NGAC

- initial ANSI standard 2013, result of NIST projects
- provisional patent
- Github open source reference distribution
- based on standardized and generic set of relations and functions to be reused in policies

a clear description of ideas (for your convenience, examples used directly in this lecture):

https://csrc.nist.gov/publications/detail/sp/800-178/final

NGAC

user, operation, object

instead of

subject, action, resource (XACML)

new components:

- **processes** (ID, memory, descriptors for resource allocations - handles)
- **administratve operations**
- **policy classes**

attributes: for users and objects

objects: refer to data

chown

NGAC Assignments and Associations

assignment: $x \rightarrow y$

meaning:

- x belongs to y
- x and y might be: users, user attributes, objects, object attributes, policy classes
- e.g.
  - `John Smith` $\rightarrow$ `CZD -Hospital-doctors`
  - `CZD-Hospital-doctors` $\rightarrow$ `doctors`
  - `file A` $\rightarrow$ `personal-data`
  - `personal-data` $\rightarrow$ AC-secured-data
- somewhat analogous to roles of users in RBAC

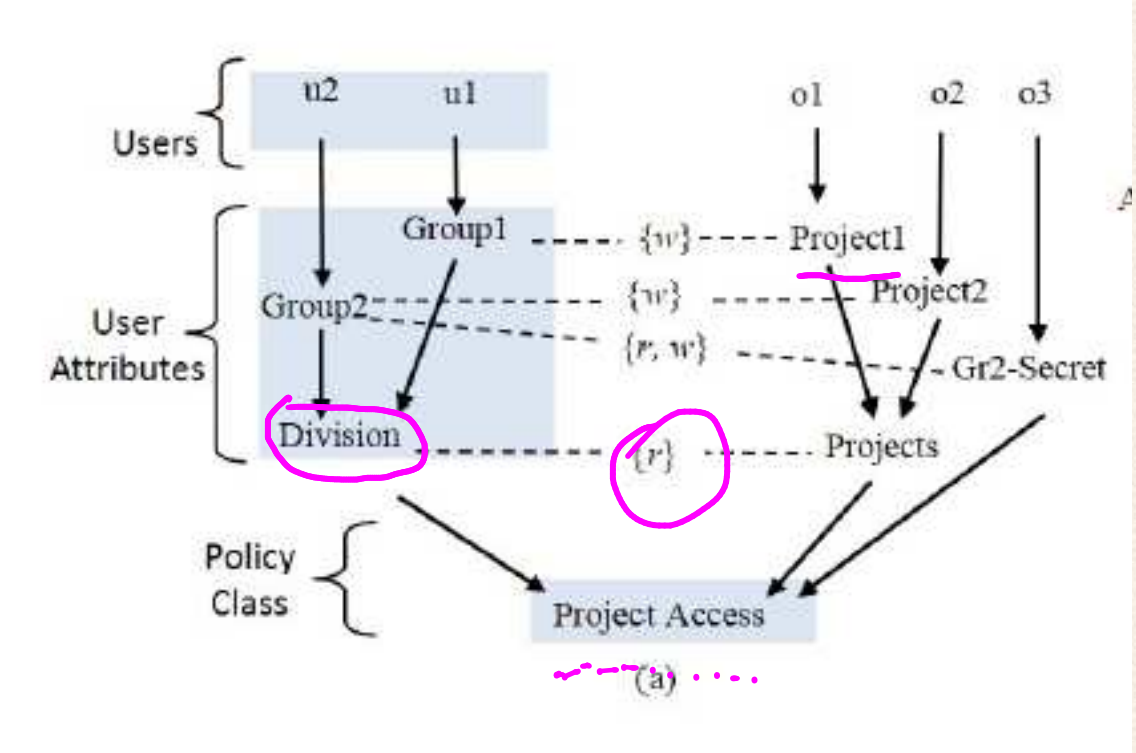
association:

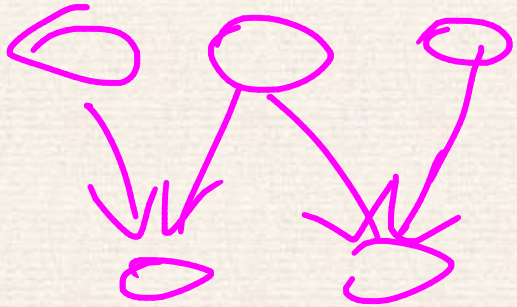
$(\mathrm{ua} \text{ --- } \mathrm{ars} \text{ --- } \mathrm{at})$ where

