Formal Modeling and Verification of Workflows with Security Considerations

Weiqiang Kong\textsuperscript{1}, Kazuhiro Ogata\textsuperscript{1,2}, Kokichi Futatsugi\textsuperscript{1}

\textsuperscript{1}Japan Advanced Institute of Science and Technology (JAIST)
\textsuperscript{2}NEC Software Hokuriku, Ltd.

Oct 21, 2005
Outline of the Talk

- Security considerations in workflow systems
- A sample workflow analyzed
- The OTS/CafeOBJ method
- Summary
Secure Workflow and Secure WFMS

- A **secure workflow** is a computer supported business process that is capable to against security threats and further satisfies the security requirements defined by the workflow modeler.

- A **secure workflow (management) system** is a workflow (management) system that can specify, manage and execute a secure workflow.
Security Considerations

- WfMC summarizes them as:
  - Authentication
  - Authorization
  - Access Control
  - Audit
  - Data Privacy
  - Data Integrity
  - Non Repudiation
  - Security Management and Administration

WfMC: Workflow Management Coalition (www.wfmc.org)
RBAC and SoD

- Role-Based Access Control (RBAC) mechanism
  - *Roles* represent agents intended to perform certain job functions.
  - *Users* in turn are assigned to appropriate roles.
  - Privileges are granted to (revoked from) *Roles*.

- Separation of Duty (SoD) constraints
  - Aim at preventing fraud within an organization.
  - Decompose a complex action into several smaller steps, which are executed by different roles.
A Sample Workflow
– travel expenses reimbursement workflow

Roles :  \( R = \{ \text{employee, manager, secretary} \} \)

Tasks :  \( T = \{ \text{apply, evaluate1, evaluate2, notify, mtransfer} \} \)
SoD Constraints in the Sample Workflow

- **Static separation of duties:**
  - apply → employee
  - evaluate1, evaluate2 → manager
  - notify, mtransfer → secretary

- **Dynamic separation of duties:**
  - A manager should not be allowed to evaluate his own travel application.
  - A secretary should not be allowed to transfer the refund of his own travel expenses.
  - A manager should not be allowed to perform both evaluate tasks in the same workflow instance.
Informal Definitions of Properties to be Analyzed – Goals of Analysis

- **RBAC mechanism & SoD constraints**
  - A subject can execute a task if and only if role has been granted privilege to execute this task and the subject belongs to this role.
  - A manager should not be allowed to evaluate his own travel application.
  - A secretary should not be allowed to transfer the refund of his own travel expenses.
  - A manager should not be allowed to perform both evaluate tasks in the same workflow instance.

- **Workflow can run successfully from the initial state to the final state with these security considerations. (Main Goal)**
Observational Transition System (OTS): General Introduction

- Assume that there exists a universal state space called $\Upsilon$, An OTS $S$ consists of $<O, I, T>$:
  - $O$: A set of observers. Each $o \in O$ is a function $o: \Upsilon \rightarrow D$, where $D$ is a data type. Given an OTS $S$ and two states $v_1, v_2 \in \Upsilon$, the equivalence between two states, denoted by $v_1 =_S v_2$, with respect to $S$ is defined as $\forall o \in O. o(v_1) = o(v_2)$.
  - $I$: The set of initial states such that $I \subseteq \Upsilon$.
  - $T$: A set of conditional transition rules. Each $t \in T$ is a function $t: \Upsilon /=_S \Upsilon /=_S$ on equivalence classes of $\Upsilon$ with respect to $=_S$. $t(v)$ is called the successor state of $v$ with respect to $t$. The condition $\zeta$, for a transition rule $t \in T$ is called the effective condition.
The Procedure of the OTS/CafeOBJ Method

1. The workflow, which consists of workflow process and security considerations, is modeled as an OTS.
   - Workflow process – sequence, and-split (joint), xor-split (joint)
   - Security consideration – RBAC and SoD

2. The OTS is then written in CafeOBJ – an algebraic specification language.

3. Express safety and liveness properties of the workflow in CafeOBJ.

4. Verify the OTS satisfies these properties by writing proof scores in CafeOBJ, and executing the proof scores in CafeOBJ system.
## Transition Rules and Observers

<table>
<thead>
<tr>
<th>Transition rules</th>
<th>apply, evaluate1, evaluate2, notify, mtransfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observers</td>
<td>mainpath, path1, path2, primainpath, pripath1, pripath2 ah, eh</td>
</tr>
</tbody>
</table>
Functions Used in the Specification

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exeby(T)</td>
<td>roles that are able to execute task T</td>
</tr>
<tr>
<td>assignto(S)</td>
<td>roles that subject S belongs to</td>
</tr>
<tr>
<td>assumedby (R)</td>
<td>subjects that role R contains</td>
</tr>
<tr>
<td>auth(R, T, O)</td>
<td>authorization record, which means for an object O, role R is granted the privilege to execute task T</td>
</tr>
<tr>
<td>exe(T, S, O)</td>
<td>execution record, which means for an object O, task T was executed by subject S</td>
</tr>
</tbody>
</table>

