Overview

- Top-Down Role Engineering
- Bottom-Up Role Engineering

Determining Role Rights from Use Cases
(Fernandez and Hawkins '97)

- Simple and complete method to determine needed rights for roles in a system
- Utilize concept of use cases
  - Standard in object oriented system development
  - Users are interviewed to elicit their ways of interacting with the system and interaction descriptions recorded in a standardized format
- Extend use cases with rights specifications
- Determine all of a role’s rights from the collection of all use cases for the system
  - In strict accordance with principle of least privilege


Scenario Driven Role Engineering
(Neumann and Strembeck '02)

- Adaptation of the software engineering process for identification of system requirements for role engineering
- Define a scenario-driven role engineering process
  - Seven key steps

1. Identify & Model Usage Scenarios
(Neumann and Strembeck '02)

- Initially, described with a short sentence
  - E.g., “enter an exam result into a patient file” in a hospital EMR
- Step sequence explicitly defined and written down
  - Structured text and corresponding diagram
- Provide a unique name to identify the scenario and facilitate search operations

2. Permission Derivation
(Neumann and Strembeck '02)

- Review each scenario
  - check which operation a subject (e.g., a user) needs to perform to complete this step
  - For each operation, define and store an (operation, object) pair in the permission catalog
    - No duplication of permissions
    - Differentiate abstract and basic permissions (i.e., derive hierarchy of permissions)
3. Identification of Permission Constraints
(Neumann and Strembeck ‘02)

- Types of constraints
  - Separation of duty
  - Cardinality
  - Time-dependencies
  - Maximum executions in an interval
- For each type of constraint, identify specific static separation of duty (SSD) constraints by consulting with domain experts

4. Scenario Model Refinement
(Neumann and Strembeck ‘02)

5. Definition of Tasks & Work Profiles
(Neumann and Strembeck ‘02)

- Scenarios that logically belong together are combined into tasks
- Tasks used to define work profiles
- Specifications for tasks and work profiles are usually very different
  - diverse organizations / diverse information systems
  - Need domain experts
  - Most challenging part is to select the correct group of scenarios for a particular task

6. Derivation of a Preliminary Role-Hierarchy
(Neumann and Strembeck ‘02)

7. RBAC Model Definition
(Neumann and Strembeck ‘02)

- Conducted by security engineers
- Cannot be automated

Benefits/Drawbacks

- Benefits
  - Formal approach
  - Comprehensive
  - Utilized in large migrations (e.g., U.S. Department of Veterans Affairs used it to implement a large RBAC system for its hospitals)

- Drawbacks
  - Only limited automation possible
  - Extremely expensive
Overview

- Top-Down Engineering
- Bottom-Up Engineering
  - The Role Mining Problem
  - ORCA
  - FastMiner
  - Tiling

Bottom-Up

- Mine roles from existing user-permission assignments
- Helps to migrate legacy systems to RBAC
- Eases the task of role engineering by serving as a tool to assist security administrator
- However, it requires manual review of the mined results
- Complementary to top-down assessment

Discovering Inherent Roles

- Access Control
- Data

- Rights
- Right1
- Right2
- Right3
- Right4
- Right5
- Right6
- Right7
- Right8
- Right9
- Right10
- Right11
- Right12
- Right13
- Right14
- Right15
- Right16
- Right17
- Right18
- Right19
- Right20
- Right21

- Want to allow overlapping permission sets
- Don’t want to force all rights to be part of a role
- Don’t discard roles that have little support

An Enterprise View of Bottom-up Mining (Kuhlman et al. 2003)

- Industrial project – large-scale customers want to migrate to RBAC
  - based on already existing access rights patterns in their production IT systems
  - Develop process for detecting patterns in a database of access rights
  - Derive enterprise roles from these patterns
  - Tool: SAM Role Miner – based on Security Administration Manager (SAM) product

Formally

- Given: $A$, an $m \times n$ binary matrix (represents a UPA user-permission relation)
- Decompose $A$ into two matrices $B$ and $C$
  - $B$ is an $m \times k$ matrix (user-role relation UA)
  - $C$ is a $k \times n$ matrix (role-permission relation PA) such that $k$ is minimal

Decomposing $A$ is Easy!

