TIIME

# MidPoint Working Group at TIIME 2024

## Parametric and Dynamic Roles

**Radovan Semančík**
Evolveum’s Co-Founder and Software Architect

# Traditional RBAC

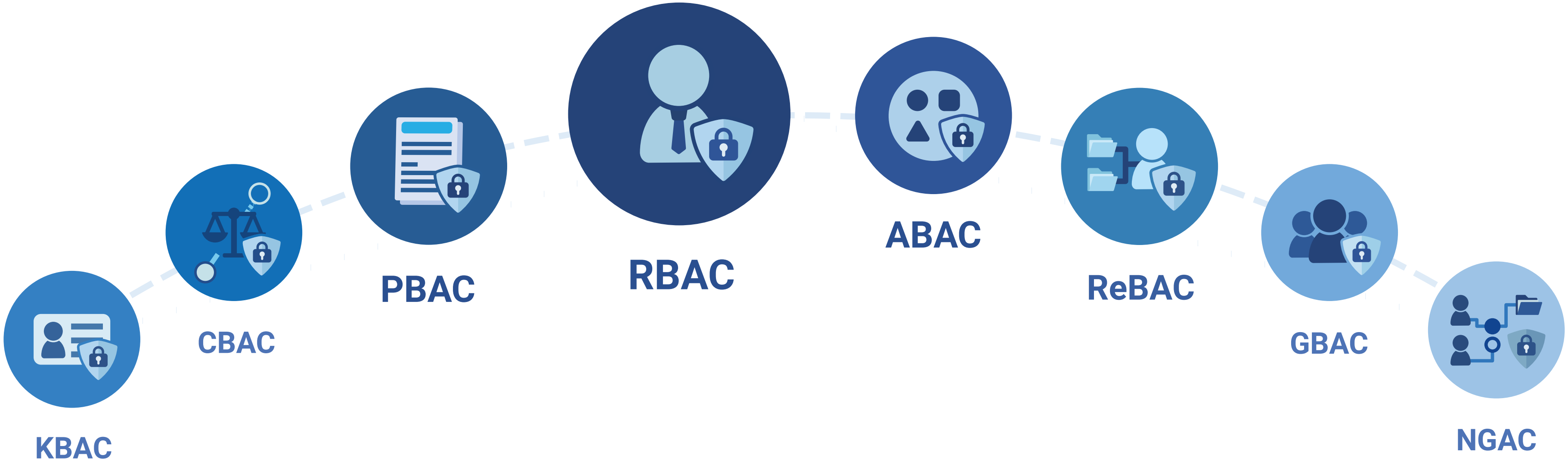
- The king of access control
- NIST (ANSI/INCITS 359-2004, INCITS 359-2012)
- Already outdated
- Static – no policy
- It does not work

# What Is The Problem With Traditional RBAC?

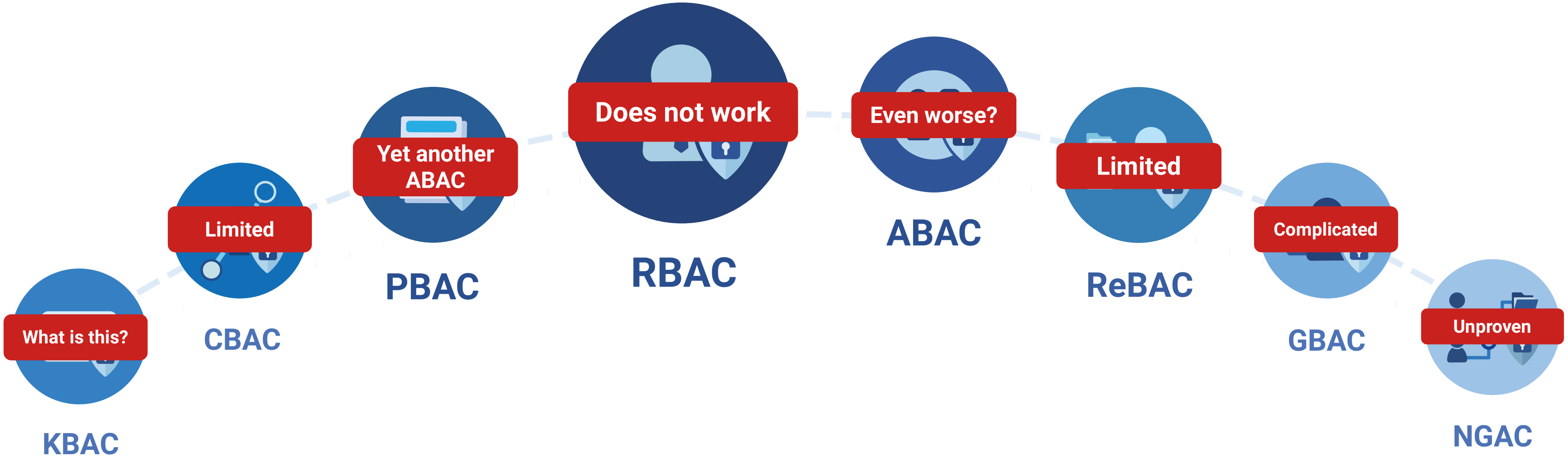
- Overuse of application roles
- Access request frenzy
- Role explosion
- Huge certification effort
- Business role duplication
- Role decay

