Distributed Application Development Technology Project

- Objective
  - Identify, investigate, prototype, and demonstrate how emerging standard technologies can be used to build distributed applications with authentication and role-based authorization using CORBA services and a Public Key Infrastructure (LDAP and X.509 certificates).
- Technology
  - Combines technology from several projects: security, Java, components, objects.
- Constraints
  - Pure Java solution, X.509v3 certificates, integration into corporate LDAP schema.

Situation

- Applications
  - Each application does its own access management in its own application specific manner
  - Currently four major COTS applications
    - Number of applications is growing
- Users
  - User base will increase from 20,000 to 50,000+ Users
- Performance
  - Role changes ripple through the system
  - Distribution of access information into new releases or builds requires between 24 and 72 hours of downtime
  - User updates require a minimum of 24 hour turnaround
  - Quick fixes to user privileges made in the application are often not updated within SSA

Target

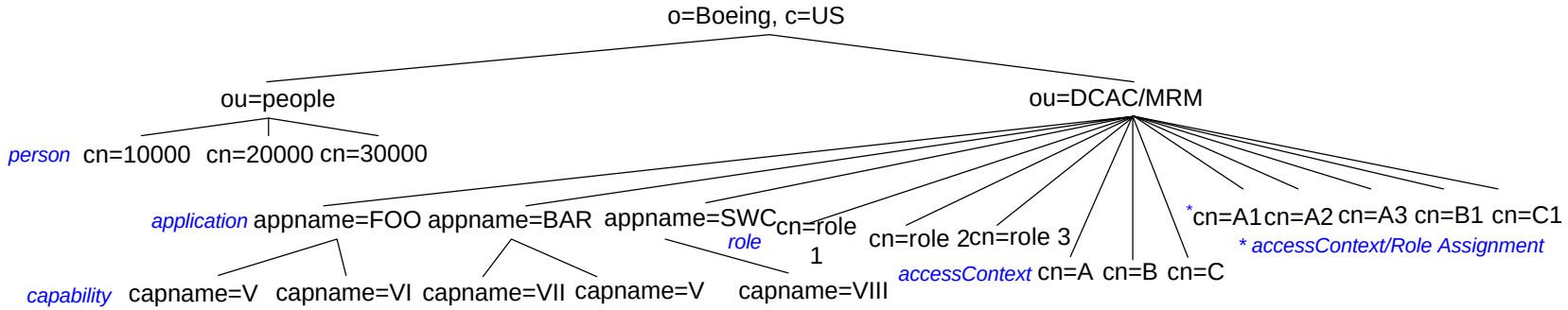
- Central management and visibility of user account and privileges using Secure System Access (SSA)
- Reduce the complexity of managing user privilege associations using Role Based Access Control (RBAC)
- Externalize user privilege associations
- Externalize the definition of roles
- Data is centrally managed and replicated
- Management interfaces used to manage data in the store
- Application interfaces used to retrieve the user's authorization policy (via CORBA, Java, DCOM)
- User's authorization policy cached on the local device for the duration of the user's session
- Take advantage of emerging technology

Prototype Summary

- Authentication
  - Used Netscape Certificate Server to generate and e-mail X509v3 certificates
  - Used the JDK 1.2 (and JCE) to read DN in certificate
- Authorization
  - Used LDAP to store and retrieve roles, capabilities and access contexts
  - Used a component based object service to access the LDAP data and make authorization decisions

