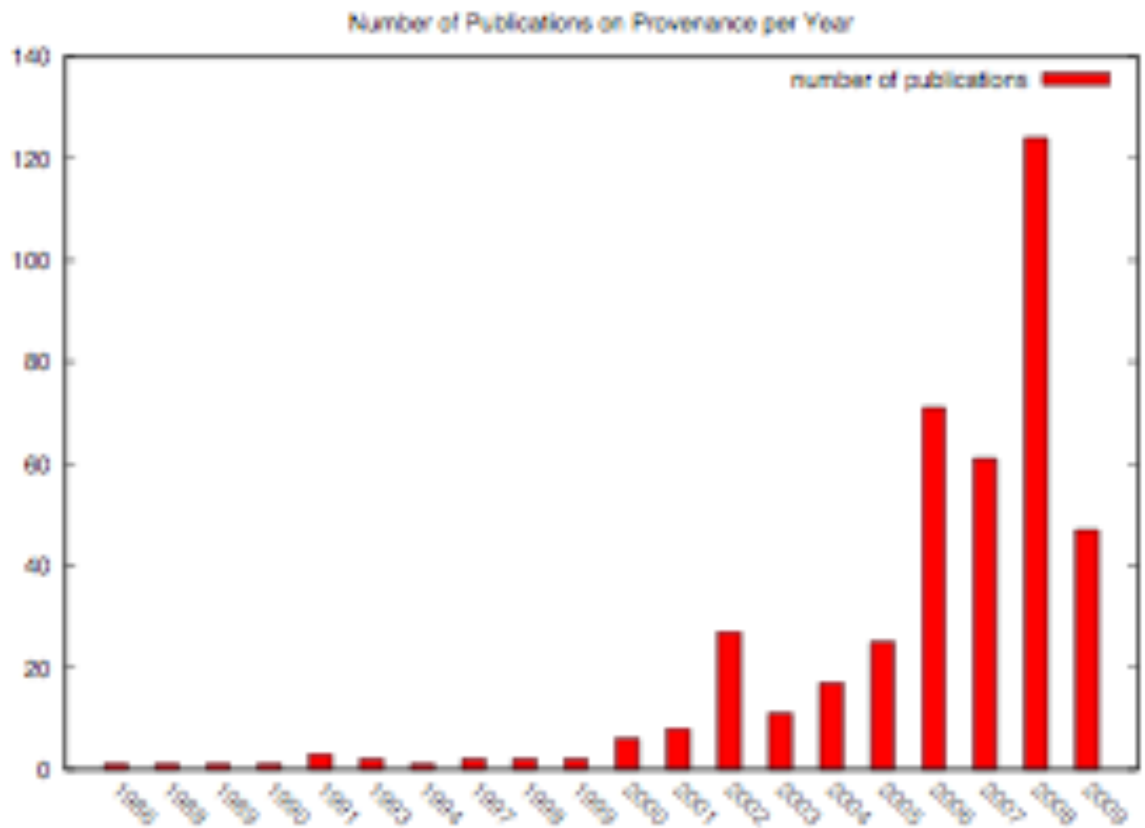
# Secure Data Provenance

April 5, 2013
CS6393 - Research Challenges in Cyber Security
(Slide available at http://drjae.com/Activities.html)

Jae Park
Institute for Cyber Security
University of Texas at San Antonio

# Provenance Publications

- From Luc Moreau, “The Foundation for Provenance on the Web”



# Dictionary Definition of Provenance

- Oxford English Dictionary
  - the fact of coming from some particular source or quarter; origin, derivation.
  - the history or pedigree of a work of art, manuscript, rare book, etc.; concretely, a record of the ultimate derivation and passage of an item through its various owners
- Merriam-Webster Online Dictionary
  - the origin, source
  - the history of ownership of a valued object or work of art or literature

# Data Provenance in Computer Systems

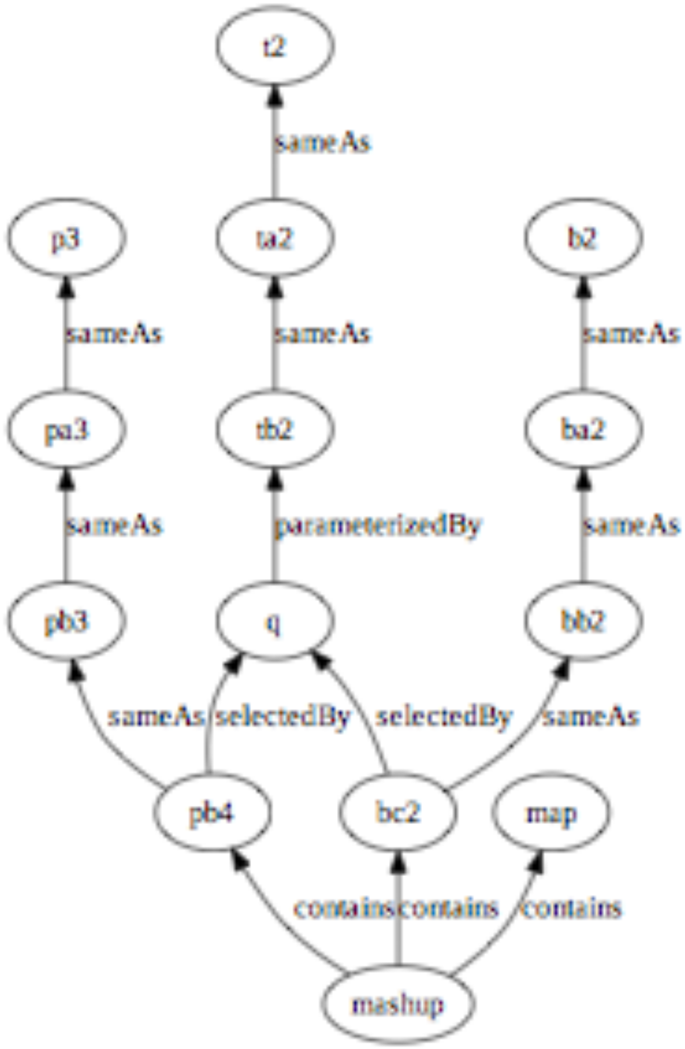
- "The provenance of a piece of data is the process that led to that piece of data"

- "In computer systems, activities are carried out by processes that take input data, input state, input configuration, and produce output data and output state. Such processes are compositional by nature and can be the result of sophisticated compositions (sequential, parallel, conditional, etc) of simpler processes." (Luc Moreau, "The Foundation for Provenance on the Web")

# Other Definition of Provenance

- Why-Provenance
  - identify the set of tuples, whose presence justifies a query result (e.g., {t2,p3,b2})
- Where-Provenance
  - identify where information was copied from
  - E.g., a typo in mashup is came from b2
- How-Provenance
  - a polynomial representation that hints at the structure of the proof explaining how an output tuple is derived
  - E.g., whole graph

# Provenance Example




| | |
|---|---|
| p3 | original photo record |
| t2 | original trail record |
| b2 | original blog record |
| pa3 | copy of p3 in cache |
| ta2 | copy of t2 in cache |
| ba2 | copy of b2 in cache |
| pb3 | copy of pa3 retrieved by q |
| tb2 | copy of ta2 parametrizing q |
| bb2 | copy of ba2 retrieved by q |
| pb4 | copy of pb3 returned by q |
| bc2 | copy of bb2 returned by q |

# Provenance Data

- Information of operations/transactions performed against data objects and versions
  - Actions that were performed against data
  - Agents who performed actions on data
  - Data used for actions
  - Data generated from actions

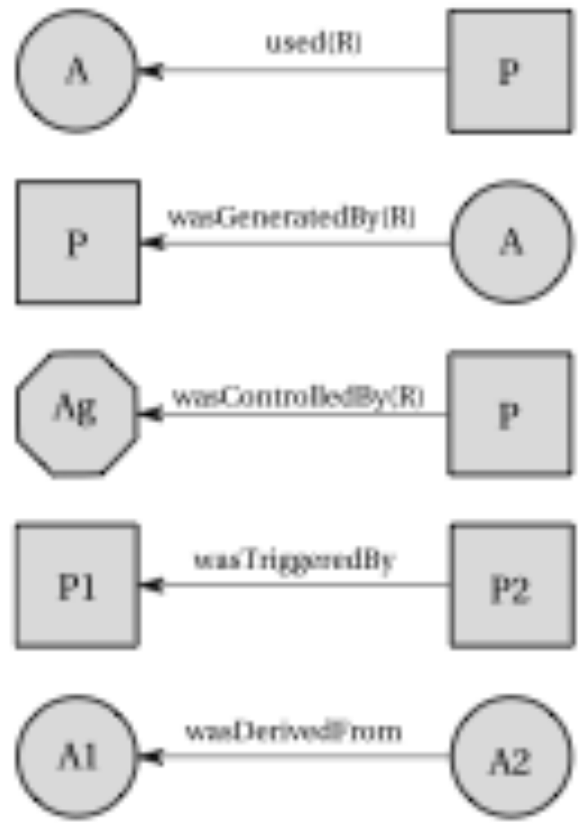
- Directed Acyclic Graph (DAG)
- Causality dependencies between entities (acting users, action processes and data objects)

- Dependency graph can be traced for the discovery of Origin, usage, versioning info, etc.