**Root of all the problems:**
***static* role assignments**

# Access Control ZOO



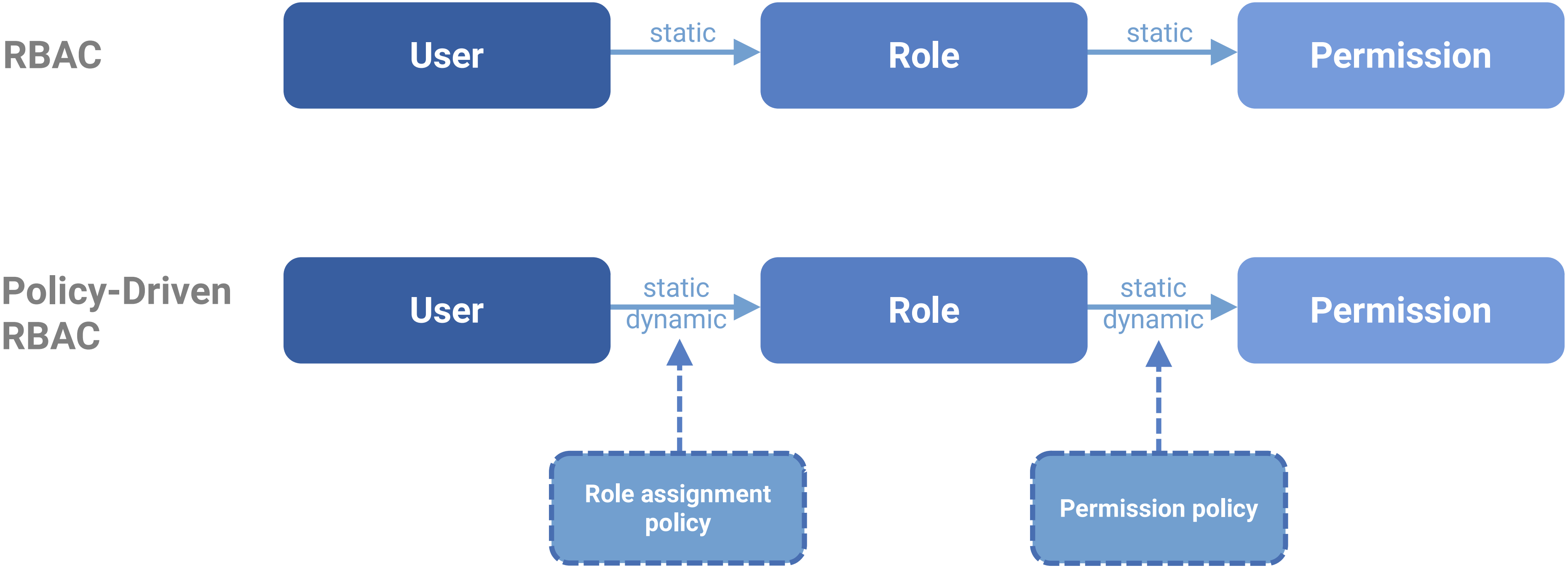
# Access Control Is Not Easy



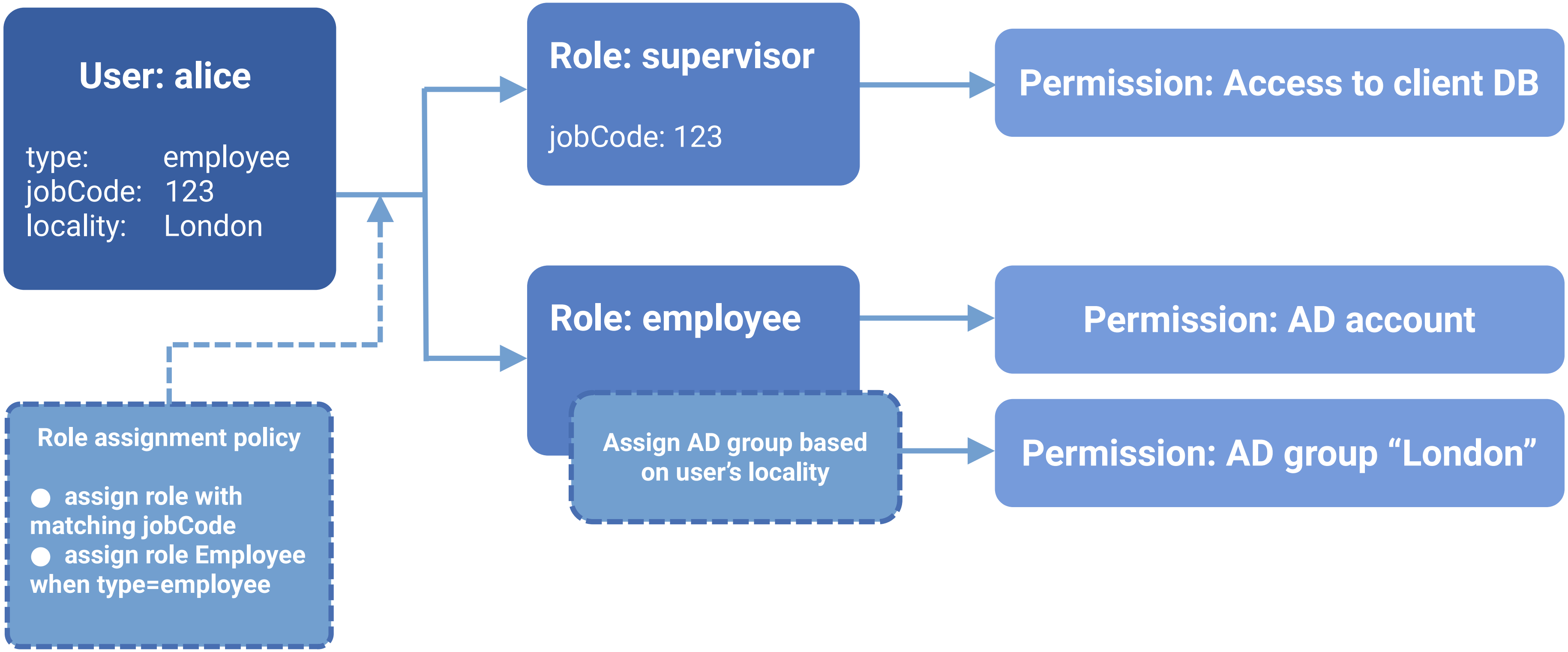
# Our Approach

- RBAC has some good parts
- RBAC is not going away anytime soon
- **Dynamic RBAC**: policy in the roles
- A bottom-up approach: from roles to policy
- AI-assisted mechanisms
