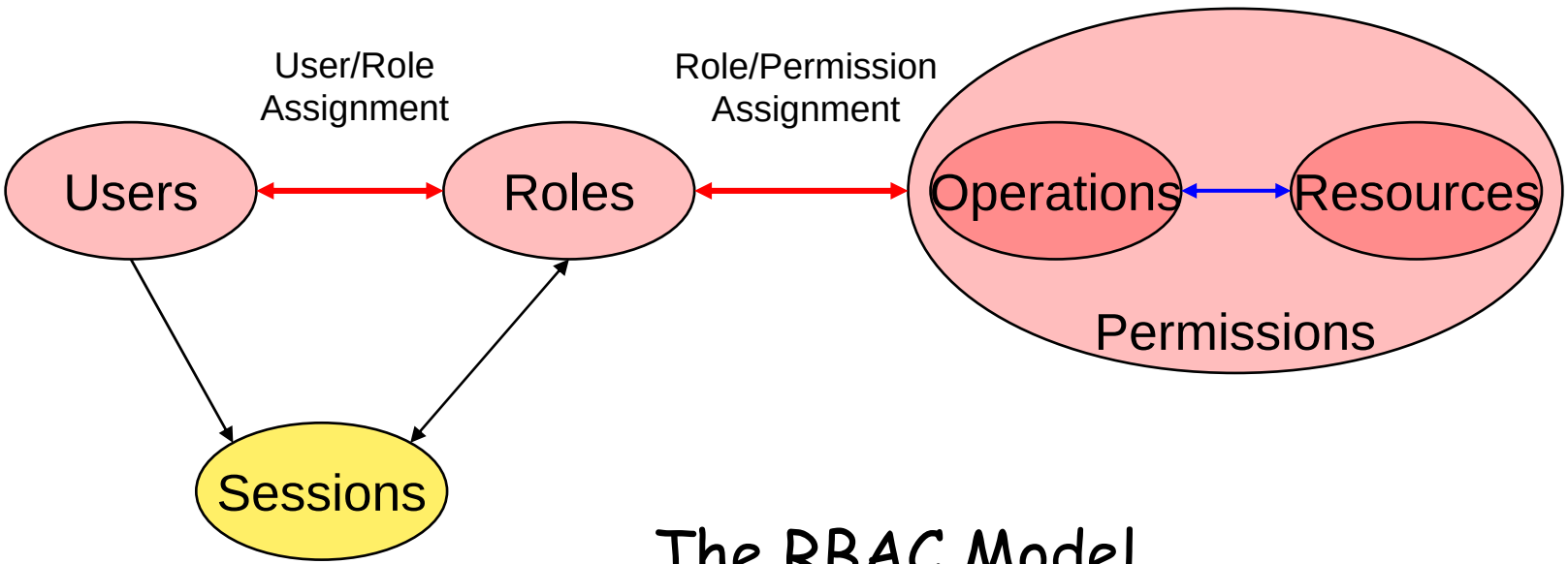
Some Usability Considerations in Access Control Systems

- A Position Paper -

Elisa Bertino, Seraphin Calo, Hong Chen, Ninghui Li, Tiancheng Li, Jorge Lobo, Ian Molly, Qhiua Wang

Grouping in Access Control

- Emerges naturally in access control management
- Basis for the Role-based Access Control (RBAC) Model



The RBAC Model

Managing Roles

- In a midsize enterprise with a few thousands employees we can easily find hundreds of roles and resources
- Large enterprises will have thousands of roles and resources
- Having roles simplifies management and improves security [IBM] but there is a very high upfront role engineering cost for the enterprise [NIST]

Approaches to creating RABC systems

- Top-down:
  - people perform a detailed analysis of business processes and derive roles from such analysis
- Bottom-up (role mining):
  - Use data mining techniques to discover roles from existing system configuration data
  - Practical value is controversial – how to mine roles with real-world meanings?