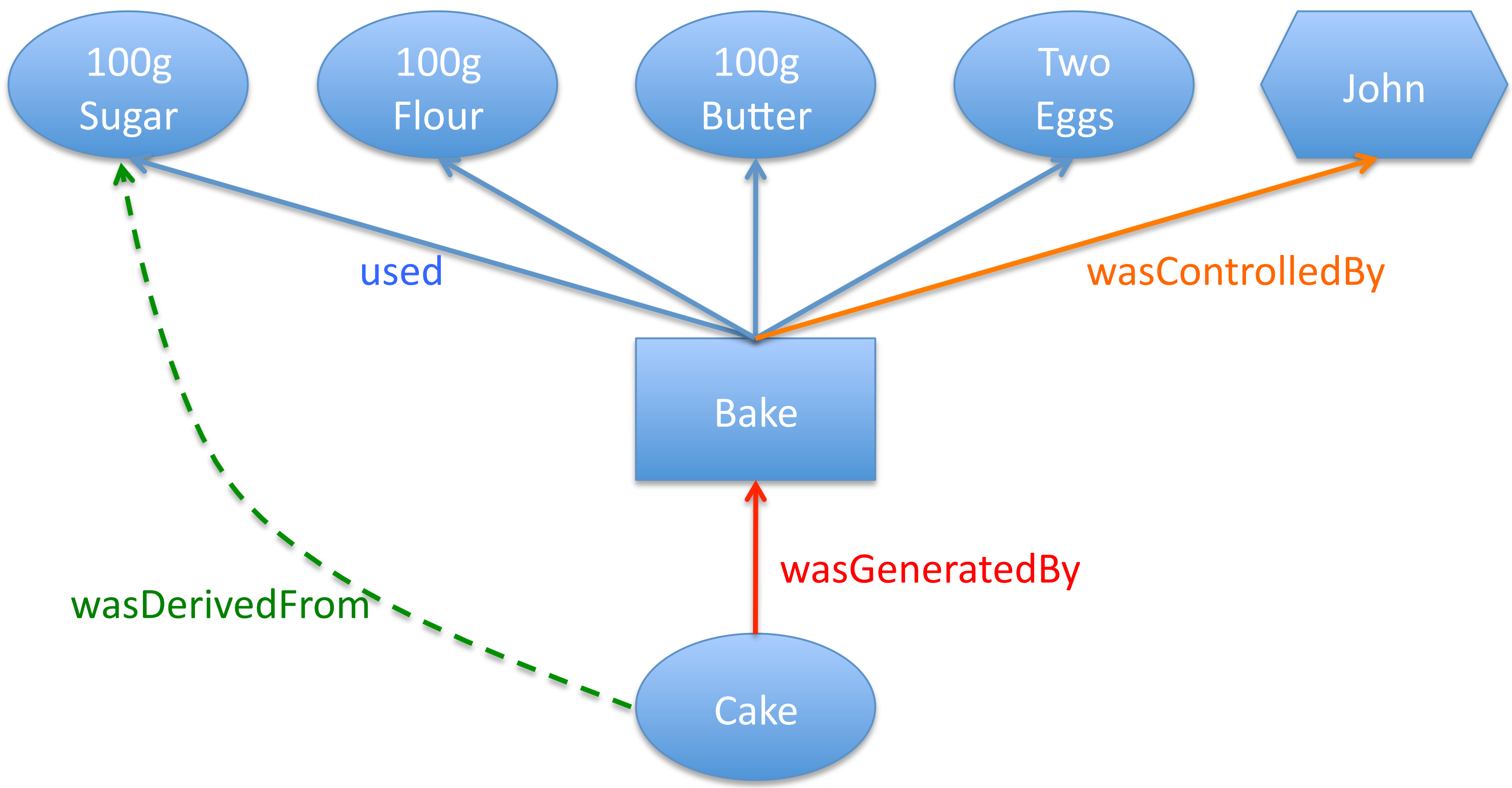
# Open Provenance Model (OPM)

- 3 Node Types
  - Artifact (ellipse): Object
  - Process (Rectangle): Action
  - Agent (Octagon/Hexagon): User/ Subject
- 5 Causality dependency edge Types (not a dataflow)
  - U: Used(Role)
  - G: wasGeneratedBy(Role)
  - C: wasControlledBy(Role)

  - wasDerivedFrom
  - wasTriggeredBy



# OPM Example



100g Sugar
100g Flour
100g Butter
Two Eggs
John
used
wasControlledBy
Bake
wasGeneratedBy
wasDerivedFrom
Cake

# Provenance-aware Systems

- Capturing provenance data
- Storing provenance data
- Querying provenance data

Provenance Data Model

- Using provenance data
- Securing provenance data

Access Control

# Access control in Provenance-aware Systems

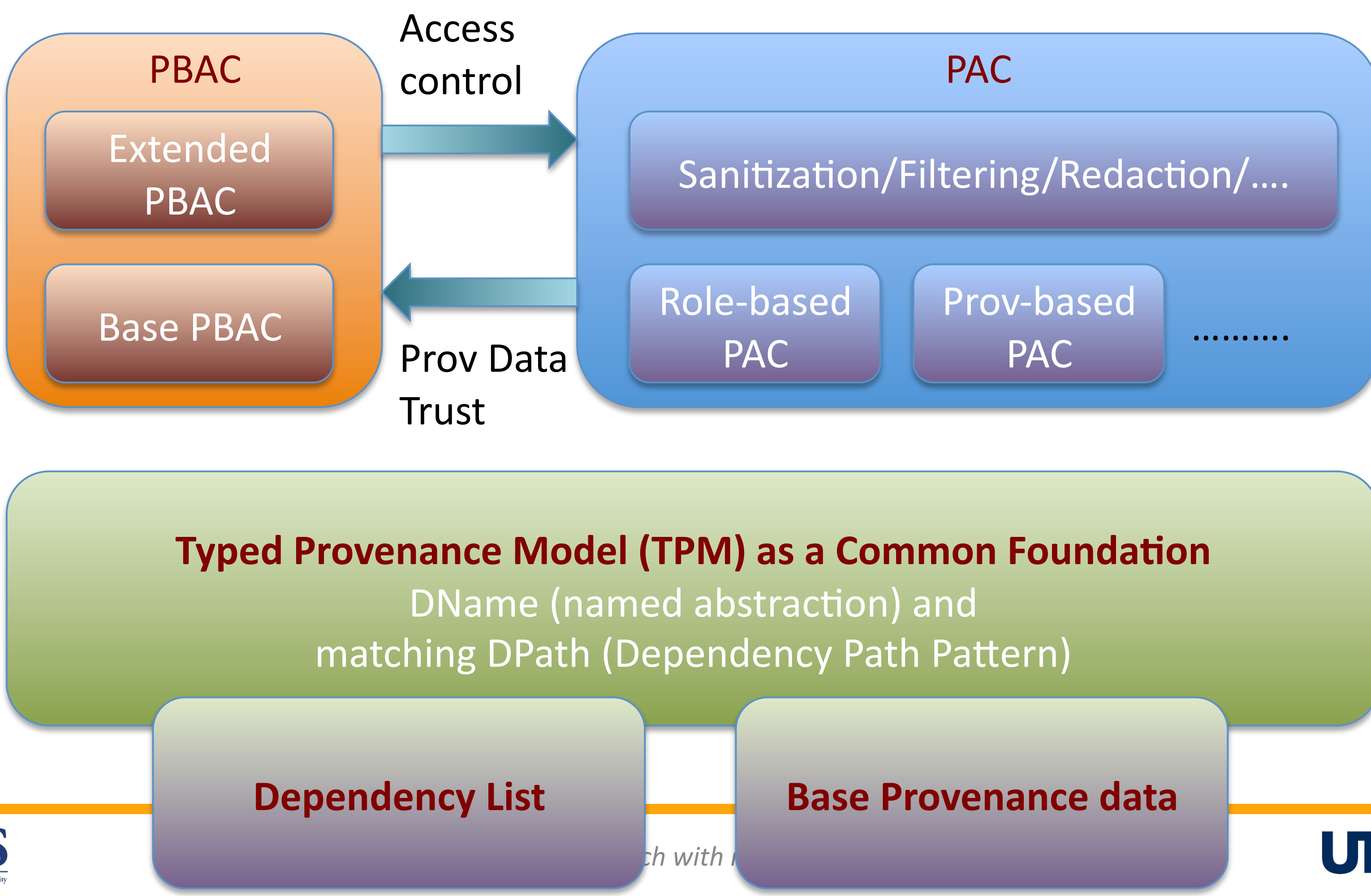
- Provenance Access Control (PAC)
  - Controlling access to provenance data which could be more sensitive than the underlying data
  - Two concerns
    - Needs access control models/mechanisms (e.g, RBAC)
    - (Meaningful) control granularity? Right level of abstraction?

- Provenance-based Access Control (PBAC)
  - Using provenance data to control access to the underlying data
  - Provenance-based policy specification

Meaningful granularity of provenance data?