- Long-term sustainability

# Policy-Driven RBAC Principle



# Policy-Driven RBAC Example (Theory)



User: alice

type: employee
jobCode: 123
locality: London

Role: supervisor

jobCode: 123

Permission: Access to client DB

Role: employee

Permission: AD account

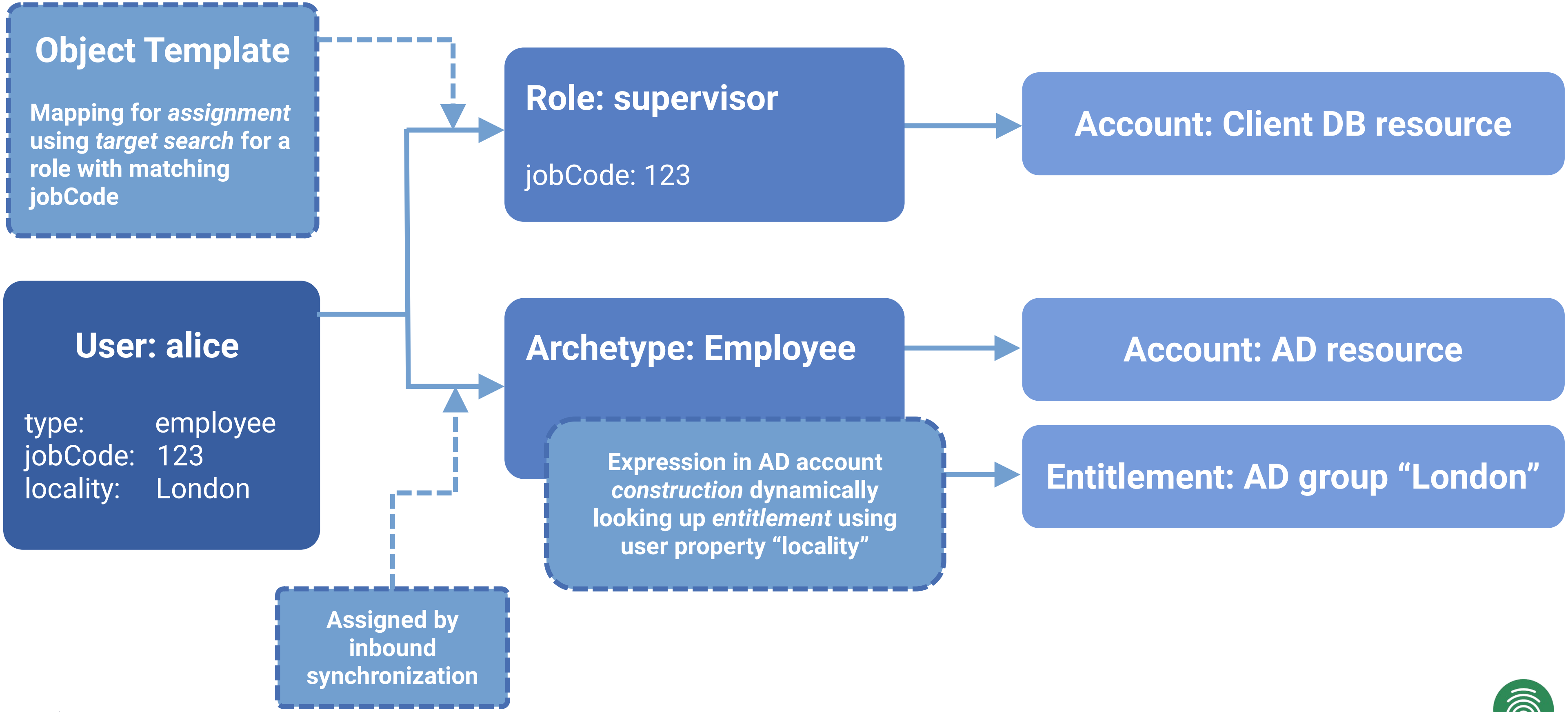
Assign AD group based on user's locality

Permission: AD group "London"