LDAP Schema

Shared Services Group
Applied Research & Technology



application

appname
description owner

capability

capname appname
description owner

role

cn
appcap description o ou owner seeAlso

accessContext

cn
uniqueMember description ou owner

accessContextRoleAssignment

cn accesscontextid roleid
dataset partitioningname description ou owner

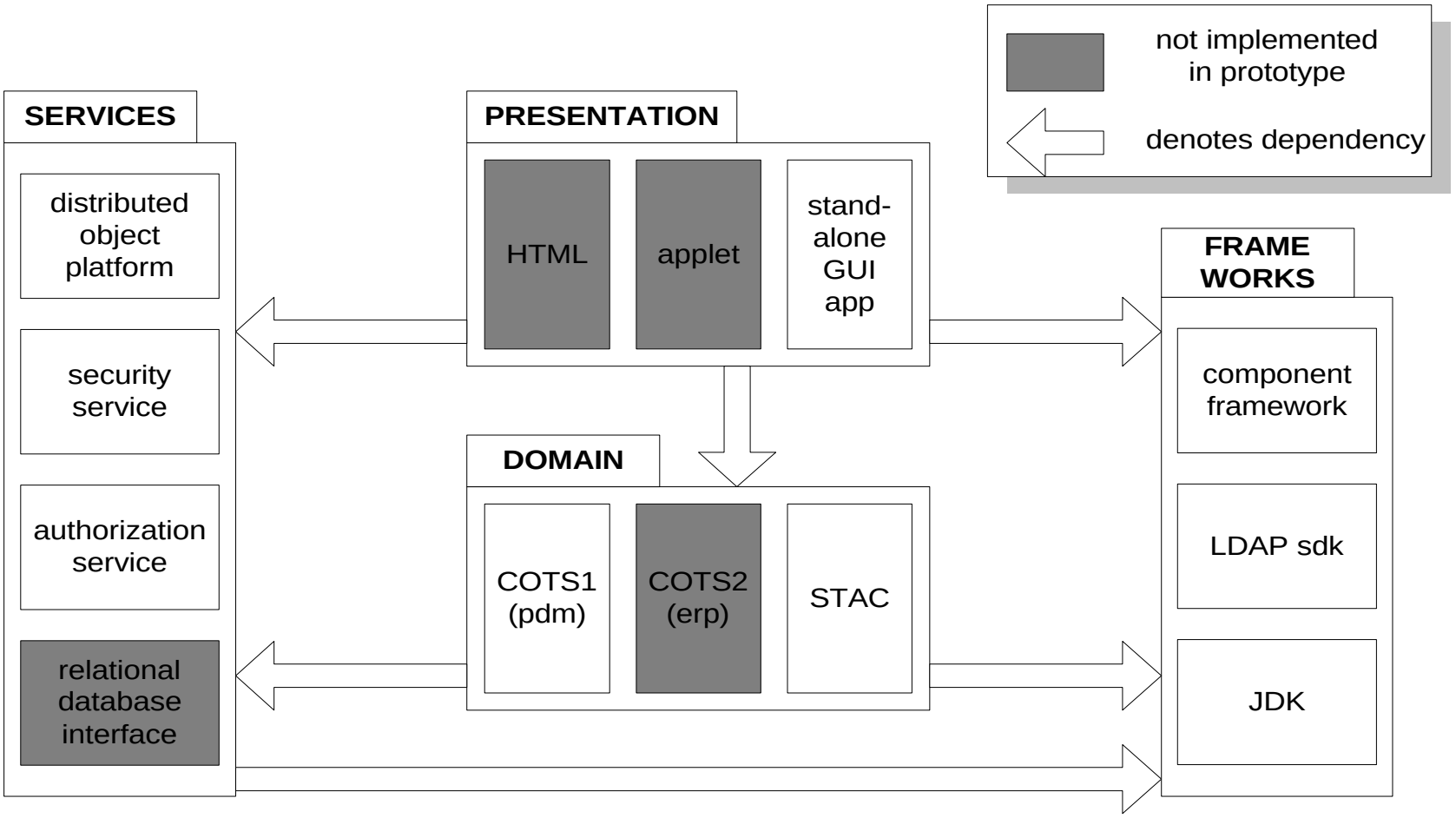
Attributes

appname	cis
capname	cis
dataset	cis
partitioningname	cis
accesscontextid	dn
appcap	dn
roleid	dn

notation: attributes above the box's internal line are required, those below the line are optional. All objects require objectClass which is not shown to reduce clutter.

BOEING®

Authorization Service Prototype Software Subsystems



Prototype: Security Abstractions-1

- **Abstractions based on *CORBA* Security model (we provisionally defined simplified interfaces).**
- **CORBA Security model probably the most flexible and extensible; mapping to Java, WWW, Microsoft models should be possible, if not straightforward. (We have not proved this.)**
- **(Note: security model mappings at this level will *not* solve security service interoperability issues.)**
- **Use of CORBA Security abstractions does not require underlying CORBA Security services or CORBA ORB. Two prototype implementations so far:**
  - **single-VM, no ORB.**
  - **CORBA ORB (OrbixWeb), no CORBA-based security.**
  - **(OrbixSSL, Orbix Security and web-based prototypes envisioned, but not implemented.)**

Prototype: Security Abstractions-2

CORBA Security:

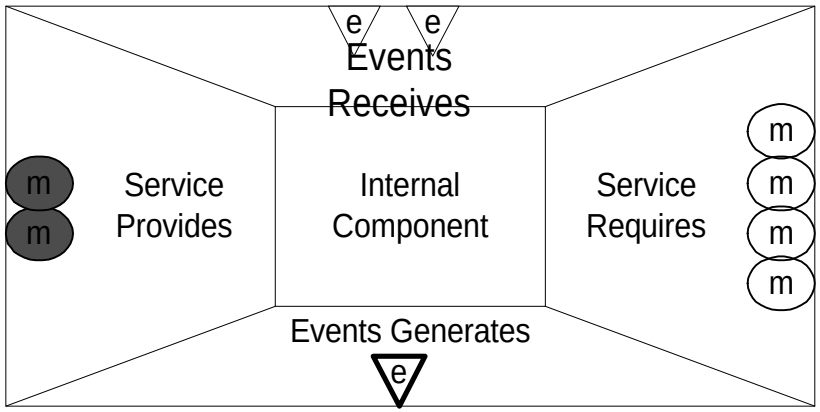
- **Users are associated with *Security Attributes*, including *identity* (e.g., user's DN name) and *privilege* attributes (e.g., DCAC "context").**
- ***Credentials* contain set of Security Attributes which will determine user's effective rights (e.g., DCAC "capabilities"?).**
- **Credentials are passed between client and target via *Current* object *(*representing current invocation thread).**
- **Credentials may be *delegated*, if there are multiple hops to target.**
- ***Security Policies* may be defined for a set of objects. (E.g., Access Policy answers the question: what are user's rights on an object given Security Attributes set in Credentials?)**