- ua is a user attribute
- ars is a set of access rights
- at is an attribute (user attribute or object attribute)

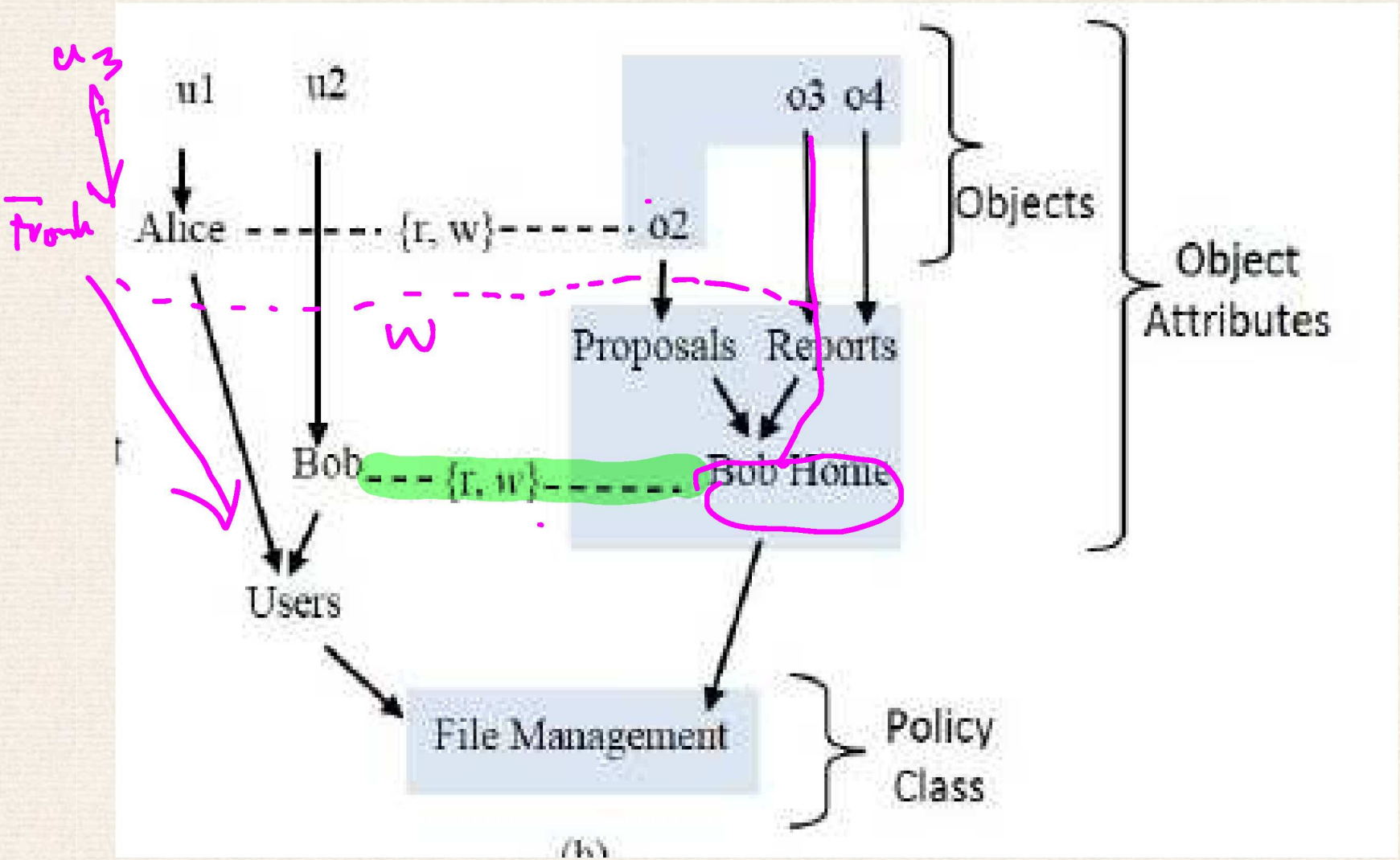
meaning: users in ua have rights in ars on policy elements referenced by at