- Extreme Case 1: Each user in a role by himself
  - $k = m$ (i.e., number of users)
  - $B = I$ (the identity matrix)
  - $C = A$ (i.e., UPA matrix)

- Extreme Case 2: Each permission in a role by itself
  - $k = n$ (i.e., number of permissions)
  - $B = A$ (i.e., UPA matrix)
  - $C = I$ (the identity matrix)

- Both are accurate decompositions!
- But $A$ is probably not minimal!
Decomposing $A$ is Easy!

- Alternative: Put all users in a single (the same) role
  - $k = 1$
- Highly likely to be very inaccurate
- So, the minimal set of roles should be the accurate set of roles
- Unfortunately, finding the minimal set is an NP-complete problem*


Let's work towards an approximation

Boolean Matrix Multiplication

$X \otimes Y = Z$

- Boolean matrix
  - $X \in \{0,1\}^{m \times k}$
  - $Y \in \{0,1\}^{k \times n}$
  - $Z \in \{0,1\}^{m \times n}$
- $z_{ij} = x_{ik}y_{kj}$
  - $\lor x_{ik} \land y_{kj}$
  - $\lor \ldots \lor x_{ik} \land y_{kj}$

Norms ($L_1$ specifically)

- $L_1$ norm of a $d$-dimensional vector $v \in X^d$ for a set $X$ is
  $$\|v\|_1 = \sum_{i=1}^{d} |v_i|$$
- Generalizes to distance between two vectors $v, w$
  $$\|v - w\|_1 = \sum_{i=1}^{d} |v_i - w_i|$$

Norms ($L_1$ specifically)

- When matrices $A$ and $B$ are in $X^{n \times m}$
  $$\|A - B\|_1 = \sum_{i=1}^{n} |a_{i} - b_{i}|$$
  $$= \sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij} - b_{ij}|$$
- $L_1$ is basically the number of bits that differ between two Boolean matrices

Norms are Useful in Physics, Statistics, and Computer Science

- $L_1$ is sometimes called
  - Manhattan distance
  - Manhattan length
  - City-block distance
  - Taxicab metric
  - ...
- In general, the $L_p$ norm is
  $$\|v\|_p = \left(\sum_{i=1}^{d} |v_i|^p\right)^{1/p}$$
- $L_2$ is usually defined as the number of non-zero elements of a vector (aka Hamming distance)… but it’s not really a norm (because it’s not absolutely homogenous)…
- $L_2$ is one of the more famous
  - It’s the Euclidean distance
δ-consistency

- $M(\text{rel})$ is the matrix form of the relation
- $U_A$, $P_A$, and $U_P$ are δ-consistent if and only if
  $||M(U_A) \otimes M(P_A) - M(U_P)||_1 \leq \delta$

Role Mining Problem (RMP)

- Given $U$, $P$, $U_P$.
- Find $R$ (roles), $U_A$, and $P_A$, such that
  - δ-consistent with $U_P$
  - minimizes the number of roles $k$

δ-approx RMP

- Given $U$, $P$, $U_P$.
- Find $R$ (roles), $U_A$, and $P_A$, such that
  - δ-consistent with $U_P$
  - minimizes the number of roles $k$

Minimal Noise RMP

- Given $U$, $P$, $U_P$, $k$
- Find a set of $k$ roles, $R$, $U_A$, and $P_A$ that minimizes:
  $||M(U_A) \otimes M(P_A) - M(U_P)||_1$

Minimal Noise (MinNoise) RMP

- Instead of minimizing the number of roles, it might be more useful to
  - Bound the number of roles, and
  - Minimize the approximation itself

Example: 0-consistency

In fact, you can not decompose $U_P$ into < 3 roles.
Example: 2-consistency

\[
\begin{array}{c|c|c|c|c|c|c}
U1 & P1 & P2 & P3 & P4 & P5 & U1 \\hline
0 & 1 & 0 & 0 & 1 & & \times \\
1 & 1 & 1 & 0 & 1 & & \\
1 & 1 & 0 & 1 & 1 & & \\
1 & 1 & 1 & 0 & 0 & & \\
\end{array}
\]

This is the optimal 2-consistent decomposition
This is the optimal 3-consistent decomposition
This is the optimal 4-consistent decomposition