Role assignment policy

- assign role with matching jobCode
- assign role Employee when type=employee

# Policy-Driven RBAC Example (Implementation)



Object Template

Mapping for *assignment* using *target search* for a role with matching jobCode

User: alice

type: employee
jobCode: 123
locality: London

Assigned by inbound synchronization

Role: supervisor

jobCode: 123

Archetype: Employee

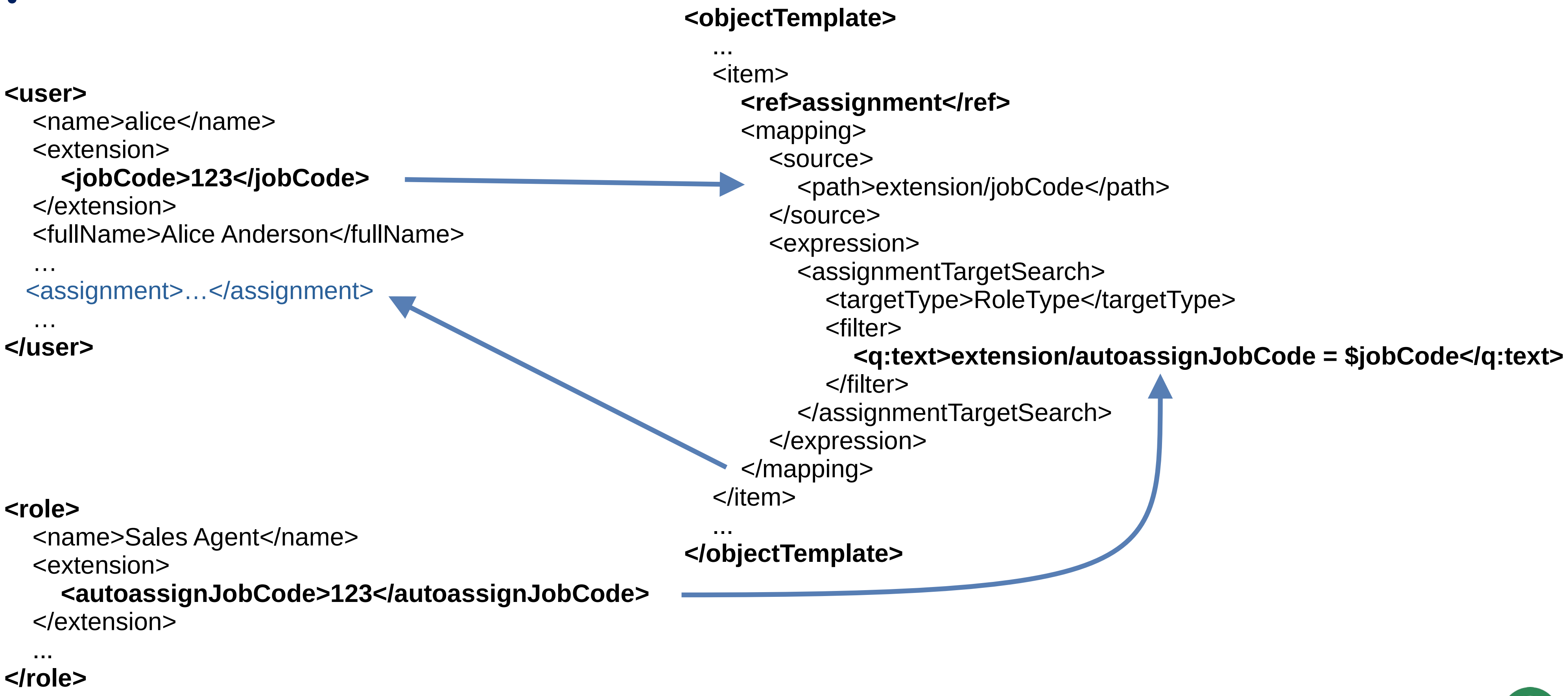
Expression in AD account *construction* dynamically looking up *entitlement* using user property “locality”

Account: Client DB resource

Account: AD resource

Entitlement: AD group “London”