NGAC example (from the NIST paper)





NGAC example

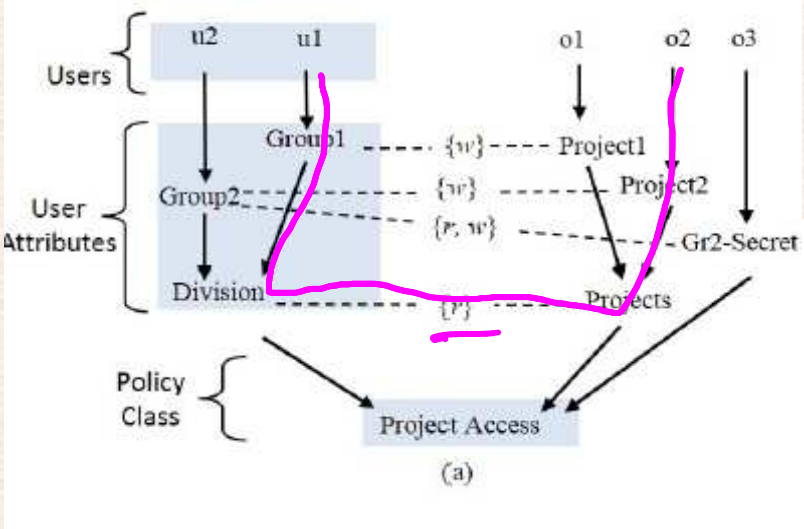

u1
u2
o3
o4
Alice
{r, w}
o2
Objects
Object Attributes
Proposals
Reports
Bob
{r, w}
Bob Home
Users
File Management
Policy Class

NGAC deriving privileges

(u, ar, e) is a privilege following from $(\mathrm{ua} \text{---} \mathrm{ars} \text{---} \mathrm{at})$ iff

- The user u is contained by the user attribute of an association;
- The element e is contained by the attribute at of that association;
- The attribute at of that association is contained by the policy class pc; and
- The access right ar is a member of the access right set of that association.

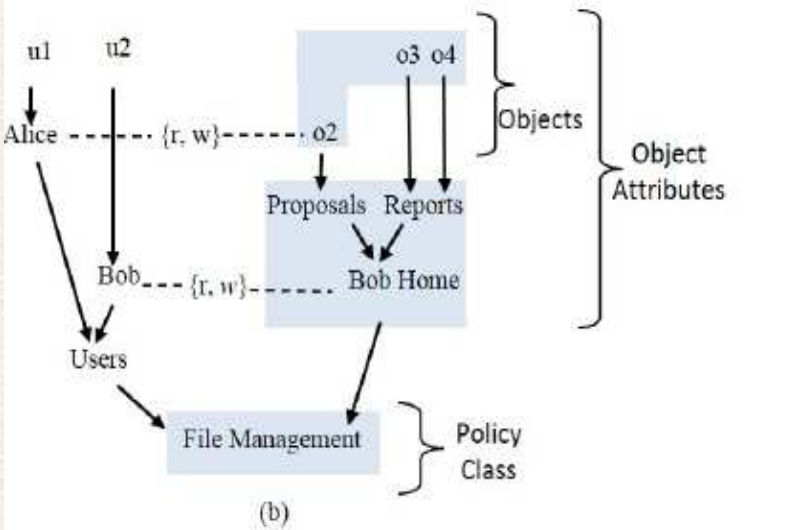
NGAC



permissio

(u1, r, o1), (u1, w, o1), (u1, r, o2), (u2, r, o1),
(u2, r, o2), (u2, w, o2), (u2, r, o3), (u2, w, o3)

NGAC



(u1, r, o2), (u1, w, o2), (u2, r, o2), (u2, w, o2), (u2, r, o3), (u2, w, o3), (u2, r, o4), (u2, w, o4)

