Specification and Construction of Secure Distributed Collaboration Systems

Anand Tripathi
Department of Computer Science
University of Minnesota, Minneapolis
http://www.cs.umn.edu/Ajanta

This work was supported by NSF grant ITR 0082215 and EIA 9818338
Team Members

- Tanvir Ahmed (Ph.D. candidate)
- Richa Kumar (currently with Microsoft)
Outline

• Introduction
  – **Research Goals:**
• Requirements in Secure Collaboration
• A Model for Coordination and Security Specifications
• Middleware Execution Model and Design Issues
  – **Policy based construction of runtime environment**
• Verification of security properties using finite state model checking
• Future Directions

May 7, 2003
Introduction
Research Goals

• Rapid construction of secure distributed CSCW (Computer Supported Cooperative Work) systems from their high level specifications
• Collaboration groups may be formed ad hoc.
• Virtual organizations spanning different independent enterprises.
• Peer-to-peer management of collaboration activities
  – No single entity trusted by all to manage all aspects of a collaboration
### CSCW Systems

#### Groupware Systems
- Multiple users cooperate using shared artifacts towards some common objectives.
- Real-time synchronous interactions
- Tightly coupled
- Unstructured and ad-hoc coordination
- Concurrency issues
- Minimal security
- Whiteboard systems, Conferencing tools

#### Workflow Systems
- Asynchronous, loosely coupled interactions
- Structured interactions based on existing business models
- Persistence of shared objects
- Security: important concern
- Client-server model with securely managed servers
- Office / Health-care systems
A Virtual Organization

Activities

Enterprise A

Coordination / synchronization

Enterprise B

Security Requirements

Enterprise C

Enterprise D
Dynamic and Ad Hoc Collaborations

- **Peer-to-peer** management of collaboration activities.
- Different participants perform functions for managing various aspects of a collaboration environment.
  - Decentralized management
- Need for a **distributed trust model** for assigning management functions to the participants.
1. A specification model for CSCW systems.
   • Security and Coordination Requirements
2. Derivation of policy modules from the specifications.
3. A policy-driven middleware for secure distributed collaboration.

[Diagram showing the flow from Specification of a Collaboration Environment through Derivation of Policy Modules from Specification to Policy Driven Distributed Middleware Components and Services, all within the Runtime Environment.]
Policy-Based Approach

• Decouples coordination and security aspects of a collaboration from the implementation of its functionality.
  – Collaborative systems may evolve with changes in administrative policies and user experience.
  – Integration of new objects, devices, or tools may be needed.
  – Collaboration environments may span multiple administrative domains.
• Different policies can be easily plugged in.
Role-Based Model for CSCW

Course Examination: Example of a CSCW Activity

Users: Examiner, Grader, Student

Roles: Examiner, Grader, Student

Objects: GradeSheet, ExamPaper, AnswerBook
Research Approach

Collaboration Systems Specification

Analysis and Verification Tools
- Consistency of coordination constraints
- Coordination Dependency Analysis
- Security conflicts in assigning management functions to users

Derivation of Policy Templates
- Object Access Control Policies
- Event Subscription/Notification Policies
- Role Management Policies

Middleware Components and Functions
- Generic managers for roles and objects
- Creation of collaboration-specific policy objects at runtime
- Integration of policy objects with generic managers & application objects
A Role-Based Model for CSCW

- A role defines a set of operations
- Role operations represent a participant’s tasks and privileges to perform actions on shared objects
  - A role represents a protection domain
    - Access rights are associated with a role
- Role operations need satisfy coordination constraints.

