Resource Access Decision Facility: Overview

Konstantin Beznosov

Baptist Health Systems of South Florida
beznosov@baptisthealth.net

Do You Have Any of the Following:

- Systems from different vendors are deployed
- Either granularity of access control needs to be fine
- Access control policies are complex and relatively dynamic

- Free lunch?

Presentation Overview

- Why you need Resource Access Decision Facility
- Main aspects of RAD specification design
- Main design decisions made by RAD submission team
- Questions (if time permits)

RAD Trivia

- **Response to Healthcare Resource Access Control (HRAC) RFP corbamed/98-02-23**
- **Final response corbamed/99-05-04**
- **FTF August 1999**
- **Who did it:**
  - **2AB, IBM, NIST, BHS**
  - **With help from: CareFlow|Net, Concept Five, DASCOM, Inc., Inprise, Los Alamos National Laboratory, NSA, Philips Medical Systems, TIS Labs**

Why Do You Need Resource Access Decision Facility?

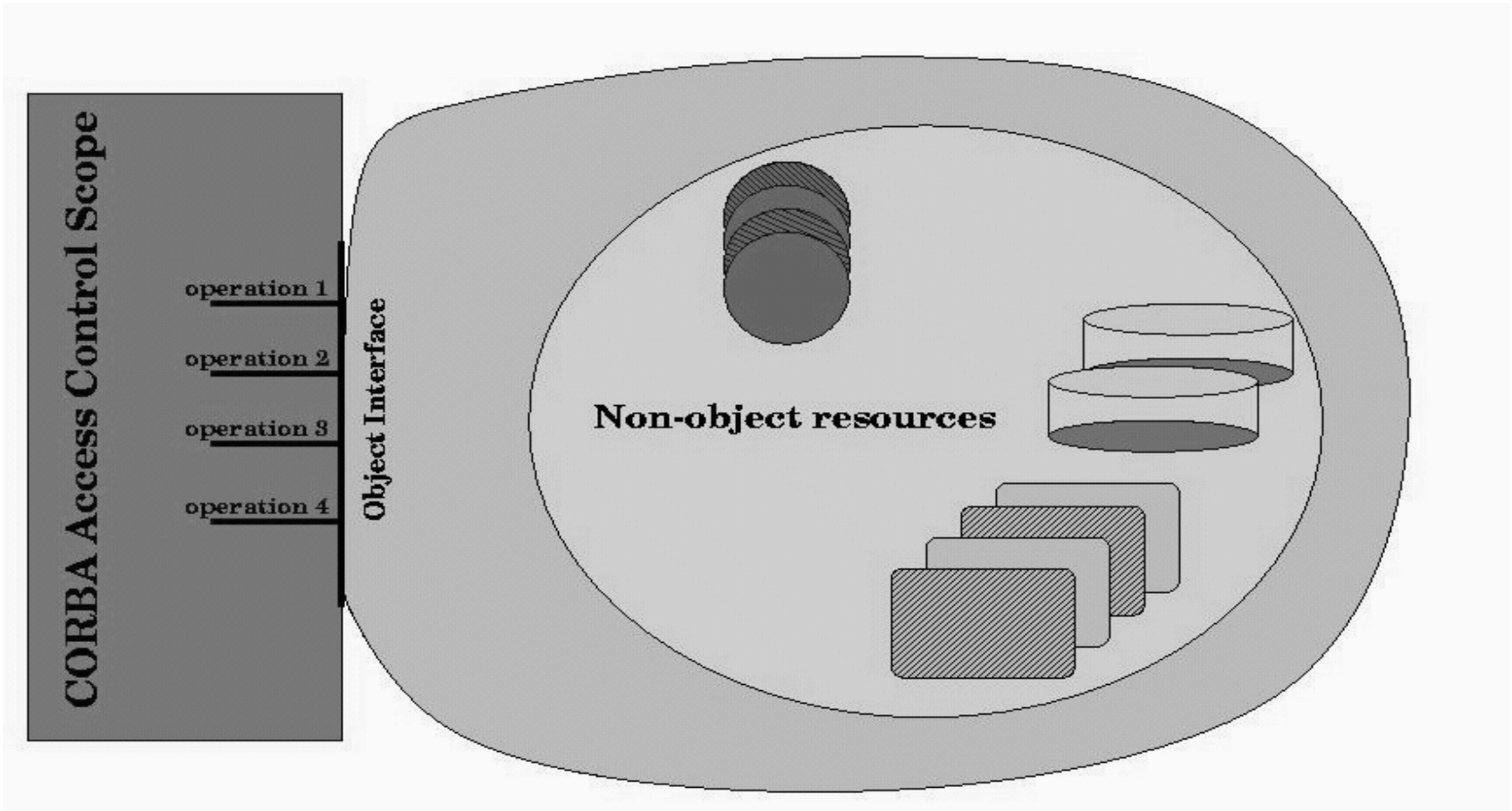
CORBASEC

- Versatile
- Accommodates most computing environments
- Optimized for the most common case
- Provides interfaces to applications if their security needs differ from the common case
- Do not pay if don't use

Why RAD?