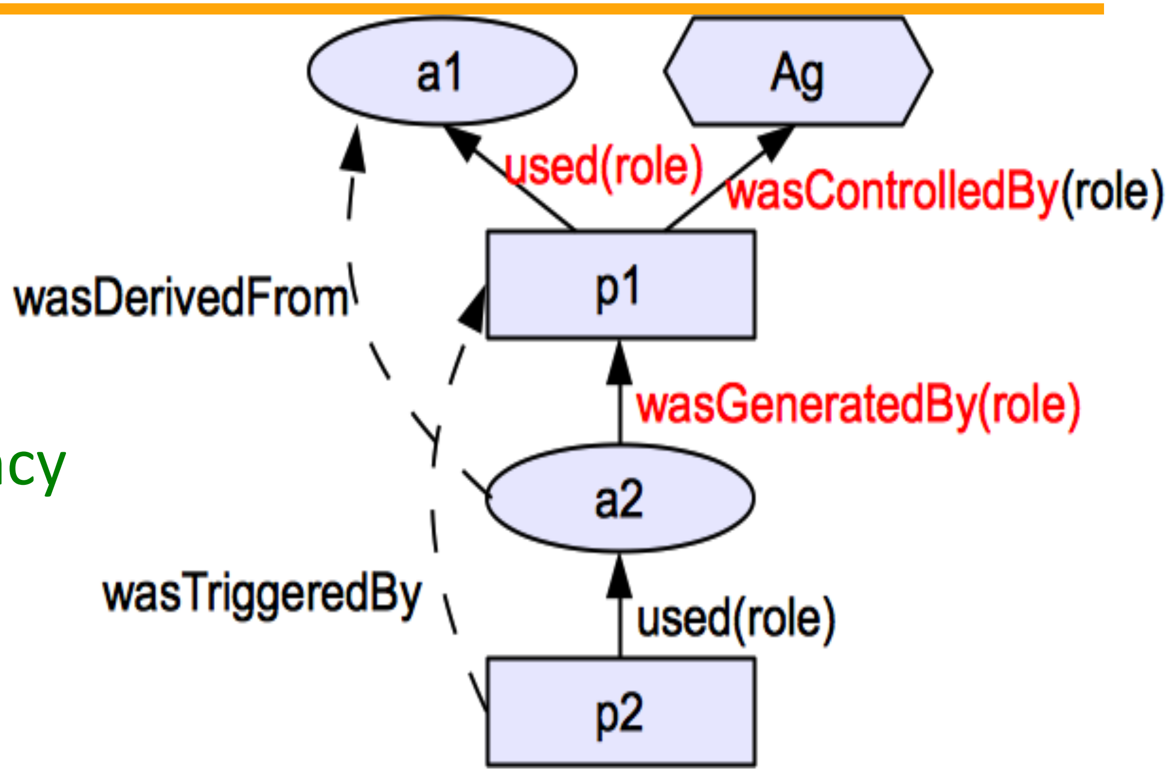
Typed Provenance Model (TPM)

# Access Controls in Provenance-aware Systems



# From Open Provenance Model (OPM)

- 3 Node Types
  - Object (Artifact)
  - Action (Process)
  - User/Subject (Agent)
- 5 Causality dependency edge Types (not a dataflow)




- Provenance data: a set of 2 entities & 1 dependency
  - E.g., (ag,p1,a1,a2): <p1,ag,c>,<p1,a1,u>,<a2,p1,g>

# Typed Provenance Model (TPM)

- Primitive (direct) dependency types
  - Variations of Used (u), wasGeneratedBy (g), wasControlledBy (c)
  - Used to capture transactions as base provenance data

- Complex (Indirect) dependency types
  - A set of pairs consisting of abstract name (dependency names) and regular expression-based composition of primitive dependency types or other complex dependency types (dependency paths)

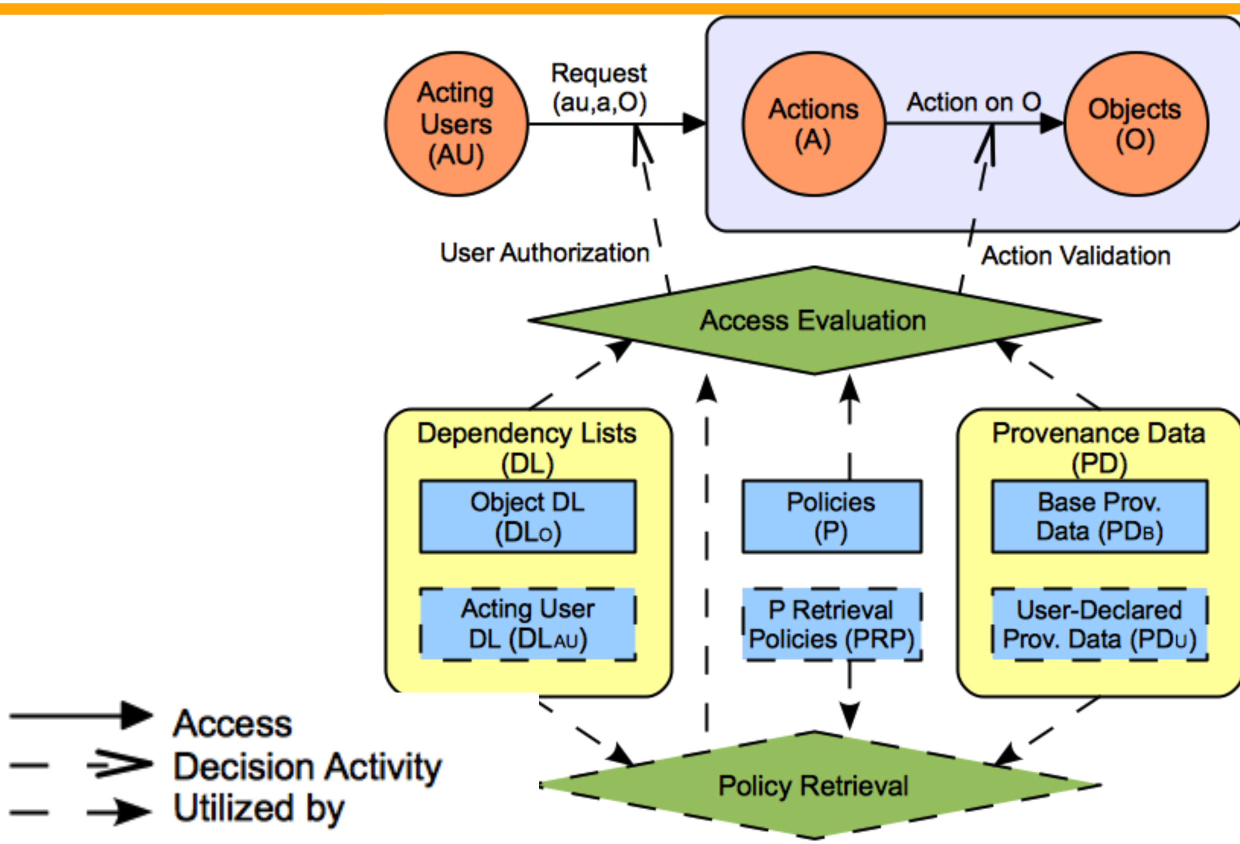
# Dependency List

- Object Dependency List ($DL_O$): A set of identified dependencies that consists of pairs of
  - Dependency Name: abstracted dependency names (DNAME) and
  - regular expression-based object dependency path pattern (DPATH)

- System-computable (complex) dependency instances
  - using pre-defined dependency names and matching dependency path patterns in DL (and querying base provenance data)
- User-declared (complex) dependency instances
  - using pre-defined dependency names in DL

- Examples
  - < wasSubmittedVof, $g_{submit}.u_{input}$ >
  - < wasAuthoredBy, wasSubmittedVof?.wasReplacedVof *.$g_{upload}$.c >

# PBAC Model Components

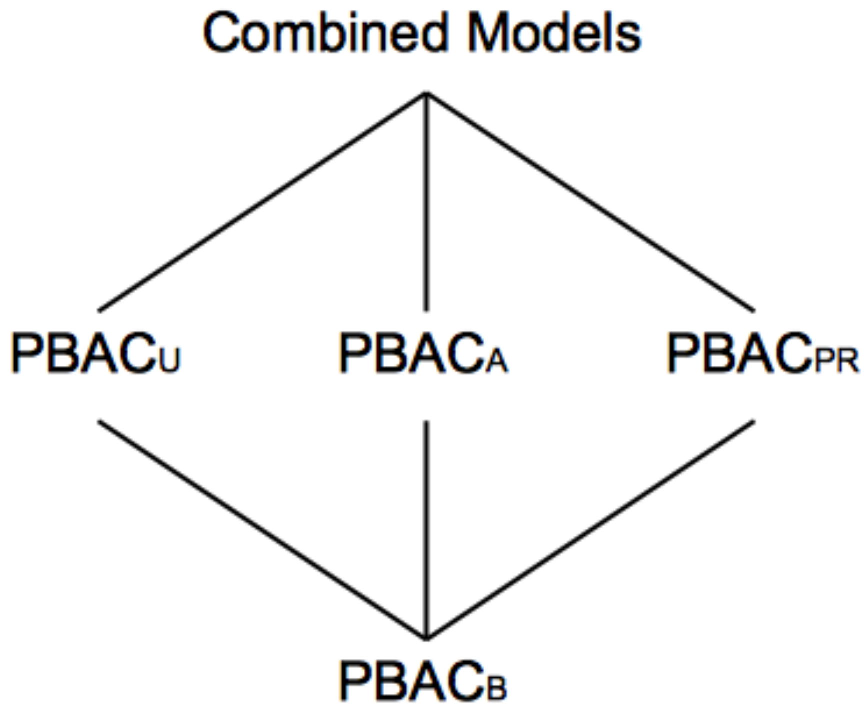



< wasSubmittedVof, $g_{submit}.u_{input}$ >

< wasReplacedVof, $g_{replace}.u_{input}$ >

< wasAuthoredBy, wasSubmittedVof?.wasReplacedVof $*.g_{upload}.c$ >

# A Family of PBAC Models



# PBAC$_B$: A Base Model

- System-captured Base Provenance Data only
  - Using sub-types of 3 direct dependencies (u, g, c)
  - No user-declared provenance data
- Object dependency only
- Policy is readily available
  - No policy retrieval required