The value of meaningful roles

- System managers need to add, remove and modify users, roles and resources on regular basis
- Not having semantic meaning makes the management task very hard
- Optimizing a snapshot of an RBAC state might not be the appropriate solution for the lifetime of the RBAC system

Limitations of top-down approaches

- Expensive:
  - Time consuming
  - Requires expertise
- Companies may not have the expertise:
  - Elicitation of company know-how is not trivial: external experts have limited knowledge of the company business
  - Companies might consider the information confidential

Building Good RBAC Systems

- There is no standard or accepted metrics to evaluate the goodness of an RBAC system with respect management and usability
- Organizations are having difficulties in designing efficient and easy-to-manage RBAC systems by themselves using top-down approaches
- Automatic tools that can help with RBAC system design and maintenance have great commercial value as more and more companies are considering RBAC implementations

Building Good RBAC Systems

- Bottom-up approaches need to make good use of all available information
- Incorporate top-down techniques
- RBAC systems are not static systems. Once an RBAC system is built and put into use, we will need to maintain it. Overtime, an RBAC system is updated to meet the changes on access needs in an organization.

Role Mining Roadmap

	Low Complexity	*Good Semantics*	*Parameterized Roles*	*Least Privilege*	*Detect Outliers*
User Permission Only	✓	`Limited`			`Limited`
With User-Attribute	✓	✓			✓
With Permission-Parameter	✓	✓	✓		✓
With Update Log	✓	✓			✓
With Usage Log	✓	✓		✓	✓

Managing Evolving Systemes

- In an already existing and evolving RBAC systems one needs to deal with:
  - Unnecessary or missing user-to-role assignments
  - Unnecessary or missing role-to-permission assignment
  - And the general proliferation of roles

Why Unnecessary/Missing Assignment?

- Initial design
  - Imprecise information
  - Permissive design
- Legacy assignment
  - Job position changed
  - Task requirement changed
  - Project finished
- Problems
  - Management cost
  - Security concerns
- Role proliferation has similar roots

Dynamic Tools

- Given a messy RBAC state resulted from a long time of usage, the administrator is unlikely to completely reconfigure the RBAC system running a role mining tool
- It will be useful to develop techniques that do three things:
  1. Given an RBAC state, come up with an optimization that updates the RBAC state in some "localized way"
  2. Given an RBAC state and a update request come up with a suggested update to the RBAC system so that the accumulated results of multiple updates will not lead to a messy state that is difficult to "understand" and manage
  3. Provide "views" of the current state of the system and the resulting state after the update

Dynamic tools

- It will be useful to develop more "holistic" techniques:
  1. keep logs on how users use their permission and develop tools in which decisions and recommendations can be justified by usage
  2. Permissions can be removed from permission-to-role assignments if the permissions have been never exercised by user in that role
  3. Roles can be merged with other roles whose permissions are often used in tandem
  4. Logs can be used to clean-up roles that have been not activated for a long time
  5. Administrators may be will to accept some controlled inaccuracies (i.e. quantiafible risk)

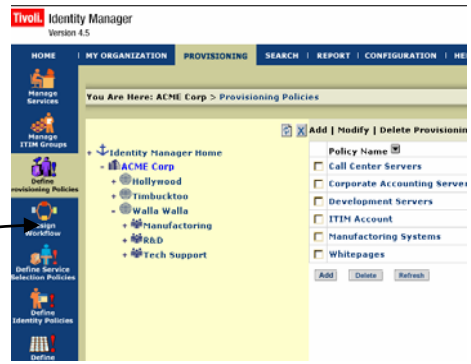
ITIM Admin Interface

- Interface requires navigation through multiple levels to get to the appropriate data for even a single role