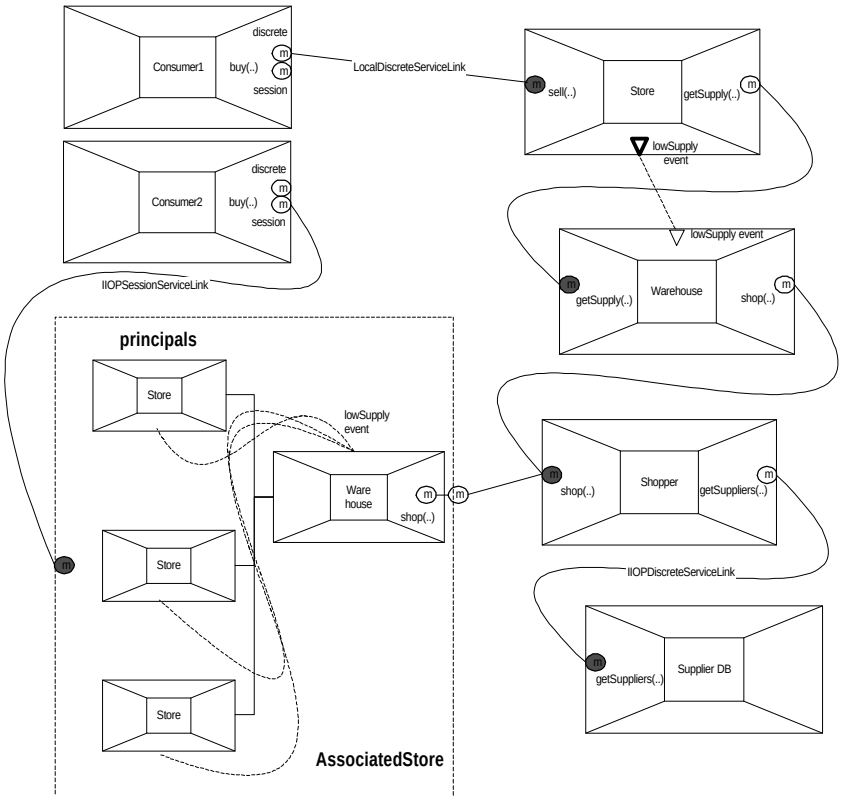
Prototype: Platform Abstractions

- Platform abstractions provide applications' view of distributed object services.
- Support retrieval of Current and Policy objects by applications.
- *Interceptors* inserted in invocation path enable communication of authorization data where underlying security services to do not support it. (E.g., SSL supports communication of identity, not other attributes.)
- Interceptors must be implemented differently depending on infrastructure (e.g., Orbix Filters). OMG is working on standard.
- Access control may be performed in Interceptors (not addressed in prototype).

Prototype: Component Model

Supports assembly from pre-built, distributed components.





BOEING

Prototype Scenario

- **User interacts with "OURSTAC" via a remote client. (Presumed to have logged on to system prior to OURSTAC startup. No further login required.)**
- **OURSTAC presents user with choice of Contexts for which the user is authorized. (*Authorization required*.)**
- **User selects a Context for the session.**
- **OURSTAC presents all the Applications for which the user is authorized in this Context. (*Authorization required*.)**
- **User selects the COTS1 application.**
- **OURSTAC connects to a remote instance of COTS1 on the user's behalf. (*Security context and delegation required*.)**
- **COTS1 presents user with choice of authorized Capabilities. (*Authorization required*.)**
- **User selects Capability ...**

Prototype APIs

Application Programming Interfaces:

- **UserAuthorizationPolicy**
  - **What contexts are authorized to user?**
  - **What applications are authorized to user in current context?**
- **ApplicationAccessPolicy**
  - **What capabilities are authorized to user for this application?**

Service Provider Interfaces:

- **AuthorizationDataAccess**
  - **Retrieves attribute and policy data from directory.**
- **CredentialsCache**
  - **Retrievals user's authenticated credentials (e.g. X.509).**