# Solution Part 1: Job Code



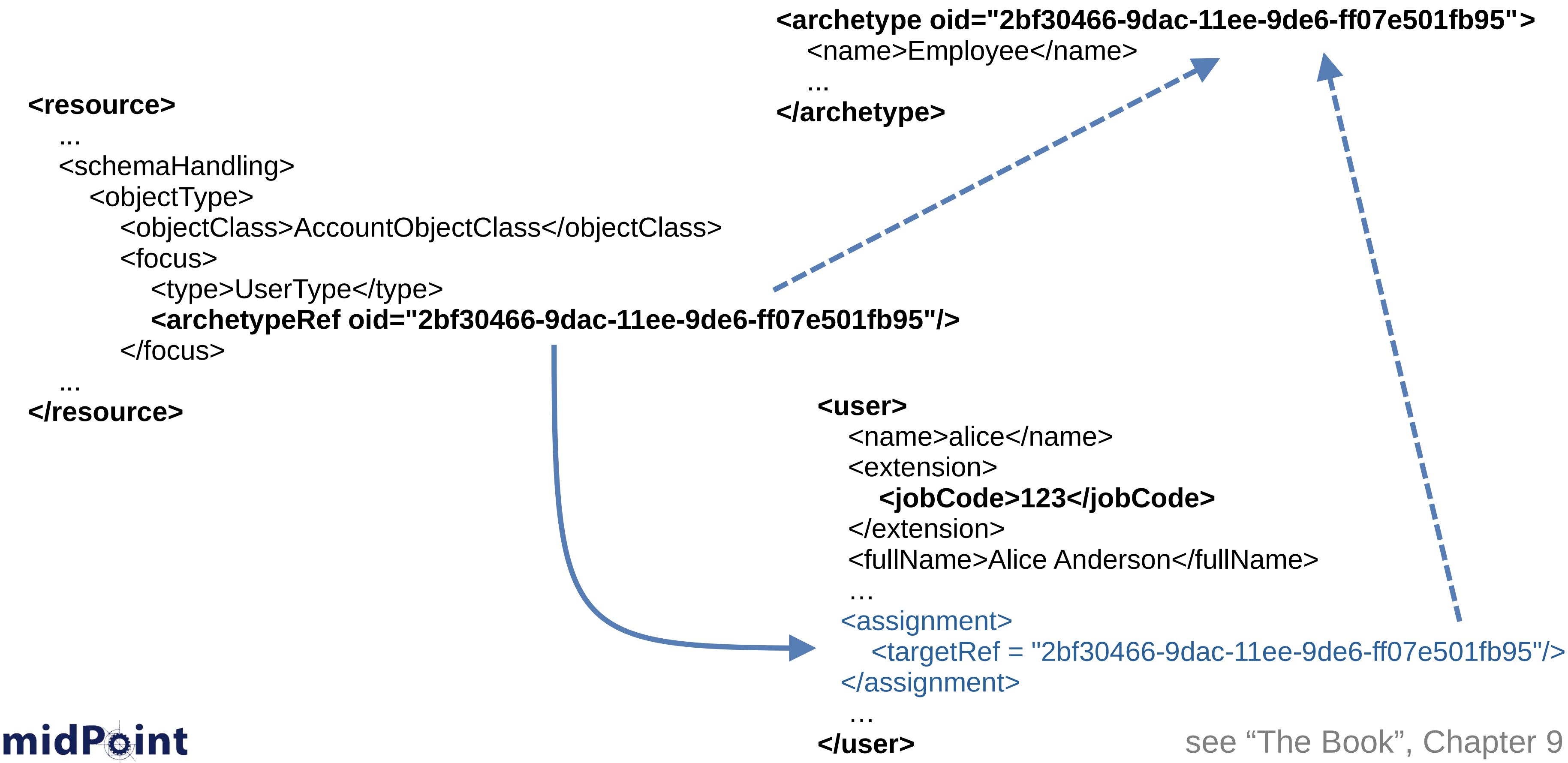
```
<user>
    <name>alice</name>
    <extension>
        <jobCode>123</jobCode>
    </extension>
    <fullName>Alice Anderson</fullName>
    …
   <assignment>…</assignment>
    …
</user>
```

```
<role>
    <name>Sales Agent</name>
    <extension>
        <autoassignJobCode>123</autoassignJobCode>
    </extension>
    …
</role>
```

```
<objectTemplate>
    …
    <item>
        <ref>assignment</ref>
        <mapping>
            <source>
                <path>extension/jobCode</path>
            </source>
            <expression>
                <assignmentTargetSearch>
                    <targetType>RoleType</targetType>
                    <filter>
                        <q:text>extension/autoassignJobCode = $jobCode</q:text>
                    </filter>
                </assignmentTargetSearch>
            </expression>
        </mapping>
    </item>
    …
</objectTemplate>
```

see “The Book”, Chapter 8

# Solution Part 2: Employee Archetype



```
<archetype oid="2bf30466-9dac-11ee-9de6-ff07e501fb95">
    <name>Employee</name>
    ...
</archetype>
```

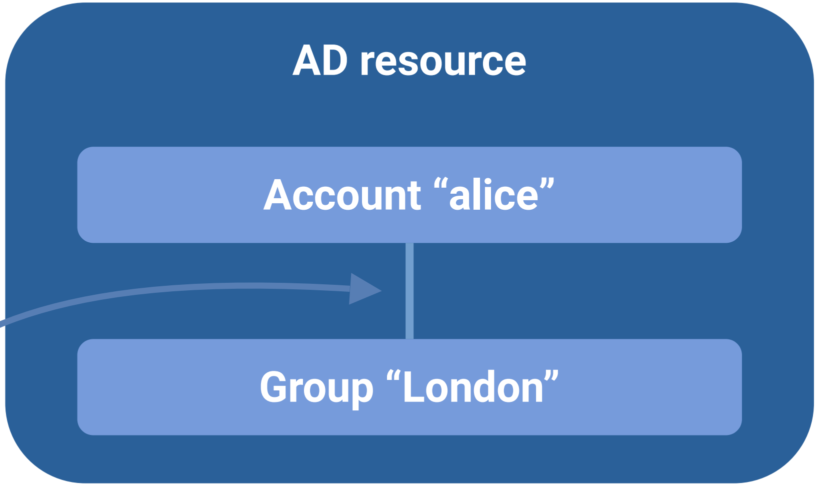
```
<resource>
    ...
    <schemaHandling>
        <objectType>
            <objectClass>AccountObjectClass</objectClass>
            <focus>
                <type>UserType</type>
                <archetypeRef oid="2bf30466-9dac-11ee-9de6-ff07e501fb95"/>
            </focus>
    ...
</resource>
```

```
<user>
    <name>alice</name>
    <extension>
        <jobCode>123</jobCode>
    </extension>
    <fullName>Alice Anderson</fullName>
    …
   <assignment>
       <targetRef = "2bf30466-9dac-11ee-9de6-ff07e501fb95"/>
   </assignment>
    …
</user>
```