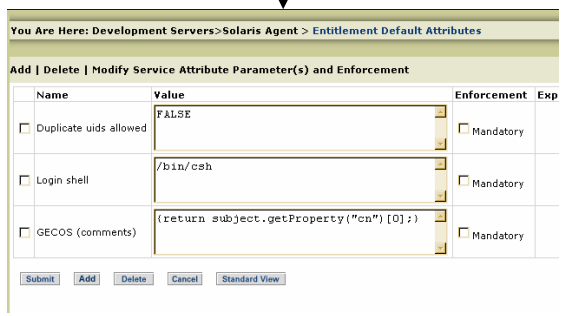
1) Identify Role

2) Identify Membership

3) Determine associated policies

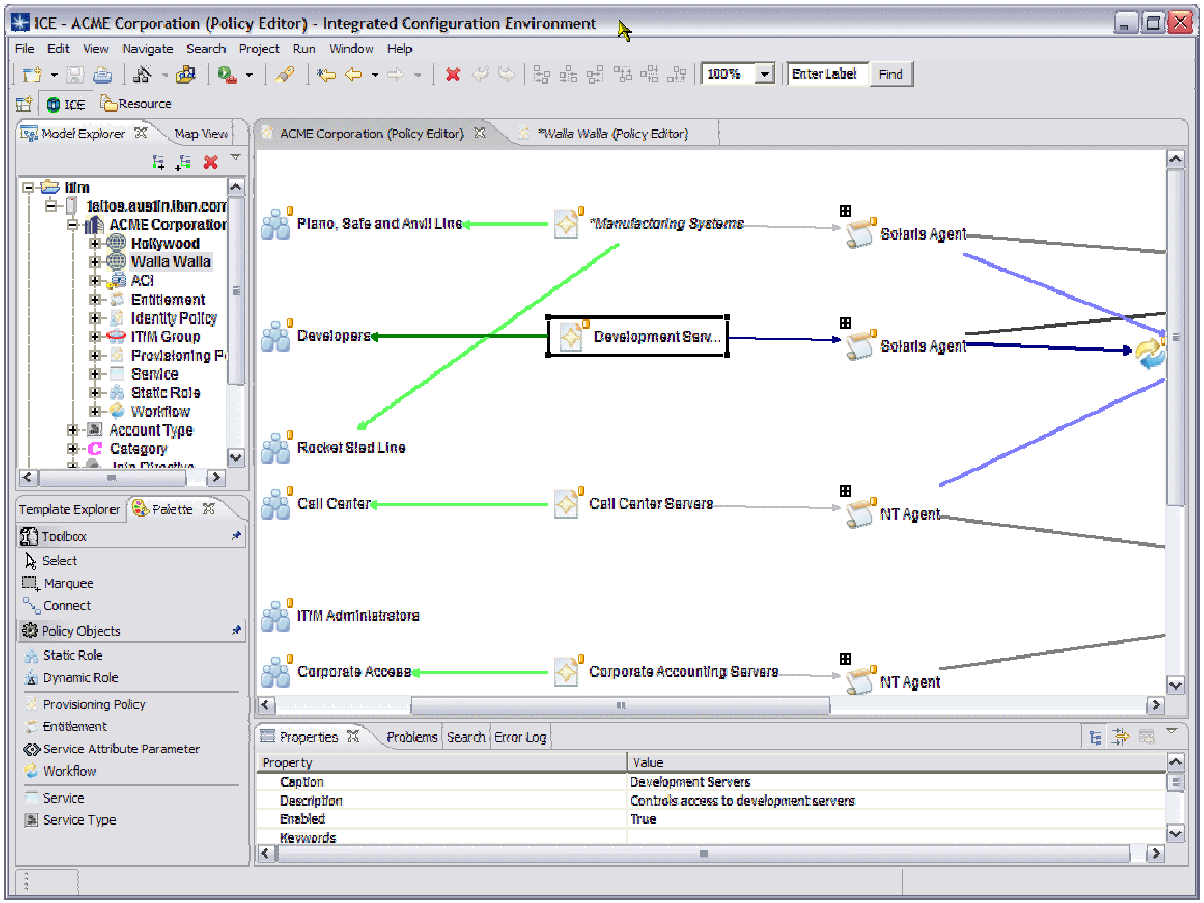


4) Examine specific policies



ITIM Graphical Interface

- Show graphs of relationships



Current Support Tools under Evaluation

- Reduce ITIM configuration complexity
  - Remove redundant or unnecessary provisioning policies
  - Merge policies with the same members or entitlements
  - Remove unnecessary role or entitlement assignment
- Assist in entitlement provisioning
  - Provision entitlements to roles
  - Provision entitlements to people
  - Facilitate policy/role reuse
- Suggest role assignments
  - Recommend a list of roles for a person based on her attributes
    - E.g. all software engineers are assigned to *Role ABC*
    - E.g. 85% of the supplement employees in Dept XYZ are assigned to *Role EFG*
  - Reduces selection scope
  - Lower likelihood of missing something

The End