- **Access control granularity**
- **Additional factors in authorization decisions**
- **Decoupling authorization logic from application logic**

Why RAD: Granularity

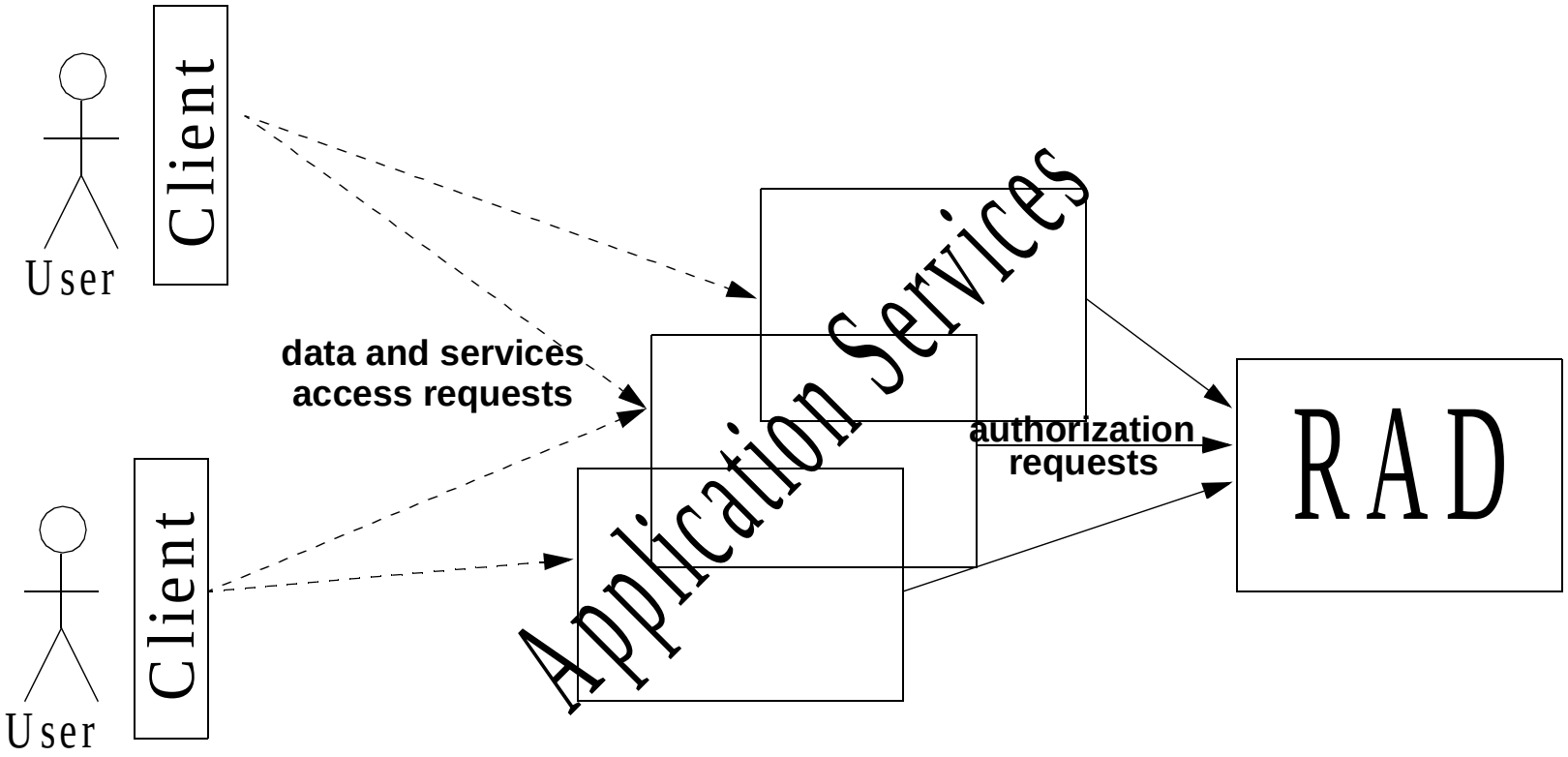


Why RAD: Additional Factors

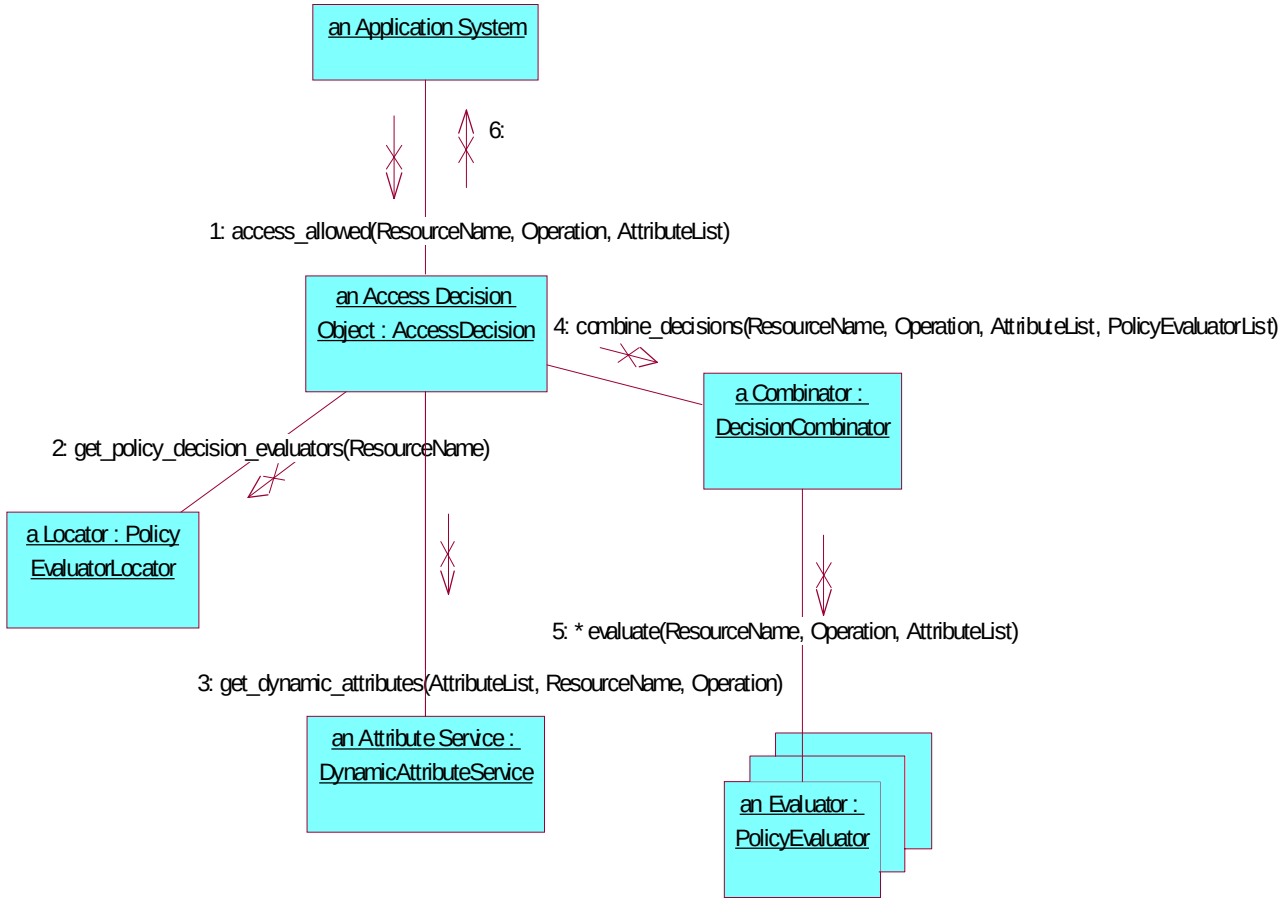
- **Some Need Authorization Decisions Based on**
  - **Standard CORBASEC Access Control Model**
    - **Name of interface and operation**
    - **Principal id**
    - **Principal role**
    - **Principal affiliation**
    - **...**
  - **Customized implementation of *AccessDecision* and *PrincipalAuthenticator***
    - **Time of service request**
    - **Location of service requester**
  - **Cannot be supported by CORBASEC Access Control Model**
    - **Relationship between the requesting principal and the "owner" of the data to be accessed**
    - **Values of input arguments on an operation.**
    - **Values of results returned from invocation of an operation.**

