Engineering Trust Management into Software

Mark Reith (mreith@cs.utsa.edu)
Jianwei Niu (niu@cs.utsa.edu)
William H. Winsborough (wwinsborough@acm.org)

University of Texas at San Antonio
20 May 07

Motivation

- Trust Management (TM) – framework for describing and reasoning about delegated access control policies
  - Security policy – a set of logic statements describing delegation of access control
  - Decentralized & dynamic
  - Non-functional requirement

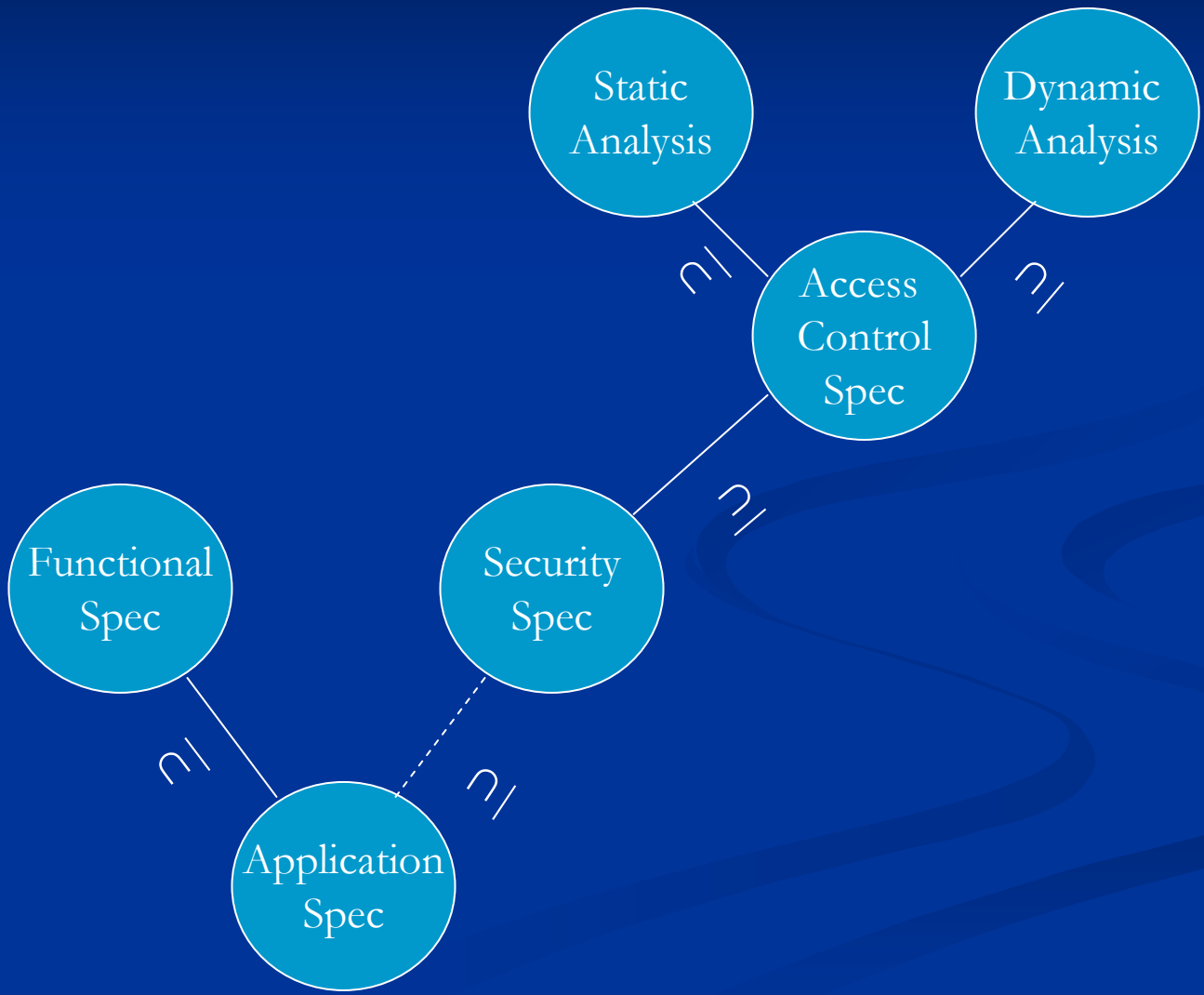
Motivation

- How to incorporate TM-type of access control into software and verify that security requirements are satisfied?
  - Security requirement - a set of invariant access control properties that must hold regardless of how the policy changes
- Use models

An Approach

- Separate requirements into categories based on specifications to be evaluated
- Develop models for each category of requirements
- Verify specifications hold per category
- Compose models to obtain model of application that meets all requirements
- Verify specifications of composed model (or slices of it)

Categorizing Requirements



Example

- An example from the real estate domain
  - Proposed software that assists in tracking and auditing real estate contracts and transactions
  - Manages files containing sensitive information on a realtor's listings and purchase agreements
  - Users (realtors) are independent contractors and thus no centralized access control authority exists

Users

- Licensed Realtors
  - Represents seller/buyer of property in negotiations
  - May have assistants who are also licensed realtors
- Board of Realtors (BOR)
  - Membership drawn from realtors
  - Oversees transactions comply with law
    - May audit transactions of a realtor
    - Audit team composed of other realtors

Approach

- Delegation could be a useful technique to:
  - Permit access to assistants
  - Permit access to auditors

Security Req's

- Assistants always need access to the files they have been granted by realtor (Availability)
- Non-realtors are not allowed access (Safety)
- A realtor is not allowed to be a auditor of his/her own transaction (Separation of Duty)
- A realtor is not allowed to be a future opposing representative on any listing or buyer's contract that they reviewed as an auditor (Chinese Wall)

Requirements

- Functional Req's
  - View/edit real estate documents, submit an offer, etc
  - Described as application behaviors
  - Modeled using state transition diagrams, etc
- Non-functional Req's: Security: Access Control
  - Described as constraints on application behaviors
  - Model the logic that determines when a constraint inhibits behaviors

