIMULATION BY PRIVATE ROLES

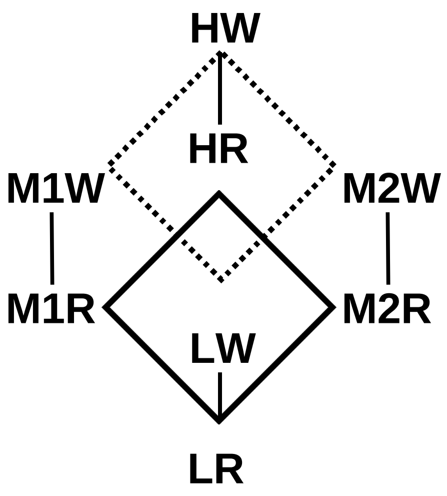
$$\begin{tikzcd}
 & HR \arrow[dl, dash] \arrow[dr, dash] & \\
M1R \arrow[dr, dash] & & M2R \arrow[dl, dash] \\
 & LR &
\end{tikzcd}$$



LBAC: STRICT *-PROPERTY SIMULATION BY PRIVATE ROLES

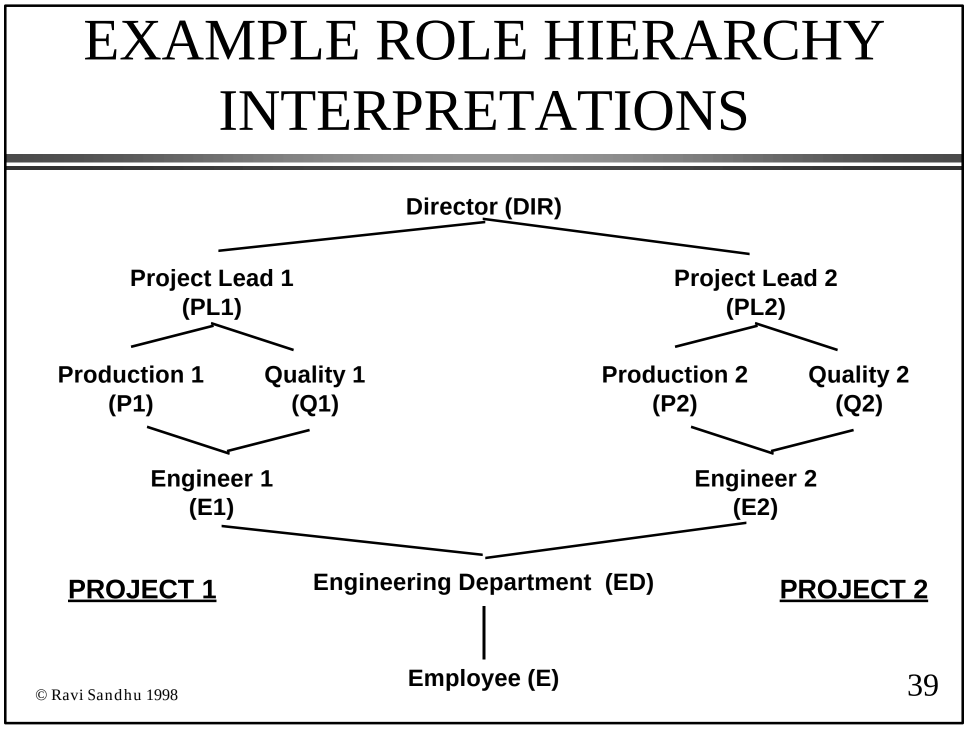
$$\begin{tikzcd}
 & HW \arrow[d, dash] & \\
M1W \arrow[d, dash] & HR \arrow[dl, dash] \arrow[dr, dash] & M2W \arrow[d, dash] \\
M1R \arrow[dr, dash] & LW \arrow[d, dash] & M2R \arrow[dl, dash] \\
 & LR &
\end{tikzcd}$$