see “The Book”, Chapter 9

# Solution Part 3: Locality Expression

```
<archetype oid="2bf30466-9dac-11ee-9de6-ff07e501fb95">
    <name>Employee</name>
    …
     <inducement>
        <construction>
            <resourceRef oid="… AD Resource ..."/>
            <kind>account</kind>
            <association>
                <ref>ri:group</ref>
                <outbound>
                    <source>
                        <path>$focus/locality</path>
                    </source>
                    <expression>
                        <associationTargetSearch>
                            <filter>
                                <q:text>cn = $locality</q:text>
                            </filter>
                        </associationTargetSearch>
                    </expression>
    ...
</archetype>
```



```
<user>
    <name>alice</name>
    <fullName>Alice Anderson</fullName>
    <locality>London</localilty>
    <assignment>
        <targetRef = "2bf30466-9dac-11ee-9de6-ff07e501fb95"/>
    </assignment>
    …
</user>
```

see “Assignment Configuration” in docs

# Role Autoassign

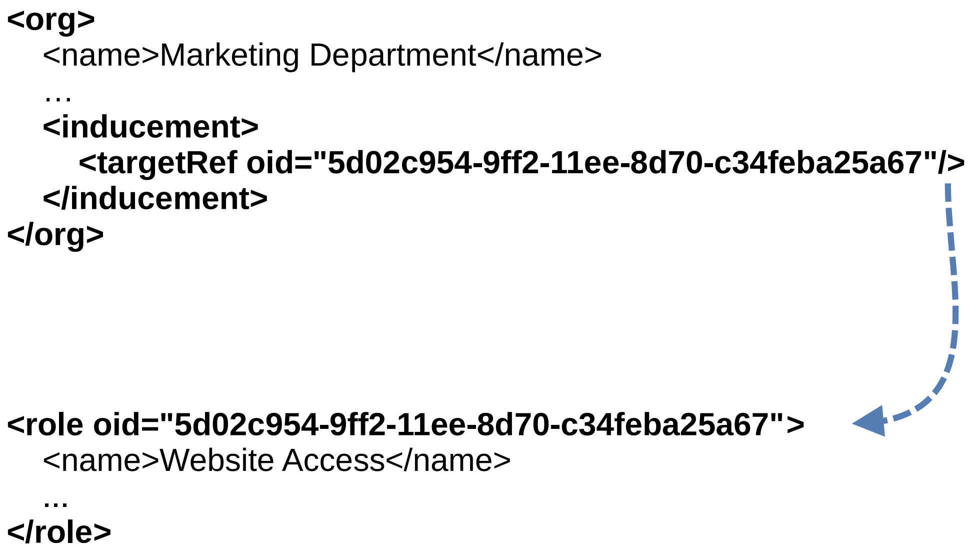
- Simplest form of role assignment policy
- Nice encapsulation
- Have no fear, it can scale reasonably
- Go for it!
- More improvements in 4.9 (metadata, maybe GUI)

```
<role>
     <name>Cook</name>
    ...
    <autoassign>
        <enabled>true</enabled>
        <focus>
            <mapping>
                <source>
                    <path>locality</path>
                </source>
                <condition>
                    <script>
                        <code>
                            locality?.norm == 'kitchen'
                        </code>
                    </script>
                </condition>
            </mapping>
        </focus>
    </autoassign>
</role>
```

see “The Book”, Chapter 8

# The Humble Inducement

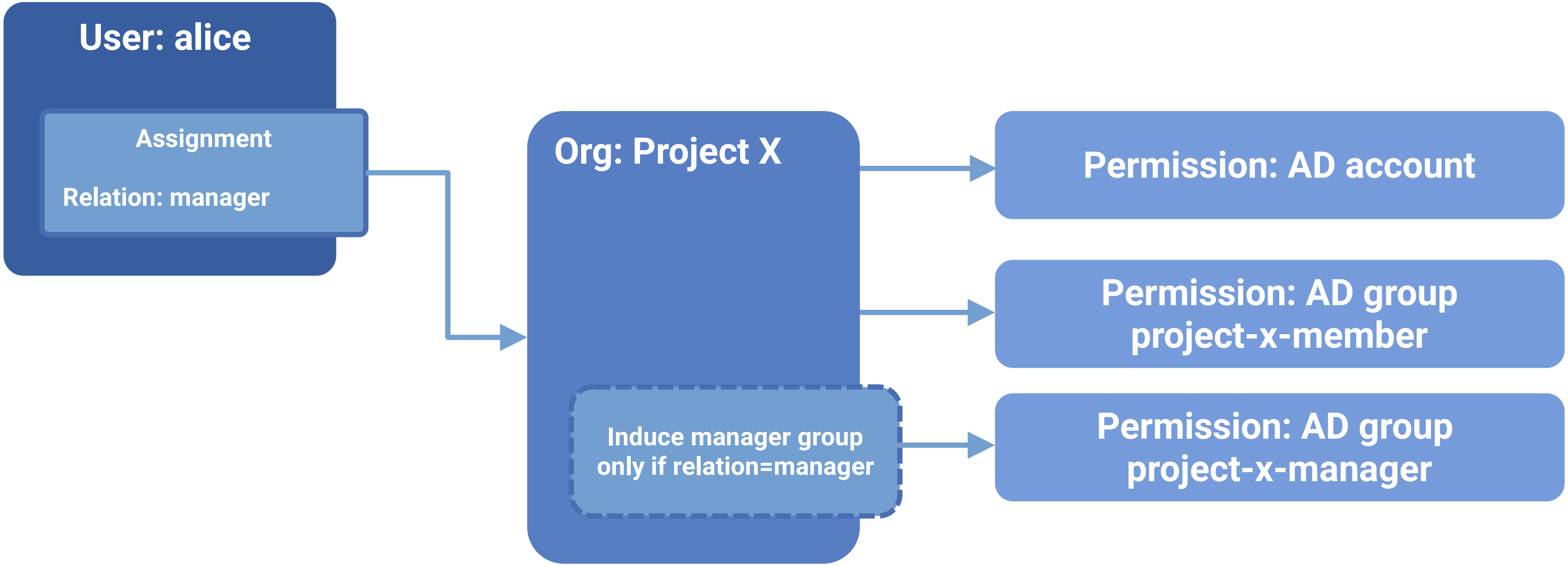
- Inducement of role from org
- Completely automatic
- Extremely simple
- **The best way ever**
- Often overlooked



```
<org>
    <name>Marketing Department</name>
    …
    <inducement>
        <targetRef oid="5d02c954-9ff2-11ee-8d70-c34feba25a67"/>
    </inducement>
</org>


<role oid="5d02c954-9ff2-11ee-8d70-c34feba25a67">
    <name>Website Access</name>
    ...
</role>
```

