Delegation and Satisfiability in Workflow Systems

Jason Crampton    Hemanth Khamhammettu
Information Security Group
Royal Holloway, University of London

SACMAT 2008
Satisfiability is an important consideration in workflow management systems (WfMSs)

- Given an authorization policy and a set of constraints, does there exist a set of authorized users that can complete the workflow?

Delegation is of increasing interest in workflow systems

- Delegation can increase flexibility in the workplace
- A successful delegation changes authorization information
One-Page Overview

Satisfiability is an important consideration in workflow management systems (WfMSs)

- Given an authorization policy and a set of constraints, does there exist a set of authorized users that can complete the workflow?

Delegation is of increasing interest in workflow systems

- Delegation can increase flexibility in the workplace
- A successful delegation changes authorization information

How does delegation affect workflow satisfiability?
Constrained Workflows

A constrained workflow authorization schema \( W = (T, A, C) \) comprises

- a set of (abstract) tasks \( T \)
- authorization information \( A \subseteq U \times T \) associates users with tasks (for which they are authorized)
- a set of constraints \( C \) specifies constraints on the execution of tasks by authorized users
Constrained Workflows

A constrained workflow authorization schema $W = (T, A, C)$ comprises

- a set of (abstract) tasks $T$
- authorization information $A \subseteq U \times T$ associates users with tasks (for which they are authorized)
- a set of constraints $C$ specifies constraints on the execution of tasks by authorized users

An instance of $W$ is created and managed by the WfMS and comprises

- a set of (concrete) tasks
- tasks are performed by authorized users that satisfy constraints
Workflow Satisfiability

An execution assignment is an assignment of concrete tasks to authorized users

- A valid execution assignment is an assignment of all tasks to authorized users, such that no constraint is violated
- A workflow schema $W$ is satisfiable if there exists a valid execution assignment for $W$
- A workflow instance is satisfiable if all pending tasks can be assigned to authorized users such that no constraint is violated
Complexity

Determining whether a schema is satisfiable is an NP-complete problem in general (Wang and Li, ESORICS 2007)...

- Checking whether an execution assignment is valid can be performed in polynomial time
- The number of execution assignments is $|T|^{|U|}$
Complexity

Determining whether a schema is satisfiable is an NP-complete problem in general (Wang and Li, ESORICS 2007)...

- Checking whether an execution assignment is valid can be performed in polynomial time
- The number of execution assignments is $|T|^{|U|}$

...although for most practical examples fast algorithms exist
Determining whether a schema is satisfiable is an NP-complete problem in general (Wang and Li, ESORICS 2007). . .

- Checking whether an execution assignment is valid can be performed in polynomial time
- The number of execution assignments is $|T||U|

...although for most practical examples fast algorithms exist

Determining whether an instance is satisfiable is equivalent to determining whether a modified schema is satisfiable (Crampton, SACMAT 2005)
Workflow Execution Models: WDEM

WfMS-driven execution model (WDEM)

- A tasklist is generated when a workflow schema is instantiated
- WfMS assigns tasks to users on basis of authorization information and ensures no constraints are violated
- User is obliged to perform the task(s) to which she has been assigned
- Tasklists may be static or dynamic
Workflow Execution Models: WDEM

WfMS-driven execution model (WDEM)

- A tasklist is generated when a workflow schema is instantiated
- WfMS assigns tasks to users on basis of authorization information and ensures no constraints are violated
- User is obliged to perform the task(s) to which she has been assigned
- Tasklists may be static or dynamic

We make two important observations

- A static tasklist is a valid execution assignment
- A dynamic tasklist is a satisfiable instance
User-driven execution model (UDEM)

- The WfMS simply manages the execution of a workflow instance
- Users initiate (access) requests to perform pending tasks
User-driven execution model (UDEM)

- The WfMS simply manages the execution of a workflow instance
- Users initiate (access) requests to perform pending tasks