LBAC: STRICT *-PROPERTY SIMULATION BY PRIVATE ROLES

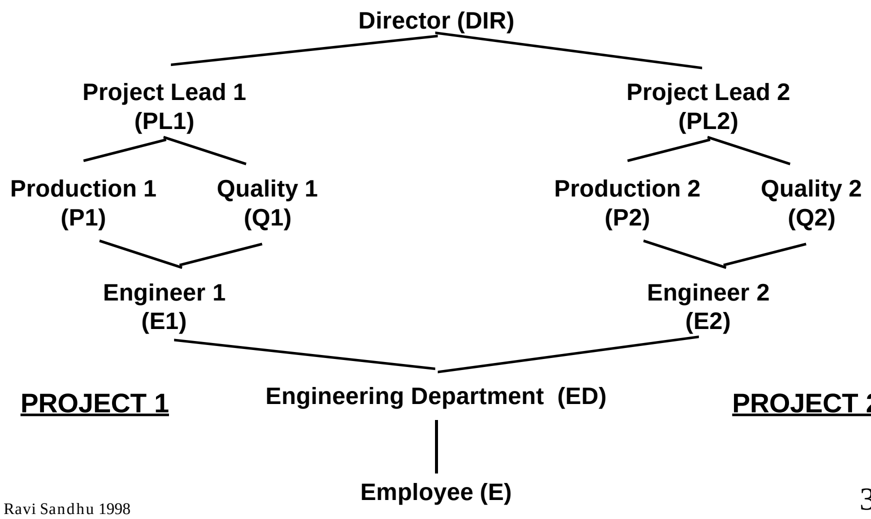


DYNAMIC SEPARATION OF DUTIES

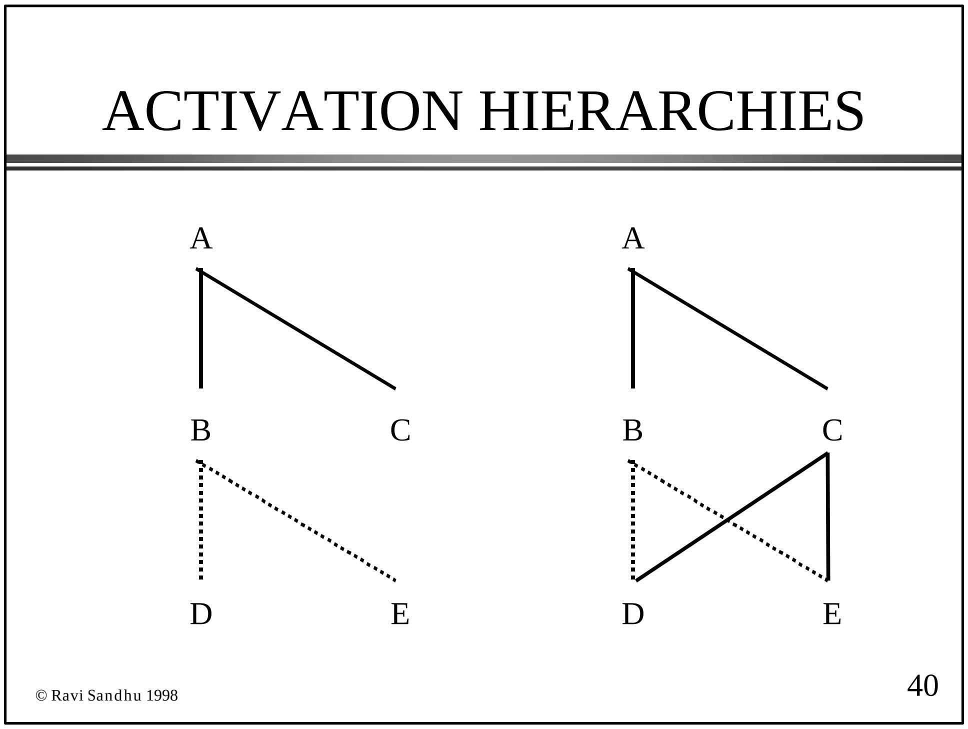
- **Roles in dynamic SOD**
  - **cannot have common seniors in role inheritance hierarchy, but**
  - **can have common seniors in role activation hierarchy**