Main Aspects of RAD Specification Design

Users, Clients, Services, RAD



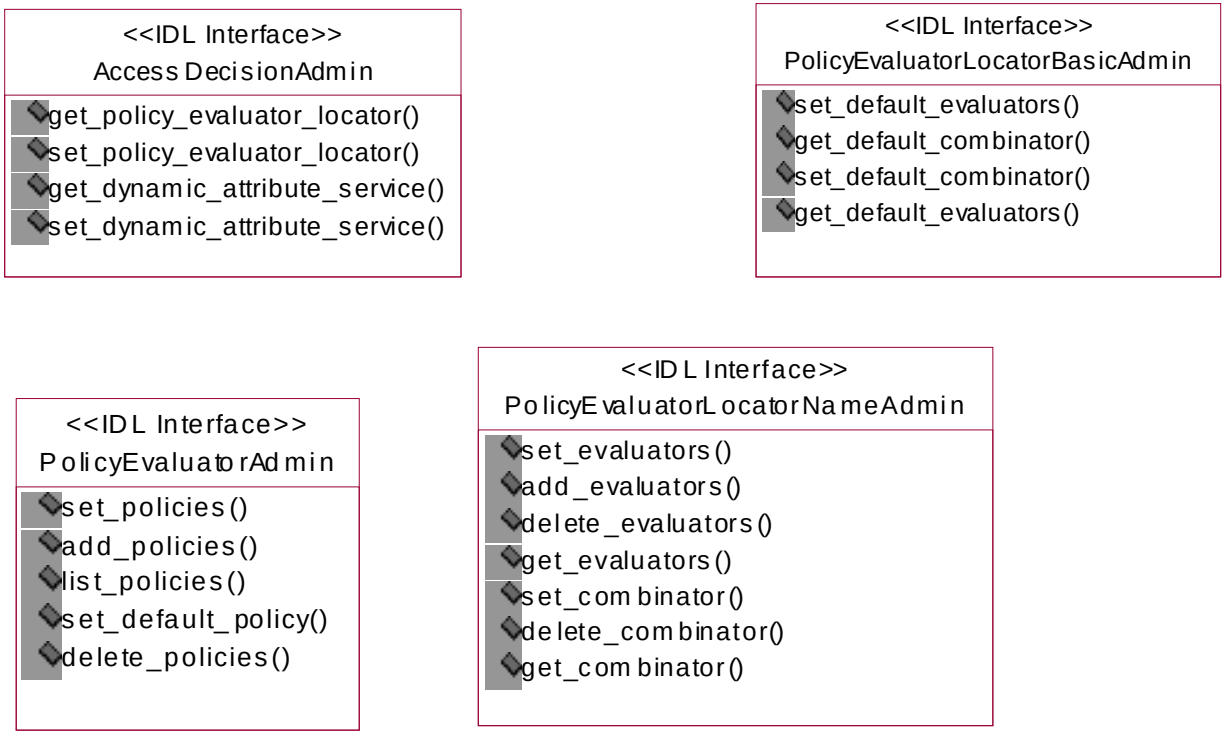
Interaction Between RAD Parts



RAD Main Parts: Runtime

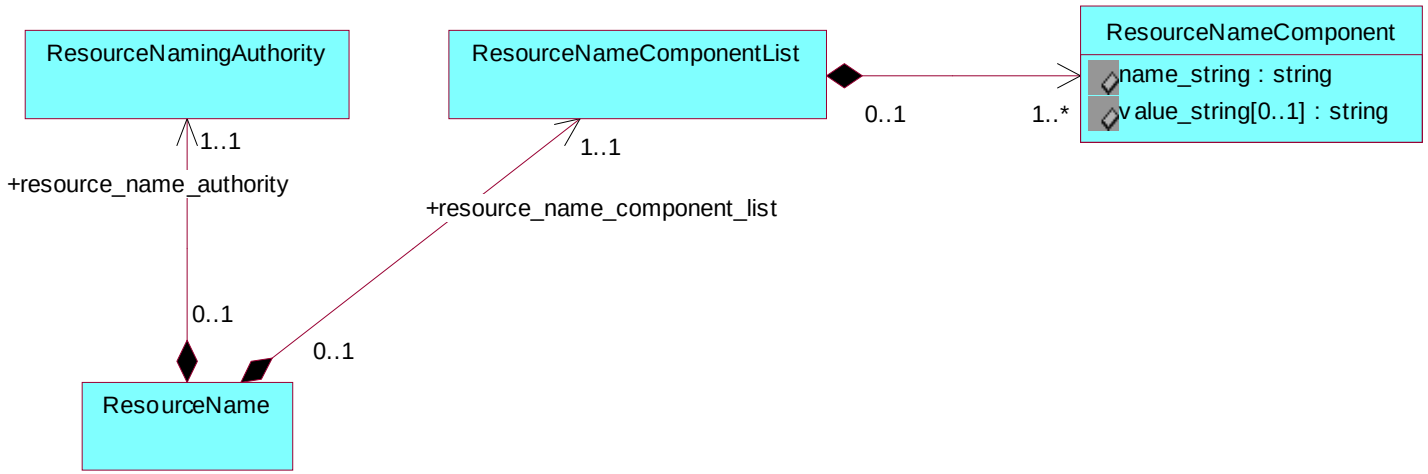
- **AccessDecision**
  - one per facility
  - the facade to the facility and a mediator
- **DynamicAttributeService**
  - one per facility, can be replace
  - updates the list of security attributes with dynamic ones
- **PolicyEvaluatorLocator**
  - one per facility, can be replaced
  - provides a list of policy evaluators and a combinator for a given authorization request
- ***PolicyEvaluators***
  - one or more per request, dynamically discovered
  - evaluate policies that they implement and return evaluation result
- ***DecisionCombinator***
  - one per request, dynamically discovered
  - calls appropriate evaluators and combines decisions from them in one grant/deny.

RAD Main Parts: Administrative



Resource and Operation Names

- **Resource names are for expressing arbitrary resources in the form of a data structures convenient for manipulations.**



- **All access operations are named**

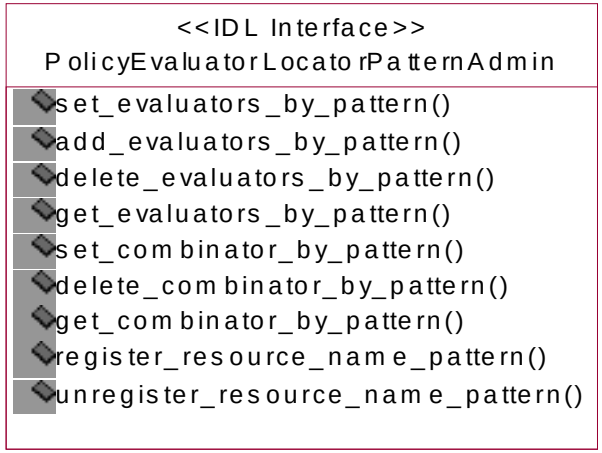
Main Design Decisions

Policies and Dynamic Attributes

- **Policies**
  - No interfaces for expressing authorization policies
  - Policies and policy engines are encapsulated in CORBA objects and can be supplied by different vendors.
- **Dynamic Attributes**
  - Request-specific factors in the form of dynamic attributes
- **CORBASEC + RAD + application service == reference monitor**

Grouping Resources and Resource Names

- **Resource are grouped using resource names**
- **Resource names are grouped using resource name patterns**



Accommodates Different Flexibility and Performance Requirements

- **Neither part of RAD have to be co-located with its clients**
  - **Only security attributes are passed, *Credentials* object is not passed**
- **Everything could be packed in the same process space as the application service, or**
- **Every single part of RAD could be a separate full-blown CORBA object with all bells and whistles.**

Design by Contract

- **Contract between the caller and the callee**
  - **Preconditions**
  - **Postconditions**
  - **Exceptions**
    - **Exception is thrown to the RAD client if something goes wrong**
- **Different treatment of run-time and administrative interfaces**
  - **Expects input errors and inconsistencies of operation invocations on administrative interfaces.**
  - **Assumes that ADO client and all RAD parts are debugged, i.e. has strict preconditions.**

Conclusions

- **RAD is useful when:**
  - **Systems from different vendors are deployed**
  - **Either granularity of access control needs to be finer, or**
  - **Access control policies are complex and relatively dynamic**
- **Authorization decisions**
  - **Made in regards to fine-grain resources**
  - **Based on factors specific to the user session, workflow, request**
- **Resource names**
  - **Abstract real resources**
  - **Could be grouped using patterns**