The workflow access control mechanism decides whether the request should be granted

- Clearly user must be authorized
- The instance must remain satisfiable if the request is granted
Introduction

Informally, delegation is an act of temporarily authorizing a user (for a permission, to perform a task, etc...)  
- The delegator may **grant** authorization to the delegatee  
- The delegator may **transfer** authorization to the delegatee
Introduction

Informally, delegation is an act of temporarily authorizing a user (for a permission, to perform a task, etc. . . )

- The delegator may **grant** authorization to the delegatee
- The delegator may **transfer** authorization to the delegatee

Task delegation can occur in two basic forms in WfMSs

- **Concrete task** delegation authorizes the delegatee to perform the delegated task **only** in the specified workflow instance
- **Abstract task** delegation authorizes the delegatee to perform the delegated task in **any** workflow instance
Delegation in Workflows

The semantics of a delegation operation depends on three factors:

- the workflow execution model (WDEM or UDEM)
- the type (abstract or concrete) of the delegated task
- the type (grant or transfer) of the delegation operation
Delegation in Workflows

The semantics of a delegation operation depends on three factors

- the workflow execution model (WDEM or UDEM)
- the type (abstract or concrete) of the delegated task
- the type (grant or transfer) of the delegation operation

Note that

- grant of concrete tasks is meaningless in WDEM
- grant and transfer of concrete tasks is meaningless in UDEM

A further question arises for transfer of abstract tasks in WDEM

- Are concrete task assignments transferred to the delegatee (cascading transfer) or not (non-cascading)?
## Summary of Delegation Operations

### Concrete Tasks

<table>
<thead>
<tr>
<th></th>
<th>Grant</th>
<th>Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDEM</td>
<td>n/a</td>
<td>Yes</td>
</tr>
<tr>
<td>UDEM</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

### Abstract Tasks

<table>
<thead>
<tr>
<th></th>
<th>Grant</th>
<th>Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Non-cascading</td>
</tr>
<tr>
<td>WDEM</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UDEM</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Introduction

Delegation modeled as access request
  - Delegation policy will decide whether request is authorized
  - Request may be granted or denied

Granting request will change authorization state
  - Granting request may result in unsatisfiable instance or schema
  - Therefore must have additional satisfiability checks when deciding delegation requests
## Concrete Tasks

<table>
<thead>
<tr>
<th>Concrete Tasks</th>
<th>Grant</th>
<th>Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDEM</td>
<td>n/a</td>
<td>Updates tasklist</td>
</tr>
<tr>
<td>UDEM</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Must check whether revised tasklist is a
- valid execution assignment (static tasklists)
- satisfiable instance (dynamic tasklists)
Abstract Tasks

<table>
<thead>
<tr>
<th>Grant</th>
<th>Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-cascading</td>
</tr>
<tr>
<td>WDEM</td>
<td>Updates A</td>
</tr>
<tr>
<td>UDEM</td>
<td>Updates A</td>
</tr>
</tbody>
</table>

Grant delegations are “monotonic”

- Any valid execution assignment remains valid
- Satisfiability not an issue for grant delegation requests
Abstract Tasks

<table>
<thead>
<tr>
<th>Grant</th>
<th>Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-cascading</td>
</tr>
<tr>
<td>WDEM</td>
<td>Updates A</td>
</tr>
<tr>
<td>UDEM</td>
<td>Updates A</td>
</tr>
</tbody>
</table>

A transfer is permitted if

- the updated workflow authorization schema is satisfiable
- all updated tasklists are valid execution assignments and/or satisfiable instances
Abstract Tasks

<table>
<thead>
<tr>
<th>Grant</th>
<th>Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-cascading</td>
</tr>
<tr>
<td>WDEM</td>
<td>Updates A</td>
</tr>
<tr>
<td>UDEM</td>
<td>Updates A</td>
</tr>
</tbody>
</table>