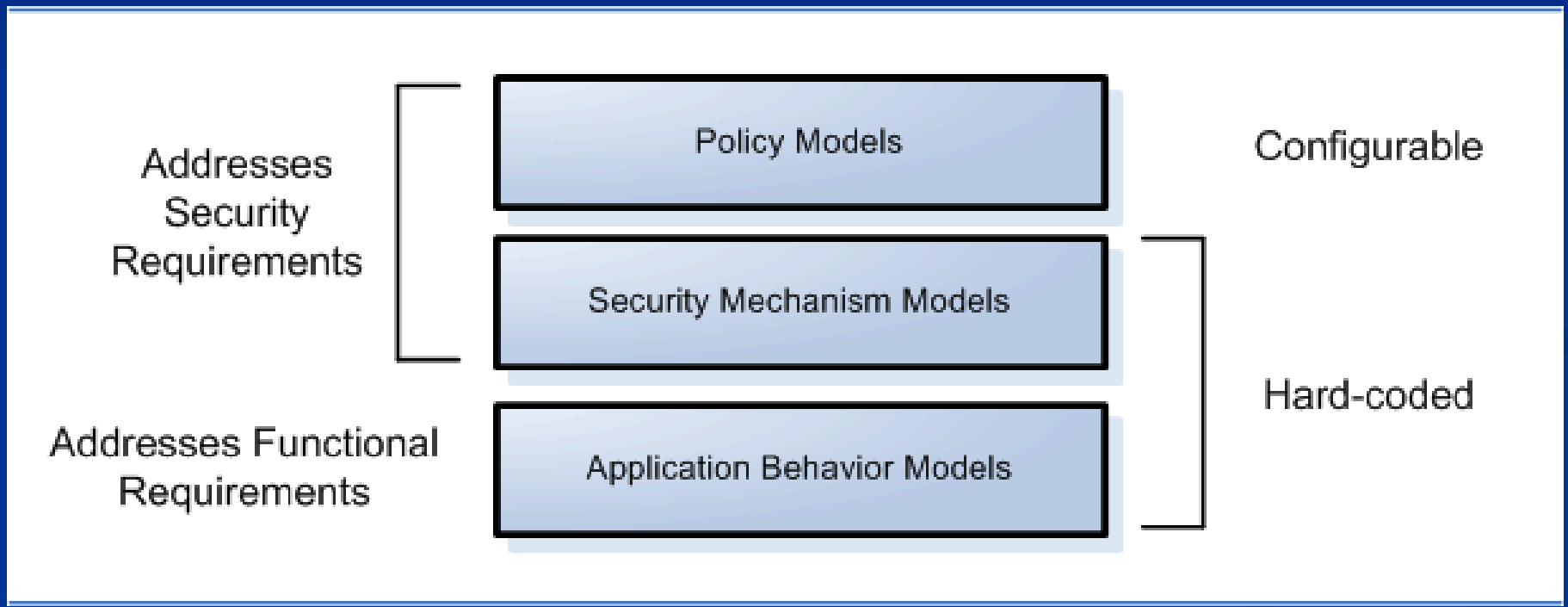
Access Control Req's

- Tools exist to statically analyze Availability and Safety properties solely from policy
- Separation of Duty requires current state information (is realtor on contract to be audited)
- Chinese Wall requires past state information (history)

Separation of Concerns

- Observe that policies (policy languages) alone may not satisfy all security requirements
- Need a means to describe mechanisms that satisfy remaining security requirements
- Need a means of describing how mechanisms are deployed to constrain application behaviors