# PBAC$_B$ Model Definitions

1) $AU, A, AT, O$ and $OR$ are acting users, action instances, action types, objects, and object roles respectively.
2) $G, U, G^{-1}$ and $U^{-1}$ are sets of role-specific variations of 'wasGeneratedBy' and 'used' dependencies and matching sets of inverse dependencies, respectively.
3) $\{$'$c$', '$c^{-1}$'$\}$ is the set of 'wasControlledBy' dependency and its inverse dependency.
4) Base provenance data $PD_B$ forms a directed graph and is formally denoted as a triple $< V_B, E_B, D_B >$:
   - $V_B = AU \cup A \cup O$, a finite set of acting users, action instances, and objects that have been involved in transactions in the system and are represented as vertices;
   - $D_B = \{\text{'}c\text{'}\} \cup U \cup G \cup \{\text{'}c^{-1}\text{'}\} \cup U^{-1} \cup G^{-1}$, a finite set of base dependency types;
   - $E_B \subseteq \{(A \times AU \times \text{'}c\text{'}) \cup (A \times O \times U) \cup (O \times A \times G) \cup (AU \times A \times \text{'}c^{-1}\text{'}) \cup (O \times A \times U^{-1}) \cup (A \times O \times G^{-1})\}$, denoting dependency edges, is the set of existing base dependencies in the provenance data.[4]
5) $DN_O$, disjoint from $D_B$, is a finite set of abstracted names for dependencies of objects.
6) Let $\Sigma$ be an alphabet of terms in $D_B \cup DN_O$. The set $DPATH$ of regular expressions is inductively defined as follows:
   - $\forall p \in \Sigma, p \in DPATH; \epsilon \in DPATH;$
   - $(P_1|P_2), (P_1.P_2), P_1*, P_1+, P_1? \in DPATH$, where $P_1 \in DPATH$ and $P_2 \in DPATH$.
7) $DPATH_B \subseteq DPATH$, is the set of regular expression using only alphabet of terms in $D_B$.
8) $DL_O : DN_O \rightarrow DPATH$, defines each $dn \in DN_O$ as a path expression. $DL_O$ is also viewed as a list of pairs of object dependency names and corresponding dependency paths.
9) $\lambda_O : DN_O \rightarrow DPATH_B$, maps each $dn \in DN_O$ to a path expression using only base dependency types $d_b \in D_B$ by repeatedly expanding the definitions of any $dn_i \in DN_O$ that occurs in $DL_O(dn)$.
10) $PE$ is a language specified in the policy expression grammar $PG$.
11) $P \subseteq PE$, is a finite set of policies.
12) $\gamma : AT \rightarrow P$, a mapping of an action type to a policy.

# Access Evaluation Algorithm

**Algorithm 1** $AccessEvaluation(au, a, O)$

```
(Rule Collecting Phase)
at ← a's action type
p ← γ(at)
RULE_UA ← user authorization rules UARule found in p
RULE_AV ← action validation rules AVRule found in p
(User Authorization Phase)
for all rules in RULE_UA do
    Extract the path rule (ObjRole, DName) from rules
    Determine the object o ∈ O, whose role is ObjRole
    Extract dependency path expression dpath_b in DPATH_B from DName using λ_O function
    Determine vertices by tracing base provenance data PD_B through the paths expressed in dpath_b that start from the object o using δ_O function
    Determine the truth value by evaluating the result against the rule
end for
UAuth ← a combined truth value based on conjunctive or disjunctive connectives between rules
(Action Validation Phase)
for all rules in RULE_AV do
    Extract path rules (ObjRole, DName) from rules
    for all path rules extracted do
        Determine the object o ∈ O, whose role is ObjRole
        Extract dependency path expression dpath_b in DPATH_B from DName using λ_O function
        Determine vertices by tracing base provenance data PD_B through the paths expressed in dpath_b that start from the object o using δ_O function
    end for
    Determine the truth value by evaluating the results of all the extracted path rules
end for
AVal ← a combined result based on conjunctive or disjunctive connectives between rules
Evaluate a final truth value of UAuth and AVal using conjunctive connective
```

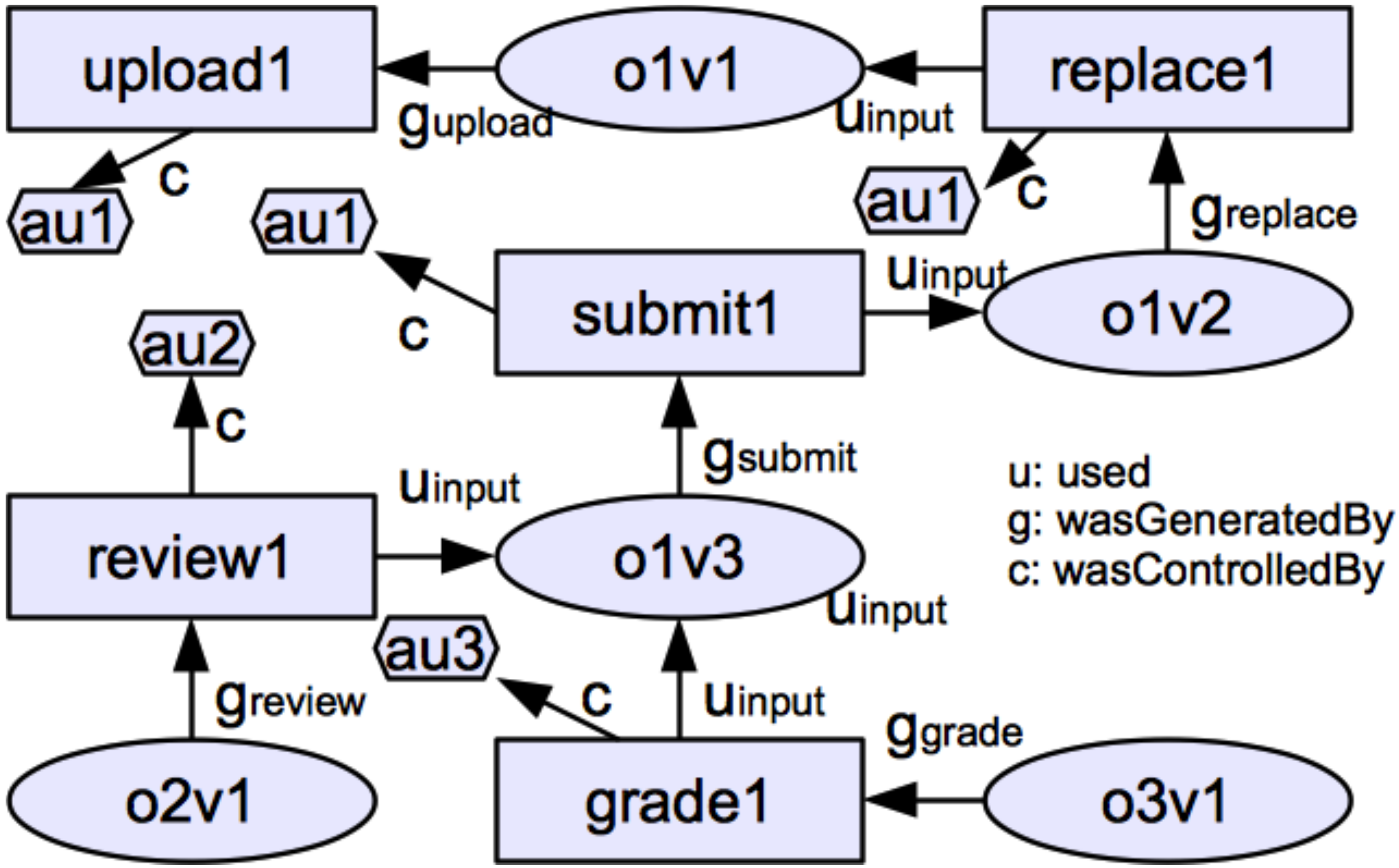
# $\text{PBAC}_\text{B}$ Policy Language Grammar

$Policy ::=$ "$allow$" $< Req >$ " $\Rightarrow$ " $< UARules >$ " $\wedge$ " $< AVRules > |$
"$true$"
$Req ::=$ "(" $< ActUser >$ "," $< ActType >$ "," $< ObjRoles >$ ")"
$ObjRoles ::= < ObjRole > | < ObjRole >$ "," $< ObjRoles >$
$UARules ::= < UARule > |$"(" $< UARules >$ ")"$|$
$< UARules >< Connect >< UARules >$
$AVRules ::= < AVRule > |$"(" $< AVRules >$ ")"$|$
$< AVRules >< Connect >< AVRules >$
$Connect ::= \vee | \wedge$
$UARule ::= < ActUser >< oper1 >< PathRule >$
$AVRule ::=$ "$|$" $< PathRule >$ "$|$" $< oper2 >< Number > |$
$< PathRule >< oper3 >< PathRule >$
$PathRule ::=$ "(" $< ObjRole >$ "," $< DName >$ ")"
$oper1 ::=$ " $\in$ "$|$" $\notin$ "
$oper2 ::=$ " $=$ "$|$" $\neq$ "$|$" $\geq$ "$|$" $\leq$ "$|$" $<$ "$|$" $>$ "
$oper3 ::=$ " $=$ "$|$" $\neq$ "$|$" $\subseteq$ "
$DName ::= dn_1 | dn_2 | \ldots | dn_n$
$Number ::= [0-9]+$
$ActUser ::= au$
$ActType ::= at_1 | at_2 | \ldots | at_m$
$ObjRole ::= o_{role_1} | o_{role_2} | \ldots | o_{role_k}$

# Example: A Homework Grading System

1. Anyone can upload a homework.
2. A user can replace a homework if she uploaded it (origin-based control) and the homework is not submitted yet.
3. A user can submit a homework if she uploaded it and the homework is not submitted already. (workflow control)
4. A user can review a homework if she is not the author of the homework (DSOD), the user did not review the homework earlier, and the homework is submitted already but not graded yet.
5. A user can grade a homework if the homework is reviewed but not graded yet.