T: Task        R: Role        S: Subject        O: Object
**Transition Rules – evaluate1 (1)**

--- effective condition

op c-evaluate1 : Sys Subject Obj Quality -> Bool
eq c-evaluate1(S,SUB,F,Q) =
    (mainpath(S,F) = applied and path1(S,F) = p1start
    and pripath1(S,F) = auth(manager,evaluate1,F)
    and (SUB /in assumedby(manager))
    and not (exe(apply,SUB,F) /in eh(S,F))
    and not (exe(evaluate2,SUB,F) /in eh(S,F)).
Transition Rules – evaluate1 (2)

-- change process states

c eq mainpath(evaluate1(S,SUB,F,Q),F1)
   = (if F1 = F then mainpath(S,F) else mainpath(S,F1) fi)
   if c-evaluate1(S,SUB,F,Q) .

c eq path1(evaluate1(S,SUB,F,Q),F1)
   = (if F1 = F then approved1 else path1(S,F1) fi)
   if c-evaluate1(S,SUB,F,Q) and Q = good .

c eq path1(evaluate1(S,SUB,F,Q),F1)
   = (if F1 = F then rejected1 else path1(S,F1) fi)
   if c-evaluate1(S,SUB,F,Q) and Q = bad .

c eq path2(evaluate1(S,SUB,F,Q),F1)
   = (if F1 = F then path2(S,F) else path2(S,F1) fi)
   if c-evaluate1(S,SUB,F,Q) .
Transition Rules – evaluate1 (3)

-- change authorizations

ceq primainpath(evaluate1(S,SUB,F,Q),F1)
   = (if F1 = F then primainpath(S,F) else primainpath(S,F1) fi)
   if c-evaluate1(S,SUB,F,Q) .

ceq pripath1(evaluate1(S,SUB,F,Q),F1)
   = (if F1 = F then auth(secretary,notify,F) else pripath1(S,F1) fi)
   if c-evaluate1(S,SUB,F,Q) and Q = bad .

ceq pripath1(evaluate1(S,SUB,F,Q),F1)
   = (if F1 = F then auth(secretary,mtransfer,F) else pripath1(S,F1) fi)
   if c-evaluate1(S,SUB,F,Q) and Q = good .

ceq pripath2(evaluate1(S,SUB,F,Q),F1)
   = (if F1 = F then pripath2(S,F) else pripath2(S,F1) fi)
   if c-evaluate1(S,SUB,F,Q) .
Transition Rules – evaluate1 (4)

-- change execution and authorization histories

ceq eh(evaluate1(S,SUB,F,Q),F1)
  = (if F1 = F then exe(evaluate1,SUB,F),eh(S,F) else eh (S,F1) fi)
  if c-evaluate1(S,SUB,F,Q) .

ceq ah(evaluate1(S,SUB,F,Q),F1)
  = (if F1 = F then auth(secretary,mtransfer,F),ah(S,F) else ah(S,F1) fi)
  if c-evaluate1(S,SUB,F,Q) and Q = good .

ceq ah(evaluate1(S,SUB,F,Q),F1)
  = (if F1 = F then auth(secretary,notify,F),ah(S,F) else ah(S,F1) fi)
  if c-evaluate1(S,SUB,F,Q) and Q = bad .

ceq evaluate1(S,SUB,F,Q) = S
  if not c-evaluate1(S,SUB,F,Q) .
Formal Definitions of the Properties

**INV1**: c-apply\((s, sub, form)\) and sub /in assumedby(employee) **implies** auth(employee, apply, form) /in ah\((s, form)\)

**INV2**: ((exe(evaluate1, sub, form) /in eh\((s, form)\) or exe(evaluate2, sub, form) /in eh\((s, form)\)) and sub /in assumedby(manager)) **implies** not(exe(apply, sub, form) /in eh\((s, form)\))

**INV3**: (exe(mtransfer, sub, form) /in eh\((s, form)\) and sub /in assumebdby(secretary)) **implies** not(exe(apply, sub, form) /in eh\((s, form)\))

**INV4**: (exe(evaluate1, sub1, form) /in eh\((s, form)\) and exe(evaluate2, sub2, form) /in eh\((s, form)\)) **implies** not(sub1 = sub2)

**LIV**: (mainpath\((s, form)\) = start and path1\((s, form)\) = p1start and path2\((s, form)\) = p2start and primainpath\((s, form)\) = auth(employee, apply, form) and pripath1\((s, form)\) = null and pripath2\((s, form)\) = null) **leads-to** mainpath\((s, form)\) = end
Summary

- Equation-based method – the OTS/CafeOBJ method:
  - Specification (RBAC and SoD) is based on equations.
  - Verification is based on equational reasoning.
  - Different workflow instances are distinguished explicitly.
  - Analysis of workflow process is combined with authorization flow.
Thanks!