# Parametric Roles / Orgs



User: alice

Assignment

Relation: manager

Org: Project X

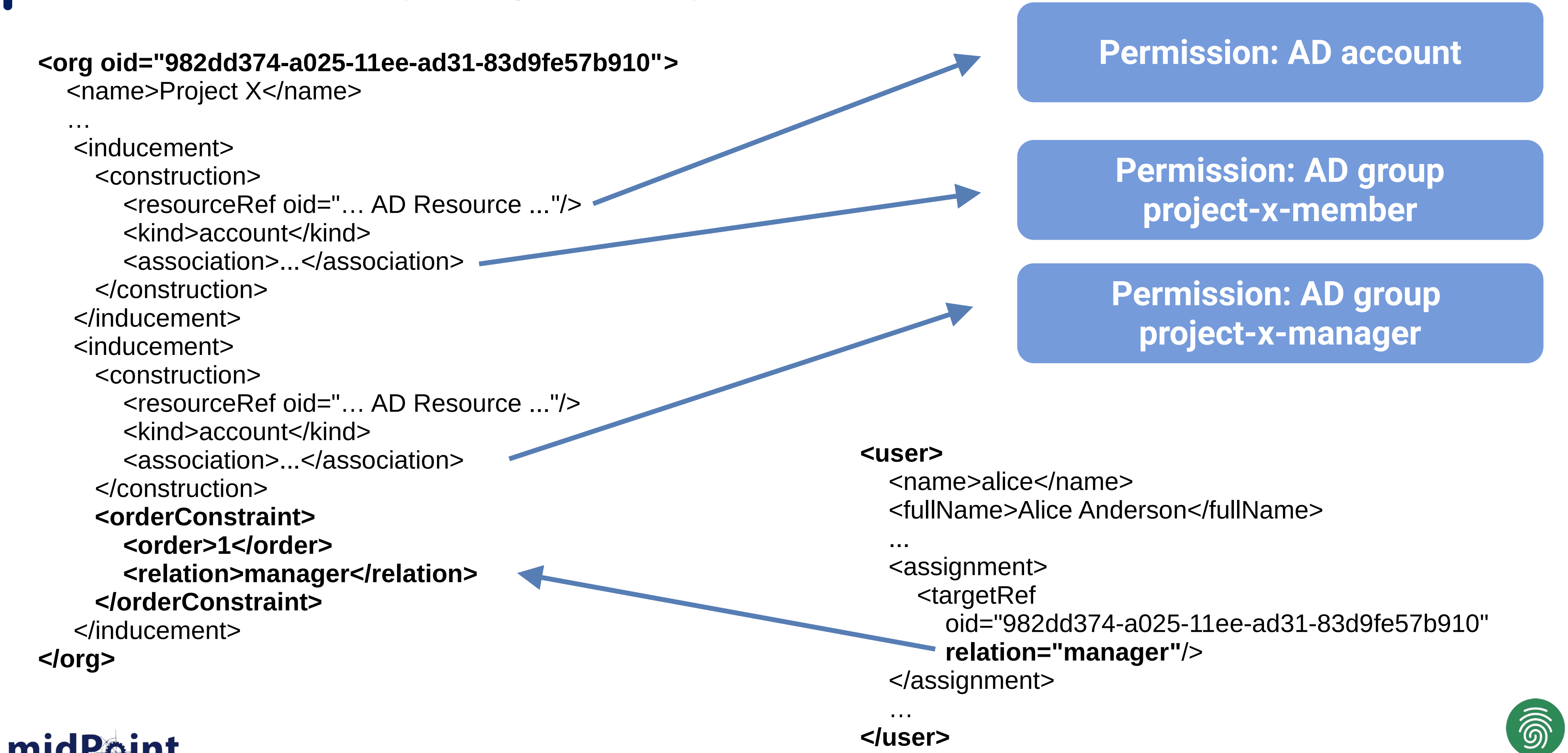
Induce manager group only if relation=manager