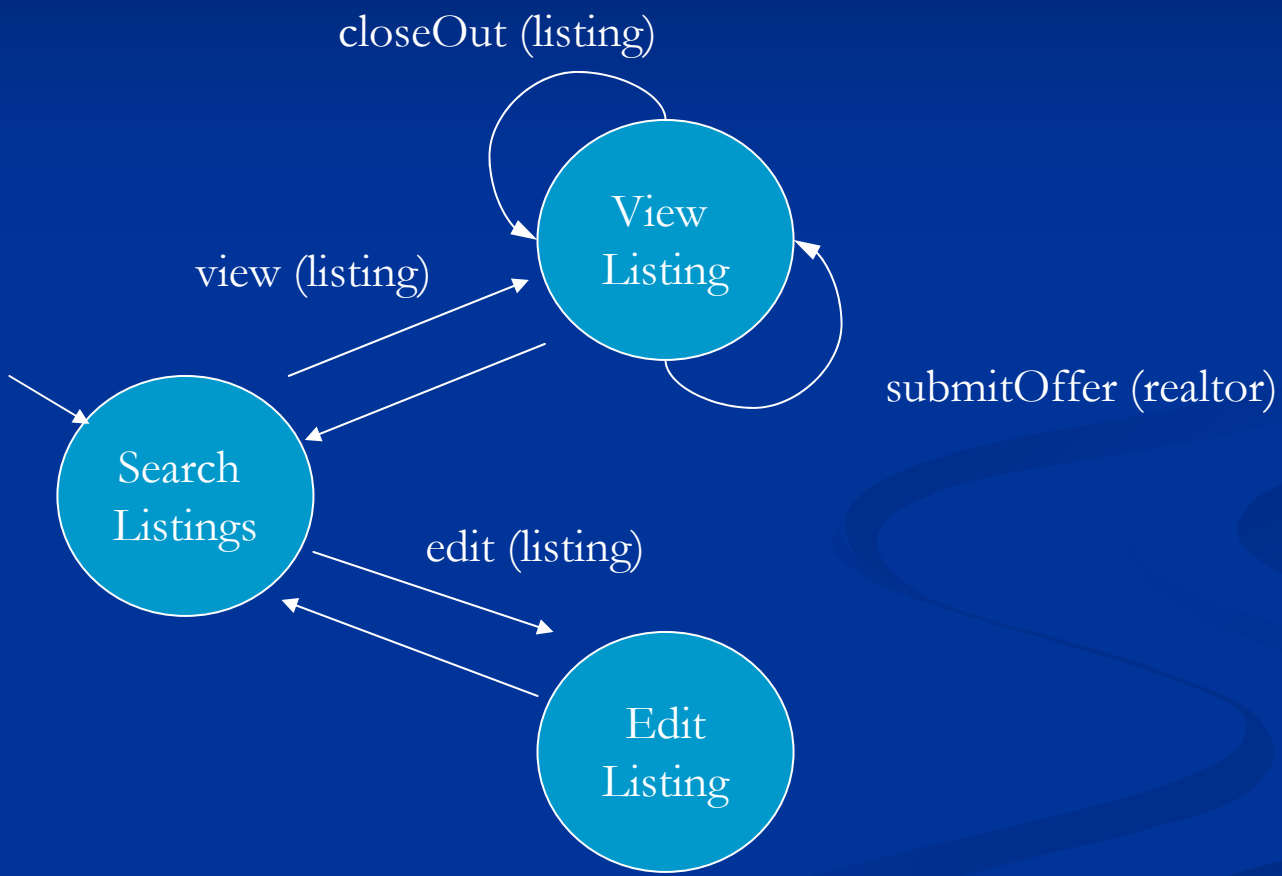
Separation of Concerns



Application Behavior Concerns

- Application Behavior Concerns describes the functional requirements of the proposed software
- Modeled as a collection of state transition diagrams or sequence diagrams