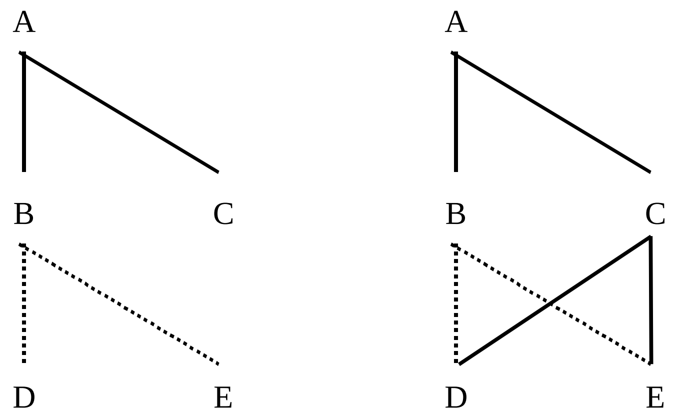
EXAMPLE ROLE HIERARCHY INTERPRETATIONS



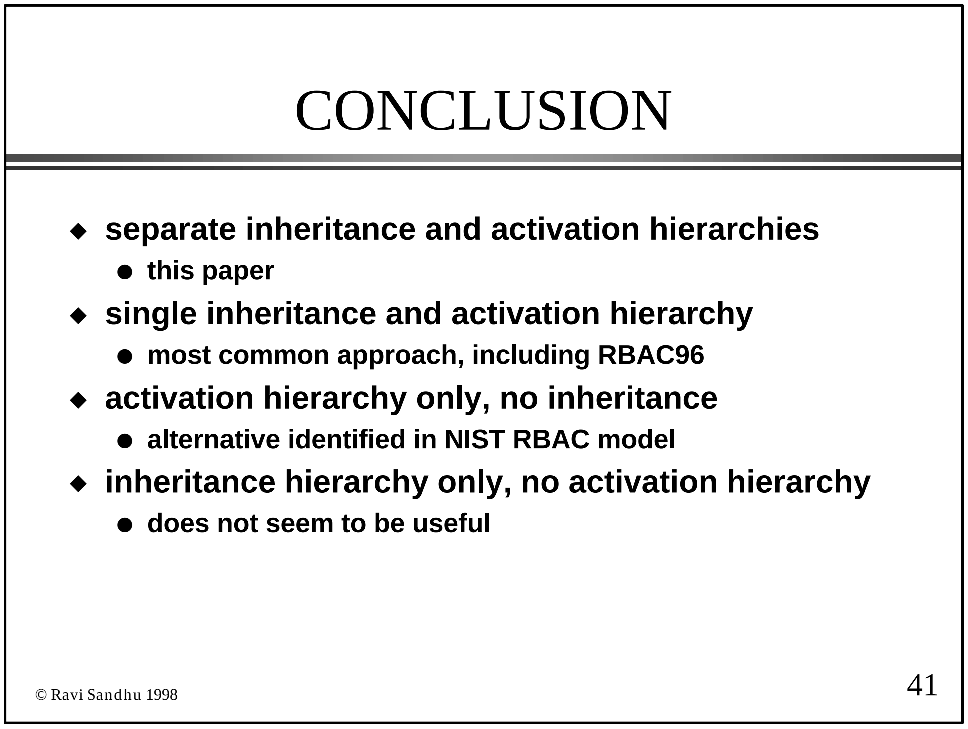
39



ACTIVATION HIERARCHIES



40



CONCLUSION

- separate inheritance and activation hierarchies
  - this paper
- single inheritance and activation hierarchy
  - most common approach, including RBAC96
- activation hierarchy only, no inheritance
  - alternative identified in NIST RBAC model
- inheritance hierarchy only, no activation hierarchy
  - does not seem to be useful

41