Permission: AD account

Permission: AD group project-x-member

Permission: AD group project-x-manager

# Parametric Roles / Orgs: Project Management



```
<org oid="982dd374-a025-11ee-ad31-83d9fe57b910">
    <name>Project X</name>
    …
     <inducement>
        <construction>
            <resourceRef oid="… AD Resource ..."/>
            <kind>account</kind>
            <association>...</association>
        </construction>
     </inducement>
     <inducement>
        <construction>
            <resourceRef oid="… AD Resource ..."/>
            <kind>account</kind>
            <association>...</association>
        </construction>
        <orderConstraint>
            <order>1</order>
            <relation>manager</relation>
        </orderConstraint>
     </inducement>
</org>
```

Permission: AD account

Permission: AD group project-x-member

Permission: AD group project-x-manager

```
<user>
    <name>alice</name>
    <fullName>Alice Anderson</fullName>
    ...
    <assignment>
        <targetRef
            oid="982dd374-a025-11ee-ad31-83d9fe57b910"
            relation="manager"/>
    </assignment>
    …
</user>
```

# Bottom-Up Approach

1) RBAC as usual (access request process)

2) Role mining

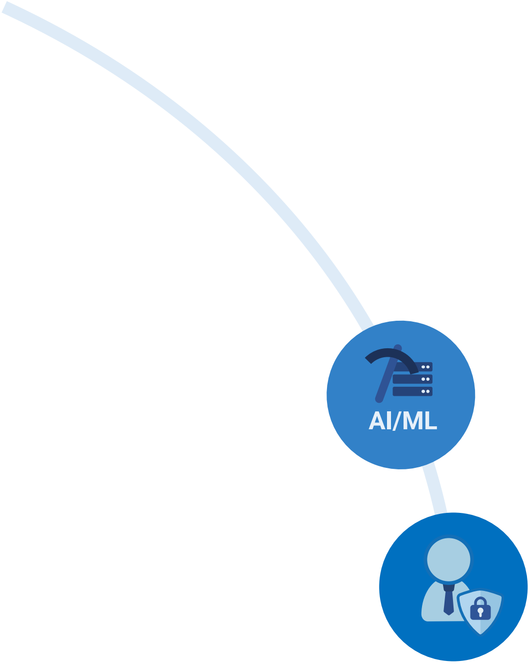
3) Assign roles automatically (inducement/autoassign)

4) Make smarter roles and rules (expressions in roles)

5) Review/decommission old roles

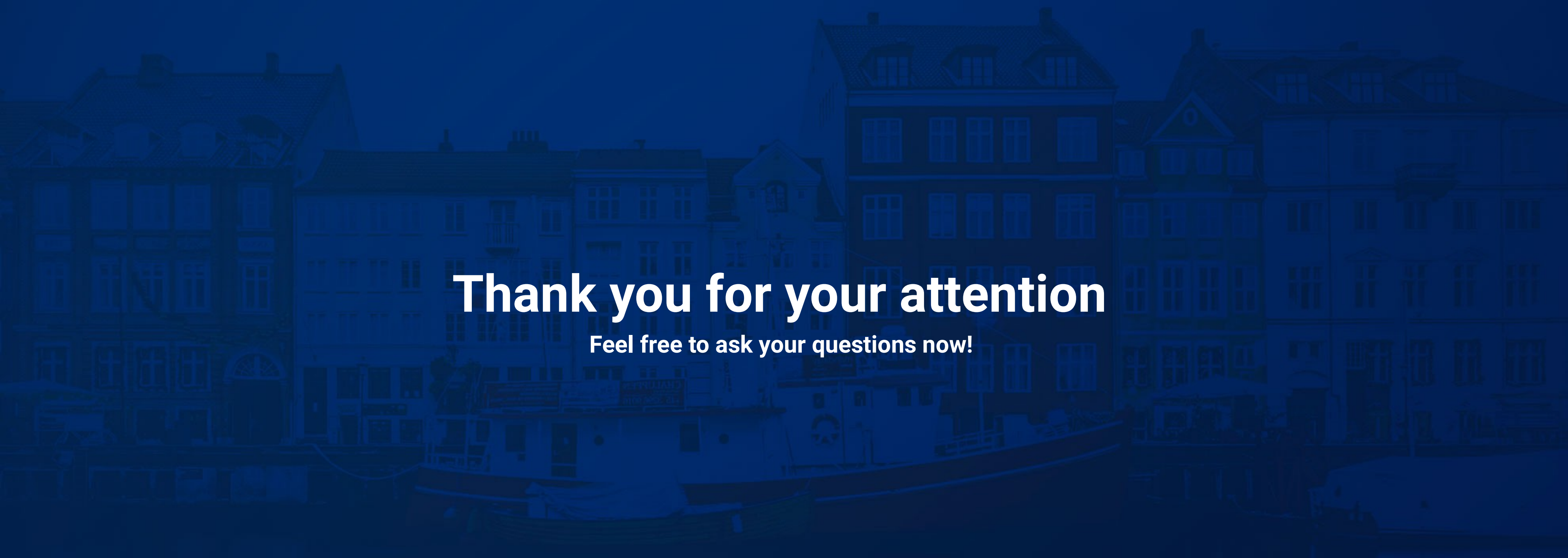
Repeat as necessary

Future: policy mining



# Conclusion

- Policy-Driven RBAC
- Bottom-up approach
- Static and dynamic parts can co-exist
- AI-assisted (role mining, etc.)
- Maintainable and sustainable



# Thank you for your attention

Feel free to ask your questions now!

TIIME

MidPoint Working Group at TIIME 2024