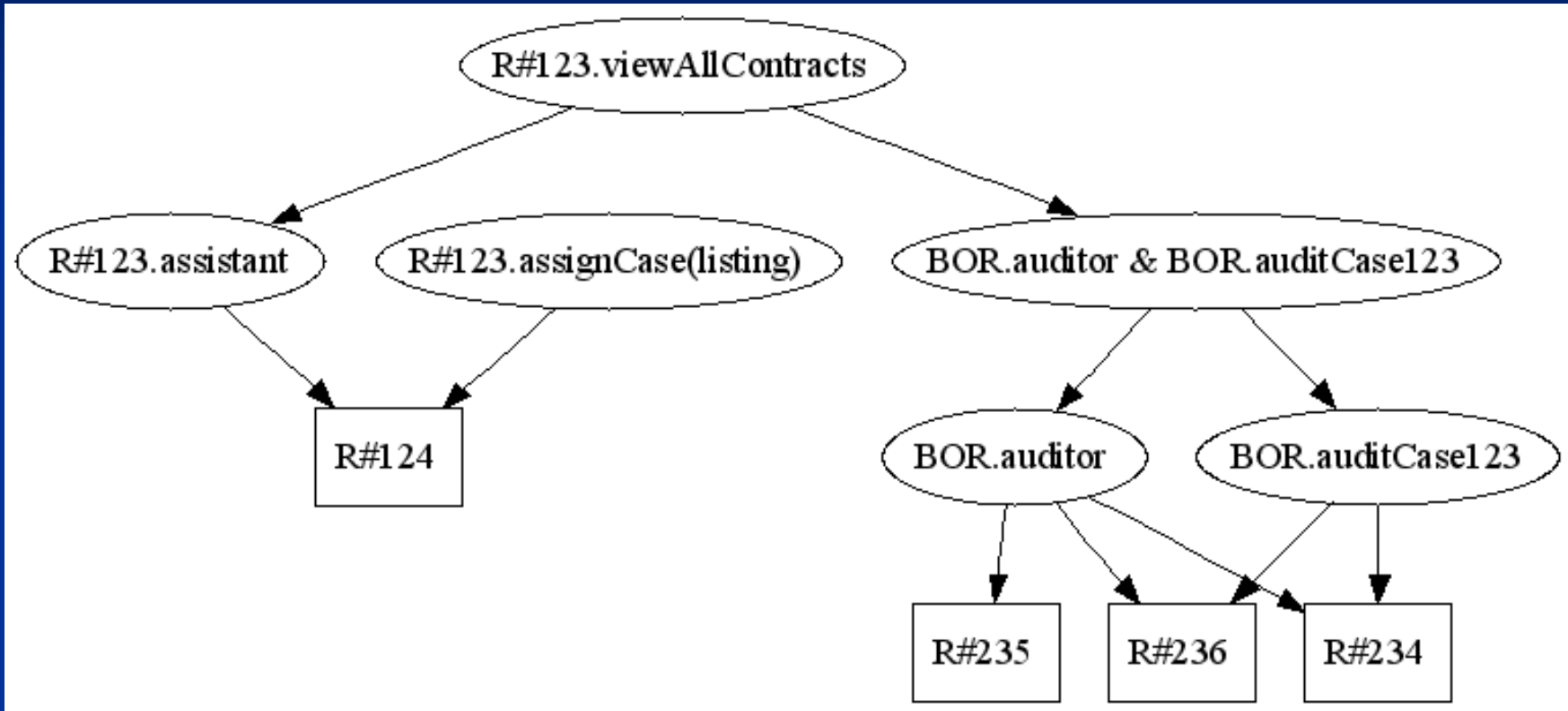
Application Behavior Concerns



Policy Concerns

- Describes how policy configuration may change
- Modeled with a state transition system or role dependency graph
- Static analysis of policy may reveal availability and safety violations by identifying counterexamples
- Demonstrated in previous work

Policy Example



Role	Permission
R#123.assignCase(listing)	view(listing), edit(listing), submitOffer(listing, *)
R#123.viewAllContracts	view(*)

Policy Related Questions

1. Can we verify that invariant access control properties hold regardless of how the dynamic policy changes?
2. How is the policy allowed to change?
3. Does the policy require application state information to determine authorization?

Security Mechanism Concerns

- Security Enforcement Point (SEP) – mechanism associated with a particular behavior, and makes a decision as to whether a principal may invoke a behavior
  - Consists of a guard component and a state component
  - Controls a particular application behavior; permits or denies behavior based on set of conditions