- Current RBAC (Role Based Access Control) models do not adequately support the dynamic and context sensitive requirements of CSCW systems.
Security and Coordination Requirements in Collaboration Systems
Requirements for Collaboration Specification

- Coordination requirements
  - participants in the same role (intra-role)
  - participants in different roles (inter-role)
- Security requirements
  - Role admission
    - Authentication and authorization of users
  - “Separation-of-Duties”
  - Dynamic access control policies
    - Requires a unified model for coordination and security
- Enforcement of security policies
  - Who can be trusted to enforce a given policy?
**Intra-Role Coordination Models**

- **Independent participation**
  - Participants in a role work independently
  - Each participant has his/her own workspace
  - No coordination among the role participants

- **Cooperative participation**
  - Coordinate among themselves
    - A role task can be performed by only one person
      - Participants in the “nurse-on-duty” role administer daily medication only once to a patient.
    - Joint participation
      - All participants must perform the role operation together
        - Three banker managers open a bank vault jointly.
  - Unrestricted participation
    - Users sharing a whiteboard in a meeting
Role Admission Constraints

- Specifies conditions that need to be satisfied when a user requests to join a role.
- For example:
  - A list of users who are allowed to join
  - List of users to be disallowed to gain membership
  - A user's current or prior membership in some other roles
  - Role membership cardinality
  - Events that must happen before a user could be admitted in a role
Separation-of-Duties

• Static separation-of-duty
  – A user can never join two security sensitive roles.
• Dynamic separation-of-duty
  – A user cannot join two security sensitive roles concurrently.
• User-user conflicts
• User-role conflicts
• Operational separation-of-duty
  – In a business process, two sensitive operations should never be performed by the same user.
  – Object-based separation-of-duties
    • A user in a role cannot perform two sensitive operations on the same object.
Dynamic Access Control Policies

- A role operation may be allowed to be executed only after the occurrence of certain events.
- Role activation or invalidation conditions need to be checked and enforced at runtime.
- History-based separation-of-duties concerns.
- Access policies for an object may change:
  - Creation/termination of activities
  - Completion of some collaboration phase
Privacy

• Hide identities of participants’ in one role from other
  – E.g., Graders do not know the identity of the candidate
• Presence of a participant may be made visible only through a role or a pseudonym in a role
Hierarchical Activity Organization

- New nested activities may be created with existing or new roles
- An activity is instantiated multiple times, possibly concurrently, with different sets of users and objects

Requirement:

Certain roles/objects in the parent activity may be needed to be bound to the roles/objects in a new activity.
A Model for Specifying Collaboration Systems
Collaboration Activity

• Activity Definition
  – It defines a naming scope for objects and roles.
  – A large-scale collaboration may involve many activities, which may be nested hierarchically.

• Activity Template
  – Defines a reusable pattern for collaboration.
Activity Template Specification

1. Role Specification
   - Role Admission Constraints
     • Role operations for users to: join, leave, admit, remove
   - Role Activation Constraints
     • Conditions that must be true for a user to invoke any of the role operations
   - Role Operations: an operation definition has two parts
     • Precondition and Action

2. Object Specification
   - Method Signature
   - Access Control

3. Nested Activity Templates
Activity Creation

• Role Assignment
  – **Static Assignment (Role Reflection):**
    • All users from a role in the parent activity are assigned to a role in the new activity (specified in activity definition).
  – **Dynamic Assignment:**
    • Users are assigned to roles in the new activity at the time of activity creation.

• Objects in the parent activity may be needed to be passed to nested activities.
  – Access control policies for such objects may need to be updated.
Example: Course Activity

Activity Course

Role Assistant
Role Instructor
Role Student

GradeSheet

Activity Examination

Role Grader
Role Examiner
Role Examinee

ExamPaper
AnswerBook

Activity ExamSession

Role Checker
Role Candidate

Role assignment
Role reflection
Object parameters
Activity, Role, Object Management

• Meta Roles
  – **Creator role**: user initiating an activity instance.
    • Creator may not always be trusted for managing the activity.
  – **Owner role** is trusted to manage the activity instance.
    • The owner role has “vested interest” in managing the entity.
Activity, Role, Object Management

- **Activity ownership:**
  - Activity template can specify a role in the outer scope as the owner.
  - If not specified, the owner of the parent activity is the owner.

- **Role ownership**
  - Activity template can specify the owner.
  - If not specified, the owner of its activity is the owner.

- **Object ownership**
  - Role creating an object is its owner.
Psuedo-variables: thisUser, thisRole, ...