# Sample Transactions



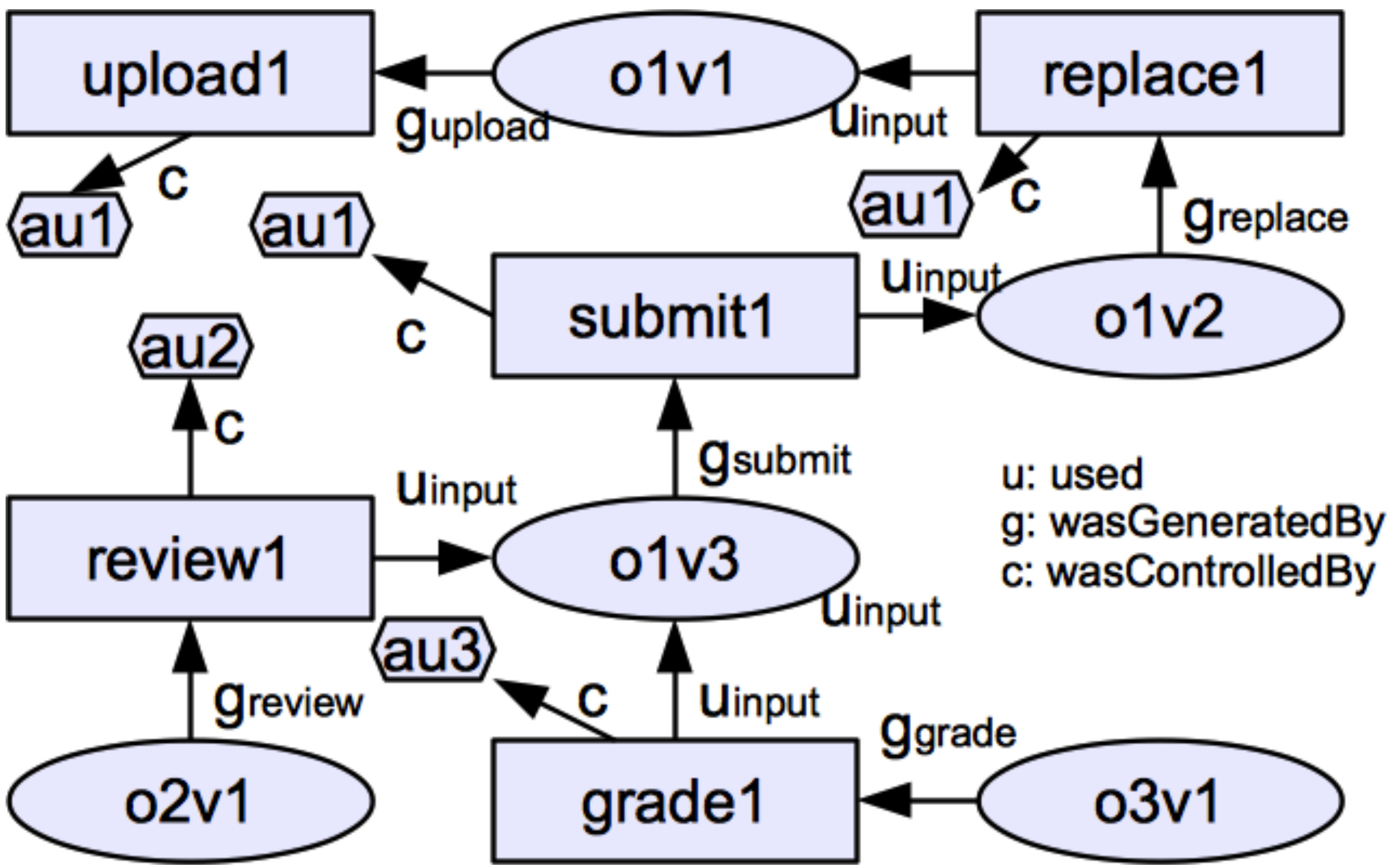
# Sample Transactions & Base Provenance Data

- (au1, upload1, $o_{1v1}$):
  - *< upload1, au1, c >, <$o_{1v1}$,upload1,$g_{upload}$ >*
- (au1, replace1, $o_{1v1}$, $o_{1v2}$):
  - *< replace1, au1, c >,< replace1, $o_{1v1}$, $u_{input}$>, < $o_{1v2}$, replace1, $g_{replace}$>*
- (au1, submit1, $o_{1v2}$, $o_{1v3}$):
  - *< submit1, au1, c >, <submit1,$o_{1v2}$, $u_{input}$ >,<$o_{1v3}$,submit1,$g_{submit}$ >*
- (au2, review1, $o_{1v3}$, $o_{2v1}$):
  - *< review1, au2, c >, <review1,$o_{1v3}$, $u_{input}$ >,<$o_{2v1}$,review1,$g_{review}$ >*
- (au3, grade1, $o_{1v3}$, $o_{3v1}$):
  - *< grade1, au3, c >, < grade1,$o_{1v3}$, $u_{input}$ >,< $o_{3v1}$,grade1,$g_{grade}$ >*

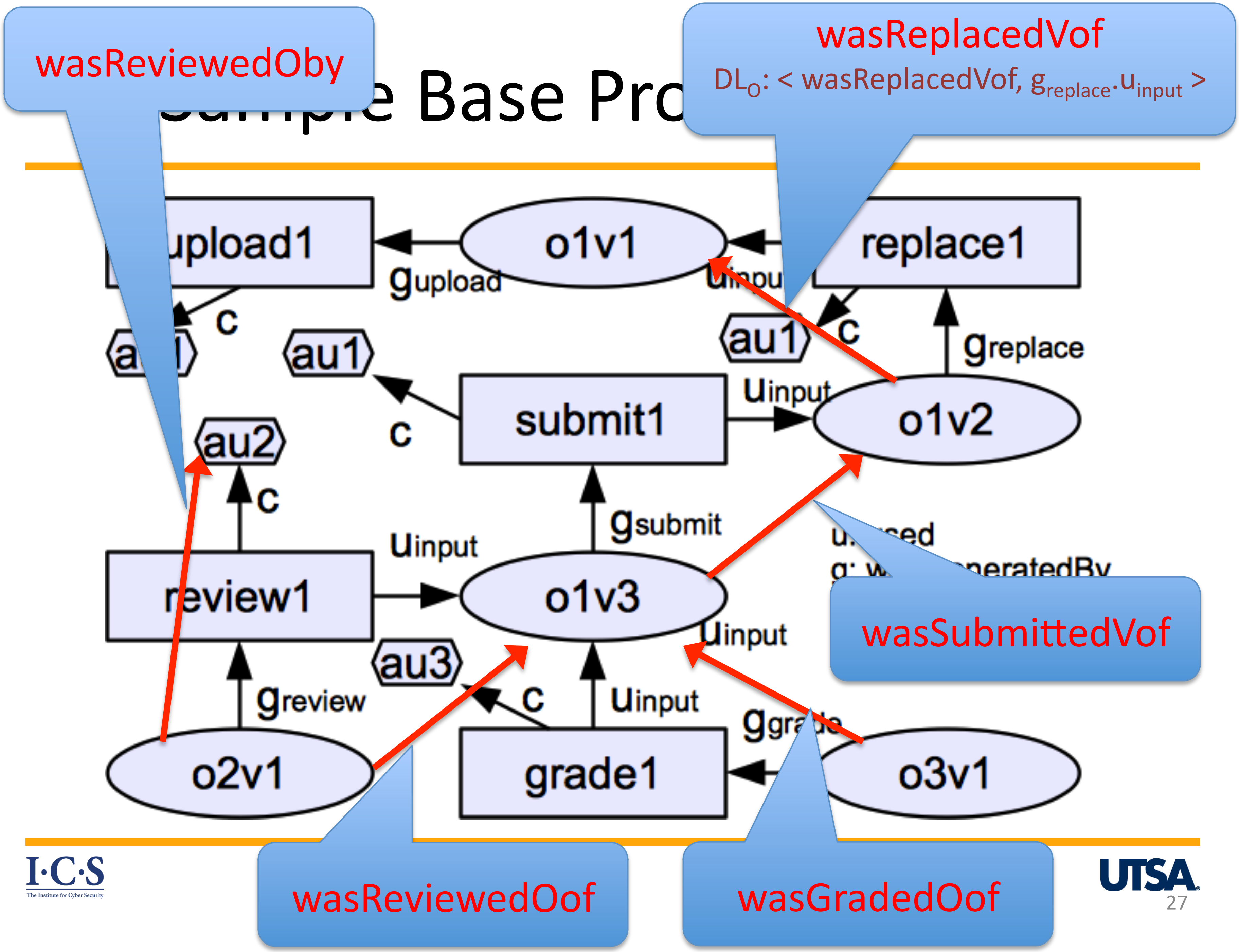
# Sample Object Dependency List ($DL_O$)