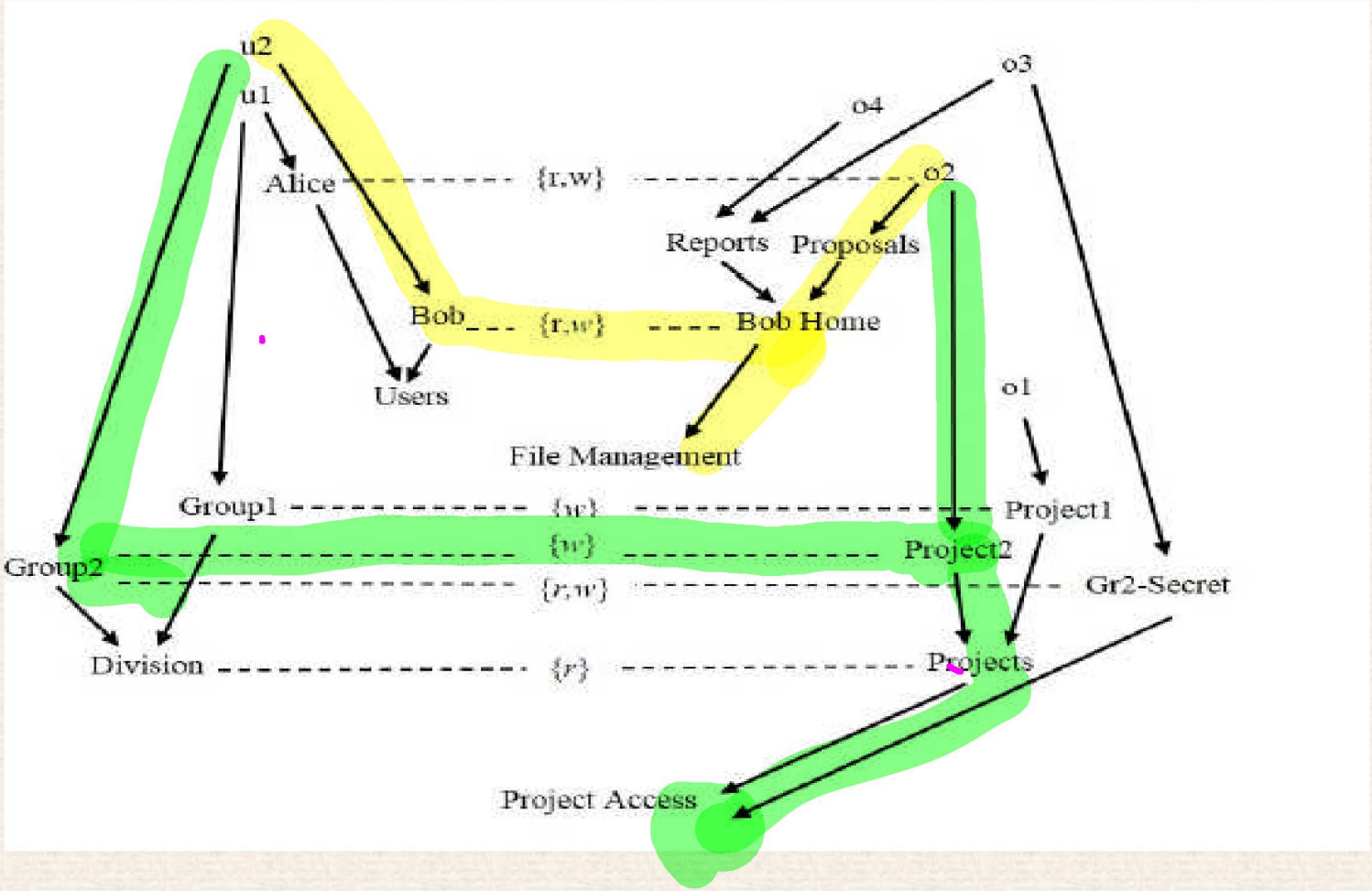
NGAC priviledges - combining policy sets

Rule: the condition mentioned must be

(u, ar, e) is a privilege iff for each policy class

- The user u is contained by the user attribute of an association;
- The element e is contained by the attribute at of that association;
- The attribute at of that association is contained by the policy class pc; and
- The access right ar is a member of the access right set of that association.