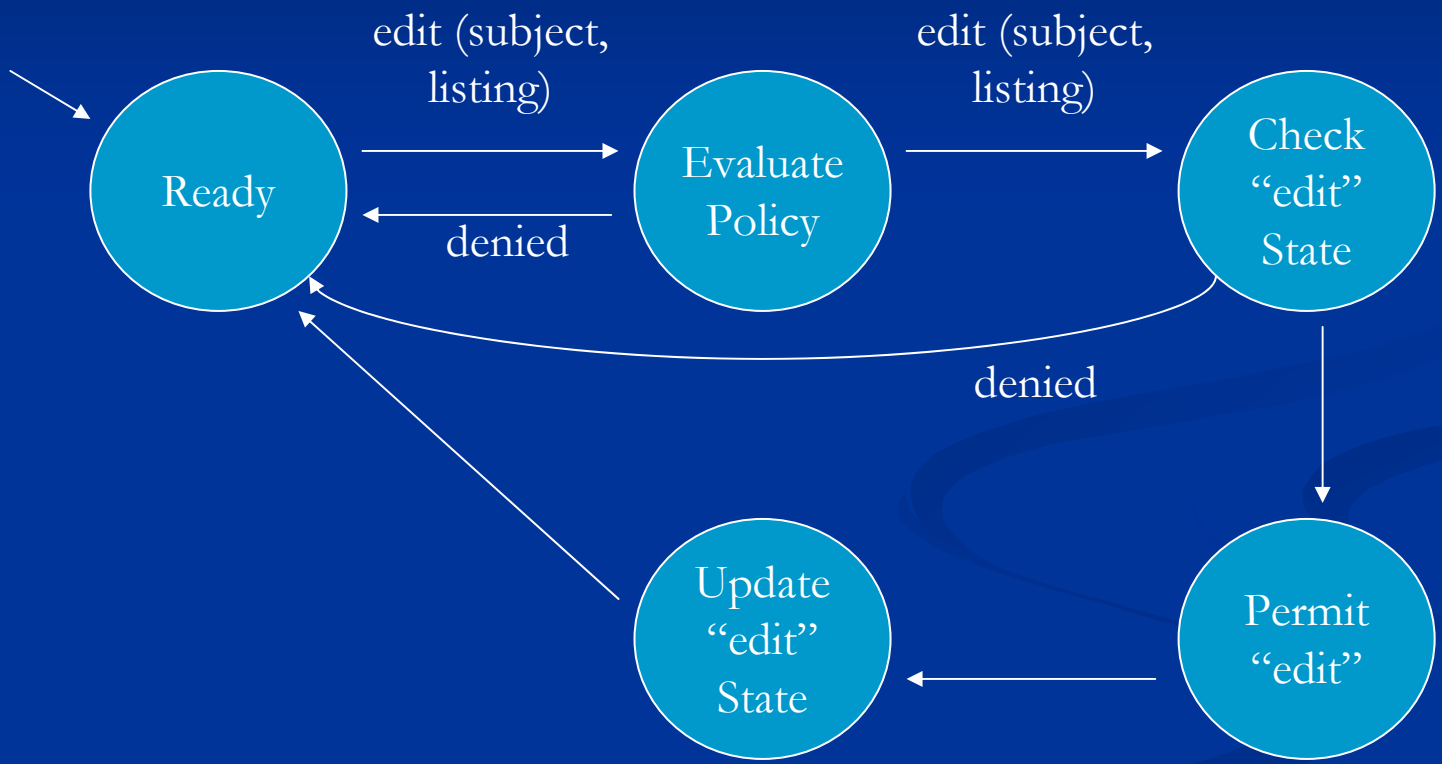
Security Mechanism Concerns

- Guard conditions:
  - Tests whether a user is authorized
  - Tests application state information
  - Tests state component information from one or more SEP's

Security Mechanism Concerns

- Models a collection of SEP's
  - Concerned with placement of SEP's to provide sufficient coverage over protected behaviors
  - Interaction of SEP's
  - Verifying security properties hold
    - Separation of Duty
    - Chinese Wall