1. < wasReplacedVof, $g_{replace}.u_{input}$ >
2. < wasSubmittedVof, $g_{submit}.u_{input}$ >
3. < wasReviewedOof, $g_{review}.u_{input}$ >
4. < wasReviewedOby, $g_{review}.c$ >
5. < wasGradedOof, $g_{grade}.u_{input}$ >
6. < wasAuthoredBy, wasSubmittedVof?.wasReplacedVof $*.g_{upload}.c$ >
7. < wasReviewedBy, wasReviewedOof$^{-1}$. wasReviewedOby >

# A Sample Base Provenance Data



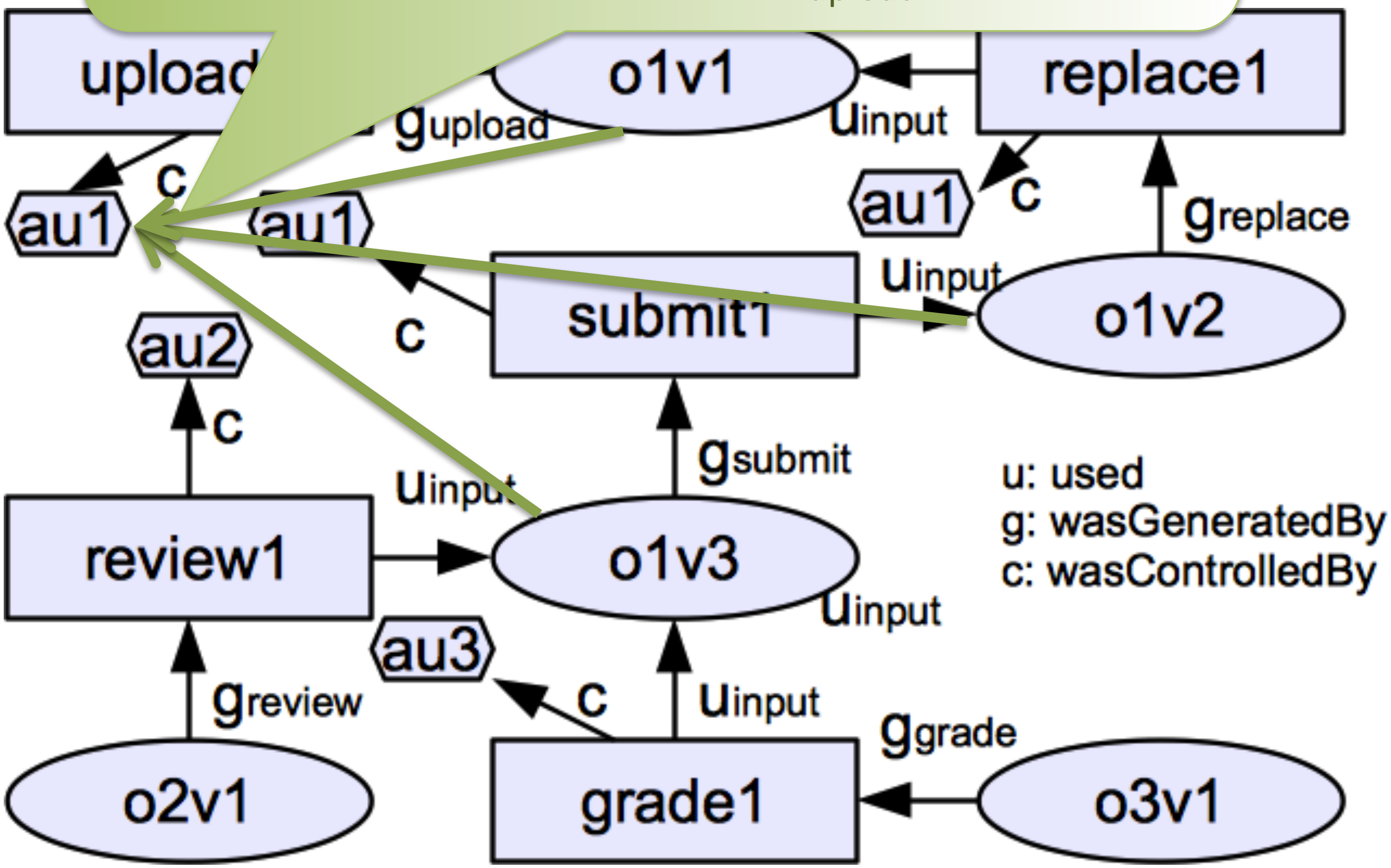
# Sample Base Pro



# A … Data

wasAuthtoredBy

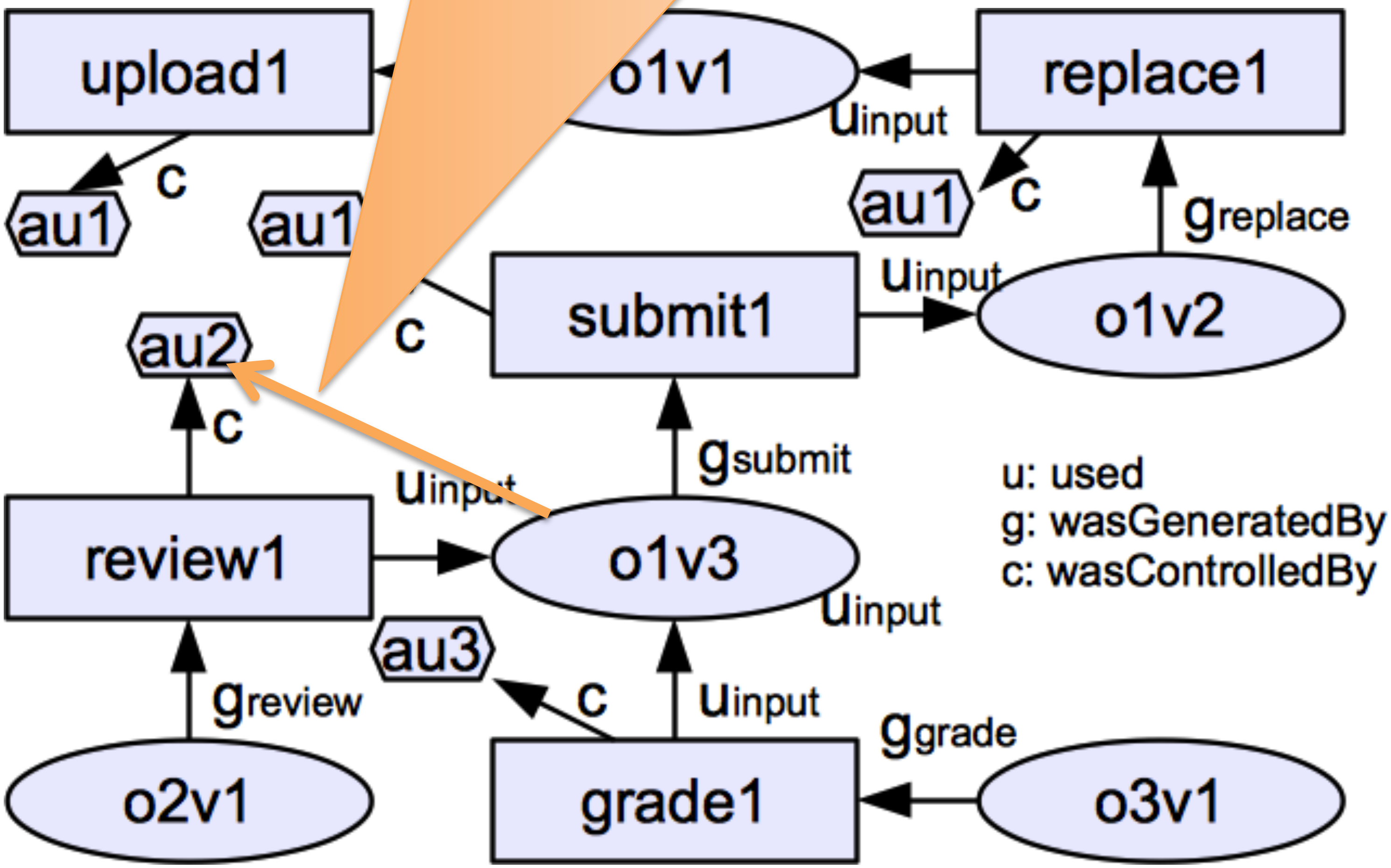
$DL_O$: <wasAuthoredBy, wasSubmittedVof?. wasReplacedVof $*$.$g_{upload}$.c >



# A San n

wasReviewedBy

$DL_O$: < wasReviewedBy, wasReviewedOof$^{-1}$. wasReviewedOby >



# Sample Policies

1. Anyone can upload a homework.
2. A user can replace a homework if she uploaded it (origin-based control) and the homework is not submitted yet.
3. A user can submit a homework if she uploaded it and the homework is not submitted already. (workflow control)

1. allow(au, upload, o) $\Rightarrow$ true
2. allow(au, replace, o) $\Rightarrow$ au $\in$ (o, wasAuthoredBy) $\wedge$ |(o,wasSubmittedVof)| = 0.
3. allow(au, submit, o) $\Rightarrow$ au $\in$ (o, wasAuthoredBy) $\wedge$ |(o,wasSubmittedVof)|=0.