NGAC example



(u1, r, o1), (u1, w, o1), (u1, r, o2), (u2, r, o1), (u2, r, o2), (u2, w, o2), (u2, r, o3), (u2, w, o3), (u2, r, o4), (u2, w, o4)

Note:

- `(u1,r,o1)` is a priviledge, since `o1` is only in the "Project Access" class, and it is a priviledge of "Project Access"
- `(u1,w,o2)` is not a priviledge, since `o2` is in both policy classes, it is ok for "File Management" but not for "Project Access"

NGAC prohibitions

- u_deny(u,ars,pe) user deny
- ua_deny(ua,ars,pe) attribute deny
- p_deny(p,ars,pe) process deny

where pe is policy element

example: write right to own tax record by a tax auditor of tax authority

efect: deny overides privileges

NGAC prohibitions with negation

u_deny(u,ars,¬pe) means that user u is denied any element that is not in pe

similarly ua_deny(ua,ars,¬pe), p_deny(p,ars,¬pe)

NGAC obligations

(ep,r) means when ep do r

- ep is called "event pattern"
- r is called "response"
- event is an execution of an operation on an object by a process (in behalf of a user)
- event may specify parameters: user, attributes , operation, attributes of the object, opeartional conditions

Application examples:

handling conflicts of interest (if a process of user A has read a file H, then it must be prevented to read file J)

approval of anothe user

NGAC decision function

control of accesses in terms of processes

- `process_user(p)`
- access requests: `(p,op, argseq)` where
  - → `p` is a process, `op` is an operation, `argseq` are arguments of the operation
- mapping: to access right and policy element pairs involved: {ar,pe}

> A process access request $(p, op, argseq)$ with mapping $(op, argseq) \rightarrow \{(ar, pe)\}$ is granted iff for each (ar_i, pe_i) in $\{(ar, pe)\}$, there exists a privilege (u, ar_i, pe_i) where u = process_user(p), and (ar_i, pe_i) is not denied for either u or p.

NGAC administrative access rights

non-administrative access rights: pertain to activities on protected rosoures

administrative access rights: pertain to activities on policy and attributes

examples:

- creating files, directories
- assigning attributes

NGAC administrative access rights example

adminstration over assignments:

ProjectAccessAdmin --- {create-u-to, delete-u-from, create-ua-to, delete-ua-from, create-uua-from, create-uua-to, delete-uua-from, create-uaua-from, create-uaua-to, delete-uaua-from, delete-uaua-to }---Division

administration over associations:

ProjectAccessAdmin --- {create-assoc-from, delete-assoc-from} --- Division.

ProjectAccessAdmin --- {create-assoc-to, delete-assoc-to, r-allocate, w-allocate} --- Projects.

NGAC delegation of administrative access rights

- initially superuser with empty data elements, attributes, relations, ⟨text-dots⟩ but with all administrative rights
- delegating administrative rights

– administrative access request $\Rightarrow$ administrative routine = set of administrative commands

syntax:

cmdname (x_1: $type_1$, x_2: $type_2$, …, x_k: $type_k$)
…preconditions …
{
$Y' = f(Y, x_1, x_2, \ldots, x_k)$
}

concrete example:

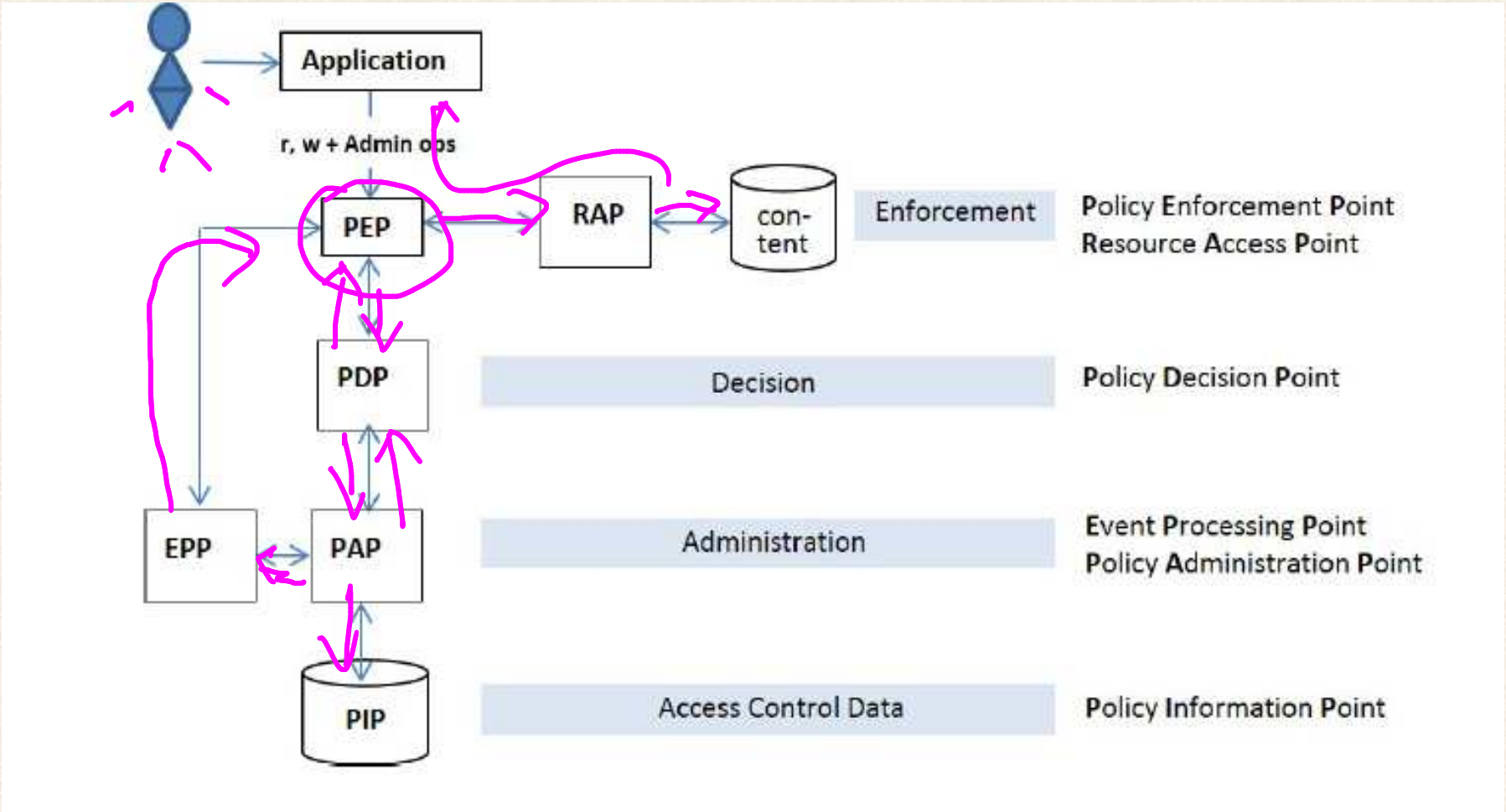
createAssoc (x, y, z)
$x \in UA \wedge y \in ARS \wedge z \in PE \wedge (x, y, z) \notin ASSOC$
{
$ASSOC' = ASSOC \cup \{(x, y, z)\}$
}

NGAC example for routine

```
create-file-mgmt-user(user-id, user-name, user-home) {
     createUAinUA(user-name, Users);
     createUinUA(user-id, user-name);
     createOAinPC(user-home, File Management);
     createAssoc(user-name, {r, w}, user-home);
     createAssoc(user-name, {create-o-to, delete-o-from}, user-home);
     createAssoc(user-name, {create-ooa-from, create-ooa-to,
                             delete-ooa-from, create-oaoa-from,
                             create-oaoa-to, delete-oaoa-from}, user-home);
     createAssoc(user-name, {create-assoc-from, delete-assoc-from}, Users);
     createAssoc(user-name, {create-assoc-to, delete-assoc-to, r-allocate,
                             w-allocate}, user-home);}
```

run create-file-mgmt-user(u1, Bob, Bob-Home) to create Bob's capabilities form the example discussed before

NGAC functional architecture


Application
r, w + Admin ops
PEP
RAP
con-
tent
Enforcement
Policy Enforcement Point
Resource Access Point
PDP
Decision
Policy Decision Point
EPP
PAP
Administration
Event Processing Point
Policy Administration Point
PIP
Access Control Data
Policy Information Point