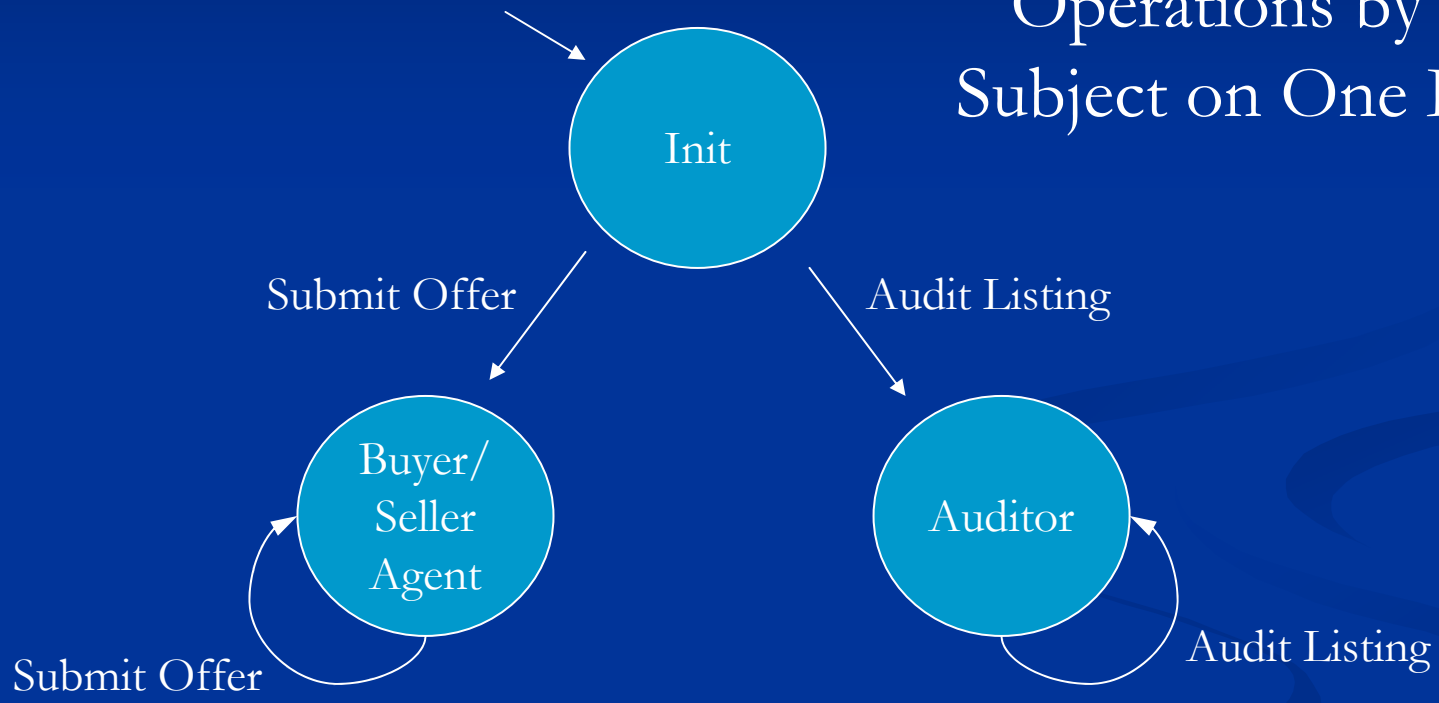
Security Mechanism Concerns



Activity Diagram for SEP (guard component)

Security Mechanism Concerns

Operations by One Subject on One Listing



State Transition Diagram for SEP (state component)

Security Mechanism Related Questions

1. Can we verify invariant security properties hold regardless of how the state of the SEP changes?
2. Is there some reachable state of a SEP that may lead to prohibited application behavior?
3. If access to an application behavior is denied by a SEP, will this cause some undesirable effect (inadvertent access, availability violation)?

Overview Security Analysis

- Can we find a counterexample (in terms of policy configuration and application behavior) where a security requirement fails?
  - Eve audited her own property contract
  - Eve audited a property contract and later served as opposing representation on the contract
  - Alice (an assistant to Bob) was denied to view Bob's property contracts due to Eve

Overview Security Analysis Questions

1. Are changes to the policy accurately reflected in the security mechanism models?
2. Is it ever possible for a policy state to override a constraint in the security mechanism models?
3. Is the policy enforceable, and if so, what security enforcement points are necessary?

Related Work

- SecureUML / Model Driven Security
  - Focus on Role-based Access Control (RBAC)
  - Constructed meta-models of user/role/permission relationships; synthesis model from appl. behavior
  - Lacking analysis of security requirements
  - Unclear how to handle delegation issues
- UMLsec
  - Emphasis on encryption over communication lines
  - Little description of how to describe access control

Future Research

- Do such requirements exist that may only be checked using a composite of all models?
- Are we always able to implement such automaton that satisfy given requirements?

Conclusions

- Multi-model analysis of access control from policy to application behavior provides a holistic view of software
- Separation of concerns allows independent analysis of each category of requirements
- Given a specific policy, we would like the ability to test the effect on application behavior

Questions?