Example: 5-consistency

\[
\begin{array}{c|c|c|c|c|c|c}
U1 & P1 & P2 & P3 & P4 & P5 & U1 \\hline
0 & 1 & 0 & 0 & 1 & & \times \\
1 & 1 & 1 & 0 & 1 & & \\
1 & 1 & 0 & 1 & 1 & & \\
1 & 1 & 1 & 0 & 1 & & \\
\end{array}
\]

This is the optimal 5-consistent decomposition

Example: 2-consistency but for \( k=2 \), it's optimal MinNoise RMP

\[
\begin{array}{c|c|c|c|c|c|c}
U1 & P1 & P2 & P3 & P4 & PA & UA \\hline
0 & 1 & 0 & 0 & 1 & P1 & R1 \\
1 & 1 & 1 & 0 & 1 & P2 & R2 \\
1 & 1 & 0 & 1 & 1 & P3 & R1 \\
1 & 1 & 1 & 0 & 0 & P4 & R1 \\
\end{array}
\]

Example of MinNoise RMP with \( k=2 \)

\[
\begin{array}{c|c|c|c|c|c|c}
U1 & P1 & P2 & PA & UA & R1 & R2 \\hline
0 & 1 & 0 & 0 & P1 & R1 & R1 \\
1 & 1 & 1 & 0 & P2 & R2 & R1 \\
1 & 1 & 0 & 1 & P3 & R1 & R1 \\
1 & 1 & 1 & 0 & P4 & R1 & R1 \\
\end{array}
\]

But it's optimal is:

\[
\begin{array}{c|c|c|c|c|c|c}
U1 & P1 & P2 & PA & UA & R1 & R2 \\hline
0 & 1 & 0 & 0 & P1 & R1 & R1 \\
1 & 1 & 1 & 0 & P2 & R2 & R1 \\
1 & 1 & 0 & 1 & P3 & R1 & R1 \\
1 & 1 & 1 & 0 & P4 & R1 & R1 \\
\end{array}
\]

Example: MinNoise RMP at \( k=2 \)

\[
\begin{array}{c|c|c|c|c|c|c}
U1 & P1 & P2 & P3 & P4 & PA & UA \\hline
0 & 1 & 0 & 0 & 1 & P1 & R1 \\
1 & 1 & 1 & 0 & 1 & P2 & R2 \\
1 & 1 & 0 & 1 & 1 & P3 & R1 \\
1 & 1 & 1 & 0 & 0 & P4 & R1 \\
\end{array}
\]

Refinement of MinNoise RMP at \( k=2 \)

\[
\begin{array}{c|c|c|c|c|c|c}
U1 & P1 & P2 & P3 & P4 & PA & UA \\hline
0 & 1 & 0 & 0 & 1 & P1 & R1 \\
1 & 1 & 1 & 0 & 1 & P2 & R2 \\
1 & 1 & 0 & 1 & 1 & P3 & R1 \\
1 & 1 & 1 & 0 & 0 & P4 & R1 \\
\end{array}
\]

Constraint: Do not flip 0's to 1's in UPA.
Overview

- Top-Down Engineering
- Bottom-Up Engineering
  - The Role Mining Problem
  - ORCA
  - FastMiner
  - Tiling

ORCA (SS ‘05)

- PRMS: permission set
- U: user set
- cluster c:
  - rights(c): permissions for c
  - members(c): users in c

Let Order be a partial order on the set of clusters
(i.e., Order ⊆ Clusters × Clusters)
We’ll use x < y for strict ordering.

ORCA Example

1. Clusters ← ∅; Order ← ∅
2. For each single permission p, define clusters c_p as
   rights(c_p) ← {p}
   members(c_p) ← {users with permission p}
3. Add new clusters c to set of all clusters
   Clusters ← Clusters ∪ {c}
4. Repeat until Clusters is stable
   4.1. Order ← {all non-subordinate elements of Clusters}
   4.2. For each cluster pair in Order, <c_d, c_p>:
       rights(c_d) ← rights(c_d) ∪ rights(c_p)
       members(c_d) ← members(c_d) ∪ members(c_p)
   4.3. w = max{|rights(<c_d, c_p>)|: |members(<c_d, c_p>)|}
   4.4. S = {<c_d, c_p>: |members(<c_d, c_p>)| = w and c_d are members of Order}
   4.5. r = max{|rights(<c_d, c_p>)|: <c_d, c_p> is an element of S}
   4.6. S = {<c_d, c_p>: |rights(<c_d, c_p>)| > r and <c_d, c_p> is an element of S}
   4.7. New cluster c, based on c_p: random(F)
       rights(c) ← rights(c) ∪ rights(c_p)
       members(c) ← members(c) ∪ members(c_p)
   4.8. Add c as a super-cluster to x and y
       Order ← Order ∪ {<c_x, c_y>}