# Sample Policies (cont.)

4. A user can review a homework if she is not the author of the homework (DSOD), the user did not review the homework earlier, and the homework is submitted already but not graded yet.
5. A user can grade a homework if the homework is reviewed but not graded yet.

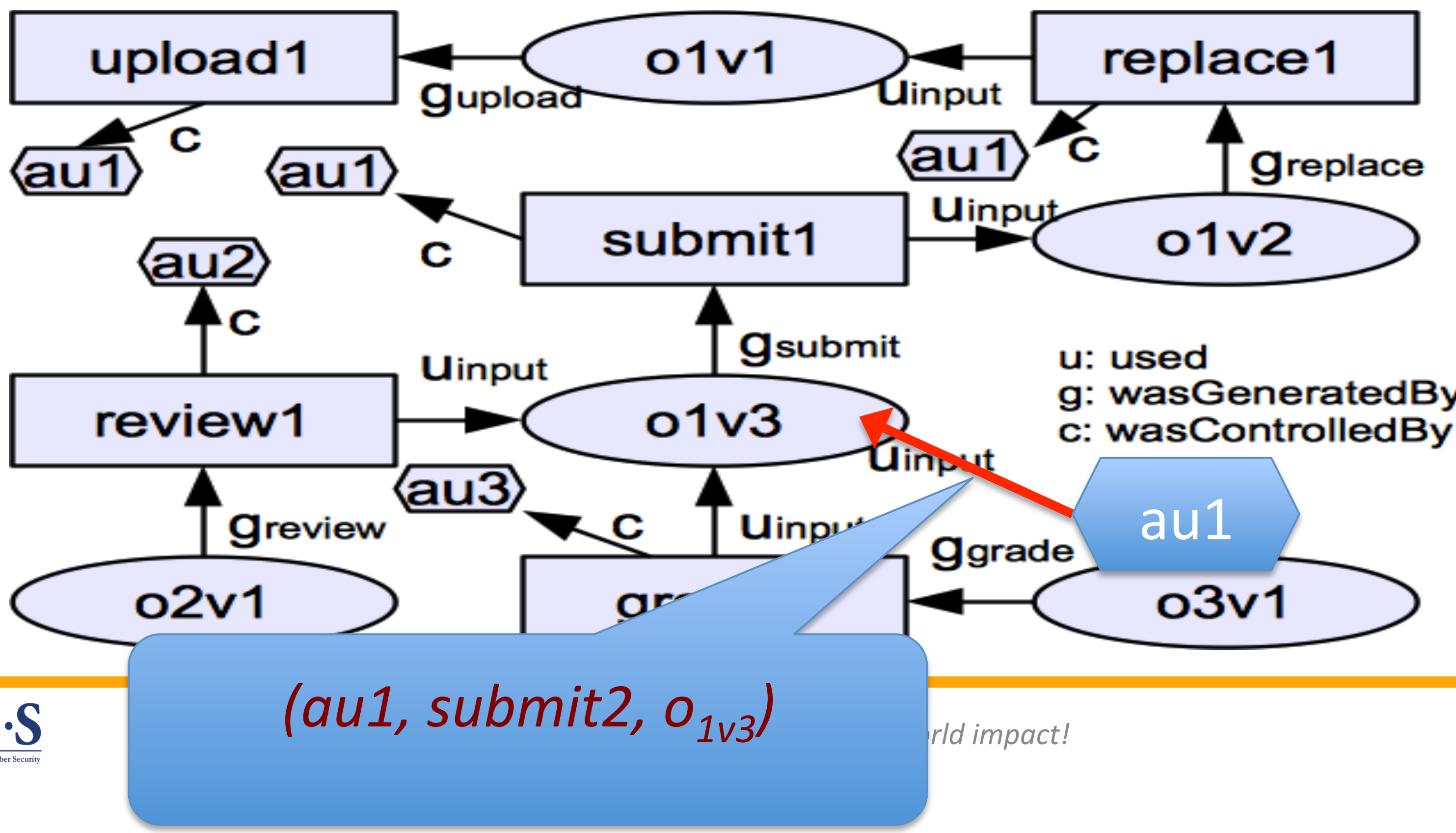
4. allow(au, review, o) $\Rightarrow$ au $\notin$ (o, wasAuthoredBy) $\wedge$ au $\notin$ (o, wasReviewedBy) $\wedge$ $|$(o, wasSubmittedV of)$|$ $\neq 0$ $\wedge$ $|$(o,wasGradedOof$^{-1}$)$|$ = 0.
5. allow(au, grade, o) $\Rightarrow$ $|$(o, wasReviewedOof)$|$ $\neq 0$ $\wedge$ $|$(o,wasGradedOof$^{-1}$)$|$ = 0).

# Access Evaluation Procedure

- Rule collecting phase
- User authorization (UAuth) phase
- Action validation (AVal) phase
- conjunctive decision of UAuth and AVal

# Access Evaluation Example

- Policy: user can submit a homework if she uploaded it (origin-based control) and the homework is not submitted already. (workflow control)

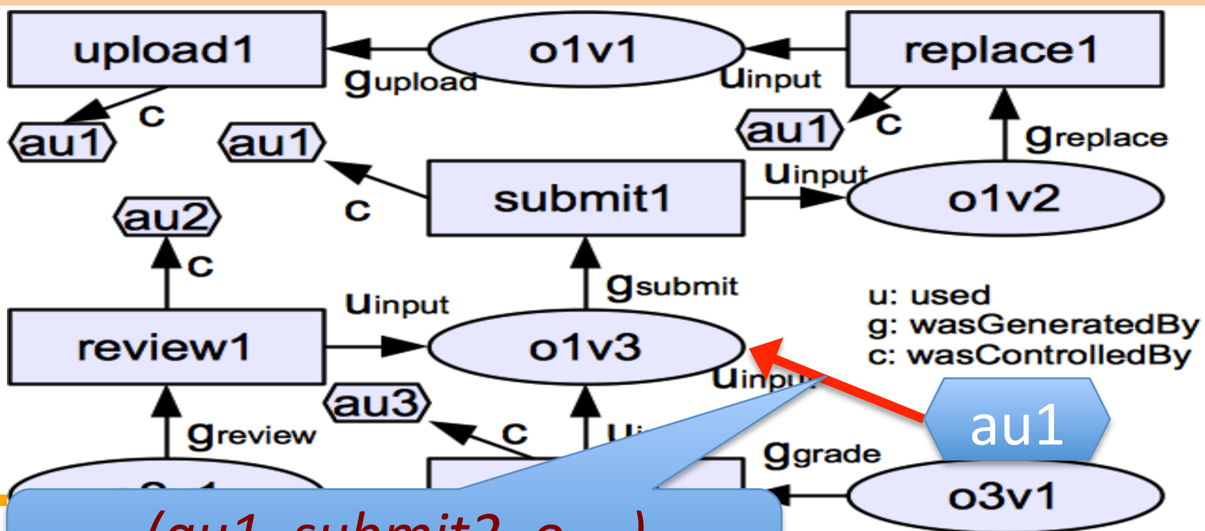


# Rule Collecting Phase

- Request: *(au1, submit2, $o_{1v3}$)*
- Action type: *submit*
- Policy for *submit*
  – *allow(au, submit, o) ⇒ au ∈(o, wasAuthoredBy) ∧ |(o,wasSubmittedVof)| = 0.*
- User authorization rule
  – *au ∈(o, wasAuthoredBy)*
- Action Validation rule
  – *|(o,wasSubmittedVof)| = 0*