Membership functions:

Boolean function member( UserID, RoleName )

member ( thisUser, Instructor )

member ( Tom, thisRole )

List of current members in a role is given by:

members( RoleName )

Number of members in a list:

# (members( RoleName ))
Event Specification

• Three types of events related to each role operation and object method execution:
  – “request”, “start”, “finish”
    • (Model by Roberts and Verjus, IFIP 1977)
    – Example: ExamSession.Checker.Grade.finish

• Event Counters:
  
  #(eventName): number of times the event has occurred

• Derived events:
  – Filtering an event list based some predicate
  – Example: opName.start(invoker=John)
ExamSession Activity

- An instance of this activity is created by a student in the Examinee role.
- Only the student creating this activity should be able to join the Candidate role.
- Only a member of the Grader role in the Examination activity can join the Checker role.
- This activity must be managed at a Grader’s node, and not at its creator’s node (i.e. student’s node).
ExamSession Specification (1)

ACTIVITY ExamSession (OWNER Grader,
    OBJECTS (ExamPaper exam, AnswerBook ans),
    ASSIGNED_ROLES Candidate) {

    TERMINATION_CONDITION # (Checker.Grade.finish) > 0

    ROLE Checker {
        ADMISSION_CONSTRAINTS
            #members(thisRole) < 1
            & member(thisUser, parentActivity.Grader)

        OPERATION Grade {
            PRECONDITION # (Candidate.Submit.finish) = 1
            ACTION ans.setGrade(data) }
    }

    ROLE Candidate { ... }
ROLE Candidate {
  ADMISSION CONSTRAINTS
    member(thisUser, parentActivity.Examinee) &
    member(thisActivity.Creator, thisUser) & members(thisRole) < 1
  ACTIVATION CONSTRAINTS
    date > DATE(Mar, 22, 2002, 8:00) & date < DATE(Mar, 22, 2002, 11:00)
  OPERATION StartExam {
    PRECONDITION #(StartExam.start) = 0
    ACTION { ans = new AnswerBook( ); exam.readPaper( ) }
  OPERATION Write {
    PRECONDITION #(StartExam.finish) = 1
    ACTION ans.writeAnswer(data)
  OPERATION Submit {
    PRECONDITION #(Write.finish) > 0
    ACTION ChangeOwner (ans, Checker) }
}
Middleware Framework
Middleware Design Issues

1. Decentralized management of roles/objects:
   - All user nodes may not be equally trusted
   - A single node may not be trusted by all users
   - How to select a node where an entity should be managed?

2. Consistency issues in distributed enforcement of coordination preconditions.

3. Security issues in event based coordination
   - Only the events from authorized entities must be used
   - Policies for authorized event subscription/notification

4. Dynamic access control policies depend on the collaboration state
   - Policies for access control of shared objects may change with time as new activities are created
Middleware Components and Services

Collaboration Specification with Application Level Objects

- Role Definitions
  - Policy Modules
  - Generic Role Managers

- Object Definitions
  - Policy Modules
  - Generic Object Managers

- Activity Definitions
  - Policy Modules
  - Generic Activity Managers

Middleware Components

- Name Service
- Public Key Service
- Activity Management

Middleware Services
User Interaction Model

- User Authentication by role manager
- Role membership certificates
- Invocation of role operations by user

- Authentication of role manager by the object manager
- Invocation of object operations on behalf of the user
Policy Modules

• Role Management Policy Modules
  – Role admission, activation, and operation preconditions

• Role-based Access Control Policy Modules
  – Policies for creating objects and activities
  – Access control on object method invocation

• Event Subscription and Notification Policy Modules
  – Entities allowed to subscribe to certain types of events
  – Entities allowed to send specific types of events
Access Control Policy (ACP)
Template

OBJECT = ACTIVITY(Course, $x). ACTIVITY(Examination, $y)
.OBJECT(ExamPaper, $z)

OWNER = $x.Instructor
ENTRY {
    SUBJECT = $y.Examiner
    PERMISSION = setPaper
    CONDITION = ( #( $y.Examiner.SetPaper.start) = 0 ) }
ENTRY {
    SUBJECT { ACTIVITY_TEMPLATE = $y.ExamSession,
               ROLE = Candidate }
    PERMISSION = readPaper }
ACP for an ExamPaper Object

\[ x = \text{Course.chemistry}, \ y = \text{..Examination.midterm}, \ z = \text{..ExamPaper.exam} \]