Random Selection: $c_{12}$

Rights: $P_1, P_2$

Members: $A, B, E, F$

Order $C_1, C_2, C_3, C_4, C_5$

<table>
<thead>
<tr>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$C_4$</th>
<th>$C_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>$P_1$</td>
<td>$B$</td>
<td>$P_2$</td>
<td>$P_2$</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$A$</td>
<td>$P_3$</td>
<td>$B$</td>
<td>$P_4$</td>
</tr>
<tr>
<td>$P_3$</td>
<td>$A$</td>
<td>$P_4$</td>
<td>$C_1$</td>
<td>$F$</td>
</tr>
<tr>
<td>$P_4$</td>
<td>$A$</td>
<td>$P_5$</td>
<td>$C_1$</td>
<td>$F$</td>
</tr>
<tr>
<td>$P_5$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
</tbody>
</table>

Ann 1 1 0 1 0
Bob 1 1 0 1 0
Carl 1 0 1 0 1
Dan 1 0 1 0 1
Ed 1 1 0 1 0
Fay 1 1 1 1 1

© 2019 Bradley Malin
ORCA to Roles

One user per role is unlikely

ORCA to Roles

All users in 1 role!
ORCA Limitations

- Does not allow a permission to belong to more than one role (except as part of hierarchy)
- Roles are highly-dependent upon order
- Chooses clusters based on maximal membership

Roles via Subset Enumeration

- Identify initial set of roles
  - Group all users having same set of permissions
- Compute all possible intersection sets
- For each candidate role, count number of users having that role
- Reorder candidate roles based on [pick your favorite] metric

Example of Complete Miner

<table>
<thead>
<tr>
<th>User</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>U3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>U5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>U7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>U8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>U9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>U10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>U11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U13</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U14</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>U15</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Fast Miner

- Limit the intersection to only pairs
- Limited to \( n^2 \) intersections
- Downside – scalability...

Experimental Validation & Results

- Setup
  - Randomized test data generation
  - Run FastMiner on generated data
  - Measure accuracy (how many of the original roles found?)
  - Measure speed
  - Results averaged over multiple (5) runs

- Results
  - As long as number of users is at least 10 times the number of roles, accuracy is good
  - As long as the number of rights is at least 10 times the number of roles, accuracy is good
  - The algorithm works well even if users can have up to 10 roles at the same time

Overview

- Top-Down Engineering
- Bottom-Up Engineering
  - The Role Mining Problem (RMP)
  - ORCA
  - FastMiner
  - Tiling
  - Minimum Perturbataion RMP

RMP vs. Tiling*

- Tile \( r \) is a set of rows and columns all with a value of 1
- Area (or cardinality) of tile is the number of 1’s
- Tiling is a collection of tiles, which may overlap

**Big Picture of Tiling**

- User-Permission Relation

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**User-Permission Relation**

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- tiling

**User-Permission Relation**

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**RMP vs. Tiling**

- Tile \( t \) is a set of rows and columns all with a value of 1
- Area (or cardinality) of tile is the number of 1’s
- Tiling is a collection of tiles, which may overlap

- Minimum Tiling Problem (MTP)
  - Given a Boolean matrix
  - Find a tiling with
    - area equal to the total # of 1’s AND
    - the least number of tiles.

- Dual: Each tile is a role!