A transfer is permitted if

- the updated workflow authorization schema is satisfiable
- all updated tasklists are valid execution assignments and/or satisfiable instances

Necessary but not sufficient...
Example: WDEM, Dynamic, Non-cascading Transfer

- Set of tasks $T = \{t_1, t_2, t_3\}$
- Set of users $\{a, b, c\}$
- $t_1$ and $t_2$ must be performed by different users
- $t_2$ and $t_3$ must be performed by different users

<table>
<thead>
<tr>
<th>Schema</th>
<th>Before transfer</th>
<th>After transfer</th>
<th>Is satisfiable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A($t_1$) = {a, b}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A($t_2$) = {a, c}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A($t_3$) = {b, c}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tasklist</td>
<td>[(t_1, a), (t_2, c)]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: WDEM, Dynamic, Non-cascading Transfer

- Set of tasks $T = \{t_1, t_2, t_3\}$
- Set of users $\{a, b, c\}$
- $t_1$ and $t_2$ must be performed by different users
- $t_2$ and $t_3$ must be performed by different users

<table>
<thead>
<tr>
<th>Schema</th>
<th>Before transfer</th>
<th>After transfer</th>
<th>Is satisfiable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A(t_1) = {a, b}$</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>$A(t_2) = {a, c}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A(t_3) = {b, c}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tasklist</td>
<td>$[(t_1, a), (t_2, c)]$</td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>
Example: WDEM, Dynamic, Non-cascading Transfer

- Set of tasks $T = \{t_1, t_2, t_3\}$
- Set of users $\{a, b, c\}$
- $t_1$ and $t_2$ must be performed by different users
- $t_2$ and $t_3$ must be performed by different users

$b$ performs non-cascading transfer of $t_3$ to $a$

<table>
<thead>
<tr>
<th></th>
<th>Before transfer</th>
<th>After transfer</th>
<th>Is satisfiable?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema</strong></td>
<td>$A(t_1) = {a, b}$</td>
<td>$A(t_1) = {a, b}$</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>$A(t_2) = {a, c}$</td>
<td>$A(t_2) = {a, c}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A(t_3) = {b, c}$</td>
<td>$A(t_3) = {a, c}$</td>
<td></td>
</tr>
<tr>
<td><strong>Tasklist</strong></td>
<td>$[(t_1, a), (t_2, c)]$</td>
<td>$[(t_1, a), (t_2, c)]$</td>
<td>No</td>
</tr>
</tbody>
</table>
Abstract Tasks

<table>
<thead>
<tr>
<th>Grant</th>
<th>Non-cascading</th>
<th>Cascading</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDEM</td>
<td>Updates A</td>
<td>Updates A and tasklists</td>
</tr>
<tr>
<td>UDEM</td>
<td>Updates A</td>
<td>n/a</td>
</tr>
</tbody>
</table>

A non-cascading transfer is permitted if

- the updated workflow authorization schema is satisfiable
- all existing dynamic tasklists remain satisfiable instances

A cascading transfer is permitted only if

- the updated workflow authorization schema is satisfiable
- all existing dynamic tasklists remain satisfiable instances
- all updated tasklists are valid execution assignments and/or satisfiable instances
Contributions and Observations

Part of an ongoing research effort to understand delegation in WfMSs (IJIS, 7(2), 2008; SAC 2008; SACMAT 2008)

- There are different workflow execution models
- There are different delegation operations

This paper is the first to consider delegation and satisfiability in workflow systems

- Ensuring satisfiability is important when delegation is supported
- The paper also includes the study of satisfiability for role delegation in WfMSs that employ role-based access control
Future Work

Consider more fine-grained treatment of tasks

- Notion of “state” for tasks
- Typical states include: initialized, assigned and complete
- Useful for considering more complex workflow patterns

Revocation and workflow satisfiability

- Does permitting a revocation request affect workflow satisfiability?

Delegation and resiliency

- Does delegation improve resiliency?