OBJECT = Course.chemistry.Examination.midterm.ExamPaper.exam
OWNER = Course.chemistry.Instructor
ENTRY {
    SUBJECT = \text{..Examination.midterm.Examiner}
    PERMISSION = setPaper
    CONDITION = ( \#(..Examination.midterm.Examiner.SetPaper.start) = 0 )
}
ENTRY {
    SUBJECT \{ ACTIVITY TEMPLATE = \text{..Examination.midterm.ExamSession},
    ROLE = Candidate \}
    PERMISSION = readPaper \}
Verification of Security Properties
Objectives of Policy Verification

- User interactions follow coordination and task flow requirements.
- Roles do not have conflicting or inconsistent constraints.
- Confidential information cannot flow to unauthorized users.
- Authorized information can be accessed.
- Any temporal or conditional constraints on accessing objects can be satisfied.
- The safety property that no rights can be leaked to unauthorized users.
Verification of Global Properties

1. Reachability of Operations
   - Example:
     - \texttt{OPERATION Op1 PRECONDITION \#(Op2.finish) = 1}
     - \texttt{OPERATION Op2 PRECONDITION \#(Op1.finish) = 1}

2. Task Flow
   - Requirements are expressed in \textit{path expression} constructs: (Campbell and Habermann, 1974)
     - sequence(();), selection( () ) with a count (:n) restrictor
     - Example:
       \begin{verbatim}
       Examination := Examiner.SetPaper;
       Examinee.ExamSession.start
       \end{verbatim}
Global Security Properties

3. Role Based Constraints
   – Example of inconsistent constraint:
     ROLE B VALIDATION CONSTRAINTS \( ! \text{member}(A) \)
     ROLE C ADMISSION CONSTRAINTS \( \text{member}(A) \land \text{member}(B) \)

4. Confidentiality and Information Flow
   – Condition based constraints
     • “A participant of the examinee role cannot access the content of the exam paper before start of his/her own exam session”.
     • “Identity of a candidate should not be known to the grader until the grades are submitted”.
Global Security Properties

5. Integrity and Access Leakage
   – Due to owner assignments, participants of the owner roles get extended privileges.
   – Express integrity policies to check unintentional leakage of access rights.
     • “A participant of the examinee role can only modify his/her answer book, and that is allowed only during his/her exam-session”.

6. Decentralized Management
   – *Untrusted* users in administrative roles may actively try to violate sensitive security requirements
Verification Methodology

- Verification is performed by modeling the system using SPIN.
- Problem: Search space explosion
- Solution: Aspect specific verification:
  - Ignore properties, which are not in concern when verifying a specific property or can be independently verified.
Verification Methodology

- Verification methodology follows a precedence among the properties it checks.
- Based on aspects of the global requirements developed four verification models:
  1. Task Model
  2. Role Model
  3. Information Flow Model
  4. Owner Assignment Model
     i. Role constraints
     ii. Entity access and creation
     iii. Precondition check
Related Work

• Other policy based approaches:
  – COCA: Prolog based coordination policies for interactive applications
  – DCWPL: Coordination language to deal group interaction issues and uses predefined roles and functions
  – CSDL (Cooperative Systems Design Language, ICDCS’94)

• Similarity with programming methodologies: Aspect-oriented programming, Generative programming,
Conclusions

• We have developed a role based specification model and a policy-driven middleware to manage the runtime execution environment of secure distributed collaboration systems.

• The middleware support
  – Distributed management of collaboration entities
  – Derivation and enforcement of security and coordination policies based on trust relationships among users and roles

• Verification of Security Properties using SPIN
Current and Future Work

• Experimentation with collaborative systems with different security and coordination policies.
  (Tanvir Ahmed and REU students Jordan Raney and Sara Holmdahl)

• Object caching and replication
  (John Eberhard)

• Verification of security properties
  (Tanvir Ahmed, Ivan Osipkov)

• Extend this system to support ubiquitous and pervasive computing environments.
Thank you.