**Greedy Approximation Algorithm**

- Algorithm depends on finding all maximal tiles with area over a pre-specified threshold
- Depth-first strategy is applied to find the tiles
- Algorithm returns an approximation on the order of \( O(\log mn) \)

**The Algorithm**

- Two phases
  - Phase 1: Find minimum tiling of UPA
    - Strategy: Add the largest uncovered tile to current tiling until UPA is covered
  - Phase 2: Convert tiles into R, UA, PA

---

*J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: finding a minimal descriptive set of roles. Proc ACM SACMAT. 2007:175-184*
Formally

1. \( T \leftarrow \{ \} \)
2. While (\( T' \leftarrow \text{LUTM}(\text{UPA}, T) \neq \emptyset \))
   2.1. \( T \leftarrow T \cup T' \)
3. For each tile \( t \in T \)
   3.1. Create a new role \( r \) in \( R \)
   3.2. Extract permissions from \( t \) and add permissions to \( r \) in \( PA \)
   3.3. Extract users from \( t \) and add users to \( r \) in \( UA \)

Phase 1: Cover

Phase 2: Decompose

LUTM

- Assume an order over permissions
- LT: Largest Uncovered Tile
- Step 1: LT & areaLT

LUTM

1. \( P \leftarrow \{ \} \)
2. \( LT \leftarrow \{ \}, \text{AreaLT} \leftarrow 0 \)
3. For each \( p \) in \( P \)
   3.1. if uncovered area of \( t(P \cup \{p\}) > \text{AreaLT} \)
      3.1.1. \( LT \leftarrow t(P \cup \{p\}) \)
      3.1.2. Update \( \text{AreaLT} \) to have uncovered area of \( t(P \cup \{p\}) \)
   3.2. \( \text{UPA}(P \cup \{p\}) \leftarrow \{ \} \)
   3.3. for each \( q \) in \( P \), such that \( q > P \)
      3.3.1. append \( (q, U(\{p\}) \cap U(\{q\})) \) to \( \text{UPA}(P \cup \{p\}) \)
   3.4. compute \( T((P \cup \{p\}, \text{UPA}(P \cup \{p\})) \) recursively

Initialization: \( P \) will be the permission set.
LT: Will be largest uncovered tile (pointer)
AreaLT: The area of the tile

3.1.2. Update AreaLT to have uncovered area of \( t(P \cup \{p\}) \)
3.2. \( \text{UPA}(P \cup \{p\}) \leftarrow \{ \} \)
3.3. for each \( q \) in \( P \), such that \( q > P \)
      3.3.1. Add \( (q, U(\{p\}) \cap U(\{q\})) \) to \( \text{UPA}(P \cup \{p\}) \)
      3.4. compute \( T((P \cup \{p\}, \text{UPA}(P \cup \{p\})) \) recursively

For each permission, check if the uncovered area* of the current tile is larger than the best known tile so far.

*The uncovered area is # of 1’s covered by current tile that are not in existing tiling.
LUTM

1. \( P \leftarrow \{\} \)

Creates a conditional database \( U_P \)

2. for each \( p \) in \( PRMS \)
   3.1. if uncovered area of \( t(P \cup \{p\}) > AreaLT \)
       3.1.1. \( LT \leftarrow t(P \cup \{p\}) \)
       3.1.2. Update \( AreaLT \) to have uncovered area of \( t(P \cup \{p\}) \)
   3.2. \( UPA(P \cup \{p\}) \leftarrow \{\} \)
   3.3. for each \( q \) in \( PRMS \), such that \( q > p \)
       3.3.1. append \((q, U\{p\} \cap U\{q\})\) to \( UPA \)
   3.4. compute \( T((P \cup \{p\}, UPA(P \cup \{p\})) \) recursively

LUTM

1. \( P \leftarrow \{\} \)
2. \( LT \leftarrow \{\}, AreaLT \leftarrow 0 \)
3. For each \( p \) in \( PRMS \)
   3.1. if uncovered area of \( t(P \cup \{p\}) > AreaLT \)
       3.1.1. \( LT \leftarrow t(P \cup \{p\}) \)
       3.1.2. Update \( AreaLT \) to have uncovered area of \( t(P \cup \{p\}) \)
   3.2. \( UPA(P \cup \{p\}) \leftarrow \{\} \)
   3.3. for each \( q \) in \( PRMS \), such that \( q > p \)
       3.3.1. append \((q, U\{p\} \cap U\{q\})\) to \( UPA \)
   3.4. compute \( T((P \cup \{p\}, UPA(P \cup \{p\})) \) recursively

Acknowledgements

- Some of the following slides were adapted from Jaideep Vaidya’s (Rutgers U.) tutorial on role engineering and role mining

Readings for Next Lecture

Required

Optional