# User Authorization Phase

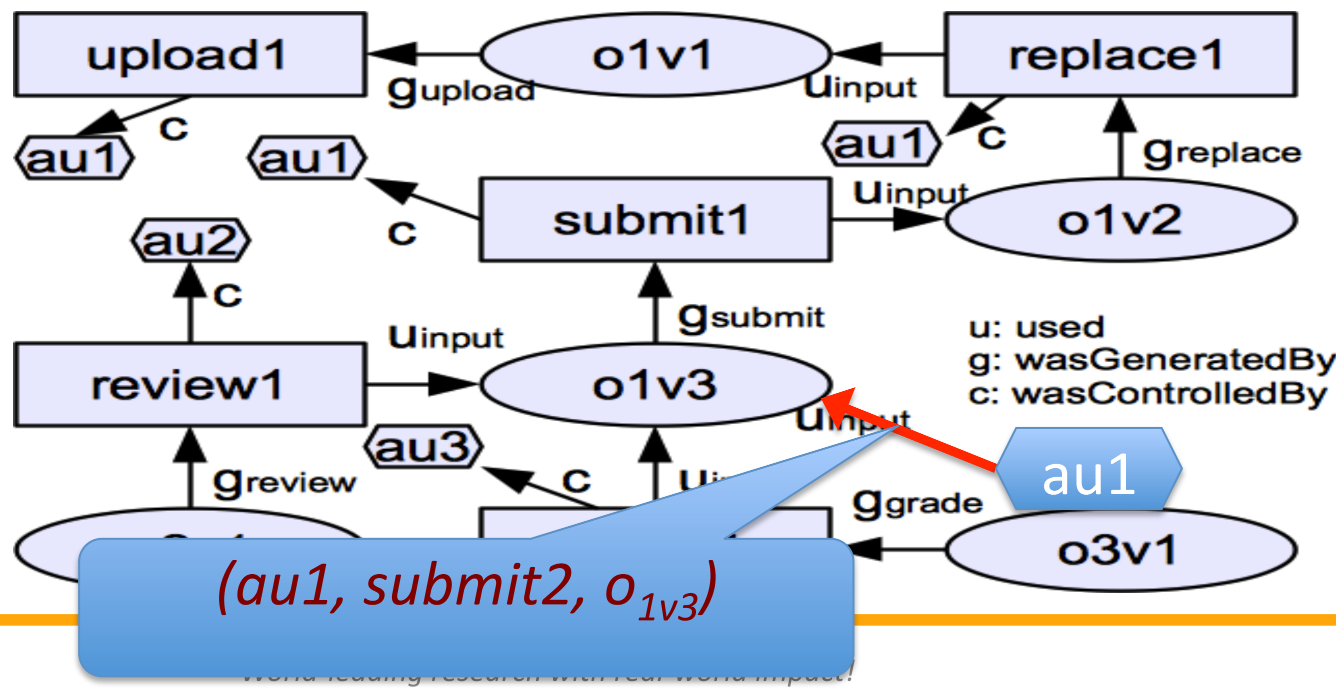
- *User Authorization Rule: au ∈(o, wasAuthoredBy)*
- Dependency List (DL)
  - *< wasReplacedVof, $g_{replace}.u_{input}$ >, < wasSubmittedVof, $g_{submit}.u_{input}$ >*
  - *< wasAuthoredBy, wasSubmittedVof?.wasReplacedVof *.$g_{upload}.c$ >*
- *au1 ∈ (o1v3, [$g_{submit}.u_{input}$]?.[$g_{replace}.u_{input}$]*.$g_{upload}.c$ ) = {au1}*



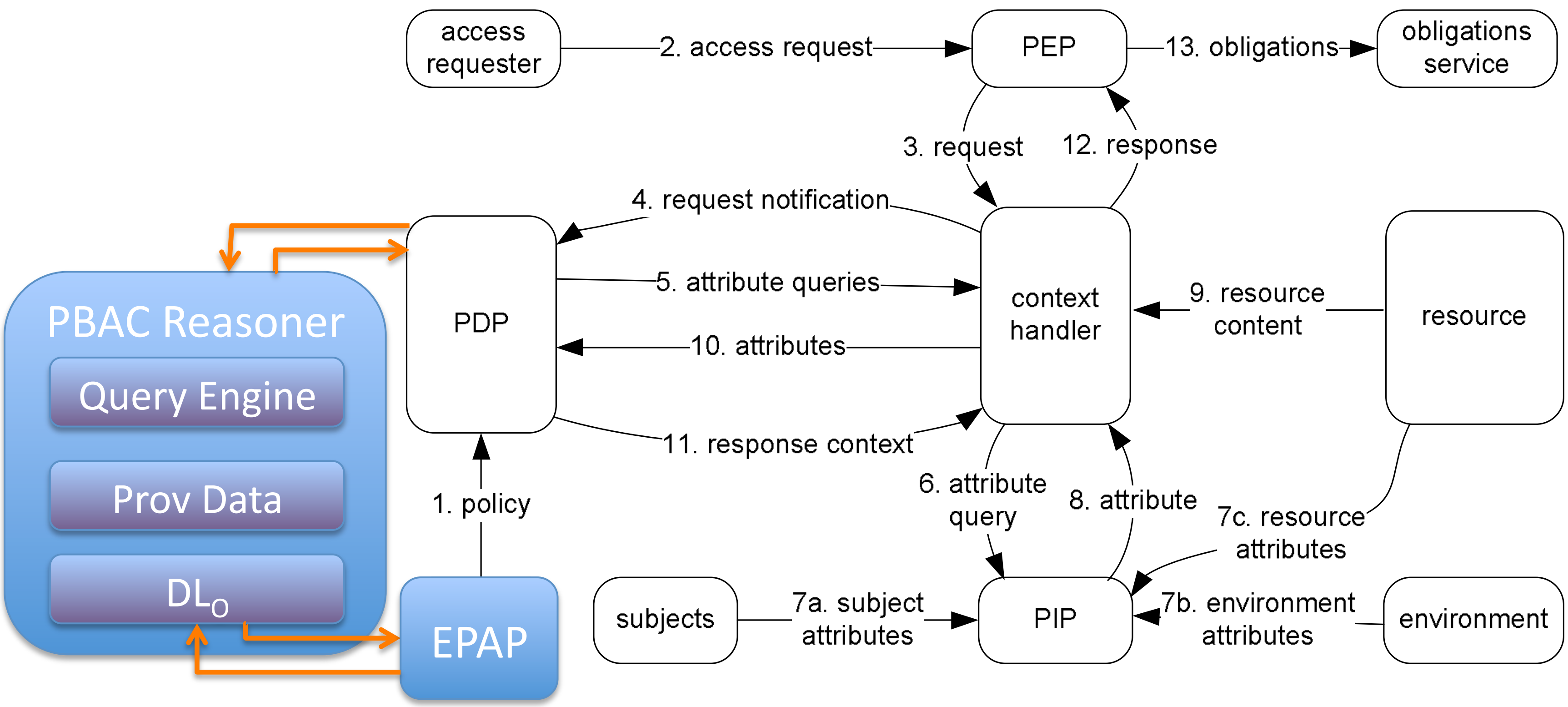

*(au1, submit2, $o_{1v3}$)*

# Action Validation Phase

- *Action Validation Rule: |(o,wasSubmittedVof)| = 0*
- Dependency List (DL): *< wasSubmittedVof, $g_{submit}.u_{input}$ >*
- *$|(o1v3, g_{submit}.u_{input})| \neq 0$*

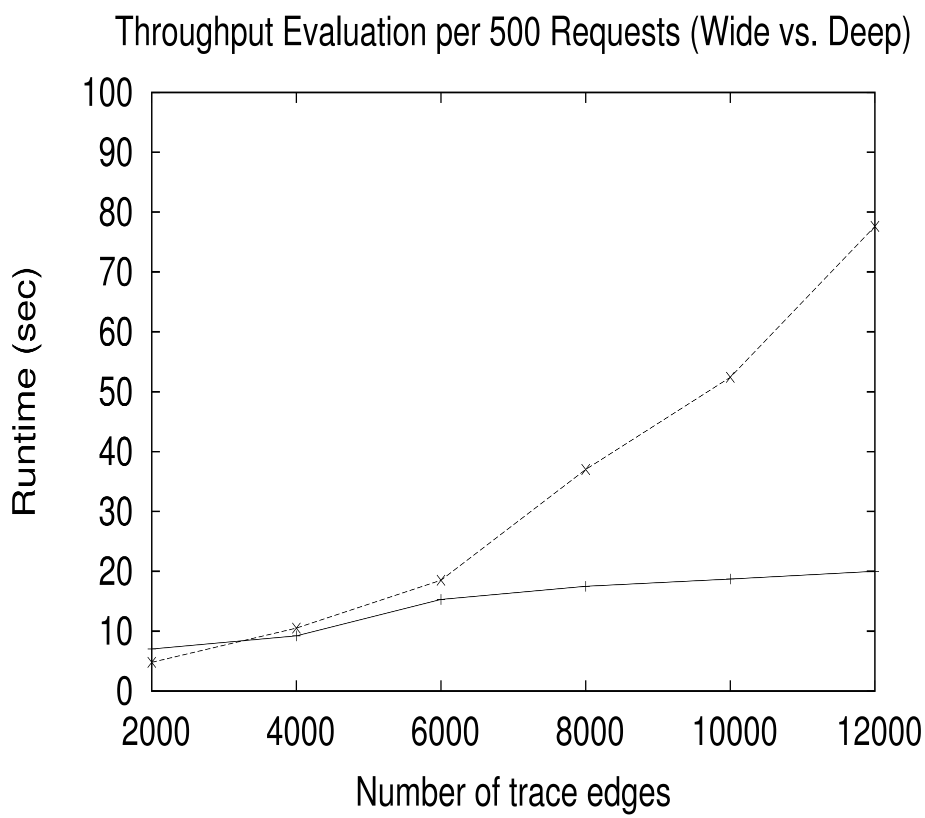


# PBAC$_B$ in XACML



# Implementation and Performance

- System
  - Ubuntu 12.10 image with 4GB Memory and 2.5 GHz quad-core CPU running on a Joyent SmartData center (ICS Private Cloud).
- Implementation
  - extended the PDP class in SUN's XACML
  - Apache Jena 2.7.4 and ARQ package to provide both the RDF-enabled data store for provenance graph and the ARQ query engine for enabling SPARQL queries.
- Results for tracing 2k/12k edges
  - 0.0096/0.154 second per deep request
  - 0.035/0.04 second per wide request



# Summary of $PBAC_B$

- Proposed a foundation for PBAC and PAC
  - the notion of named abstractions of causality dependency path patterns
  - Regular expression-based dependency path pattern
- Identified a Family of PBAC models
- Developed a Base model for PBAC
  - Supports Simple and effective policy specification and access control management
  - Supports DSOD, workflow control, origin-based control, usage-based control, object versioning, etc.

- Questions and Comments?