Extending the UML Standards to Model Tree-Structured Data and their Access Control Requirements

Dr. Alberto De la Rosa Algarín
Senior Principal Research Engineer
Loki Labs, Inc.

alberto@lokilabs.io
https://lokilabs.io

Introduction

- Today's Applications and Systems Built around Multiple Technologies
  - APIs, Cloud Computing, Web Services, Data Mining, etc.
- Alternative Data in Tree-Structure Standards
  - XML, RDF, JSON, OWL, etc.
- What are the Top Security Challenges?
  - Integrate Security Requirements of Existing Systems
  - Consolidate in Support of Newly Developed Application

Main Research Questions

- How do we Provide a Solution that Operates across Various Contexts?
  - Information Exchange, Databases, Web Services, etc.
  - Integrates Local and Global Security
- How do we Integrate and Support Major Access Control Models?
  - Role-Based Access Control (RBAC)
  - Lattice-Based Access Control (LBAC)
  - Discretionary Access Control (DAC)
- How Can we Make Security Policies Changes without Impacting Each Document?

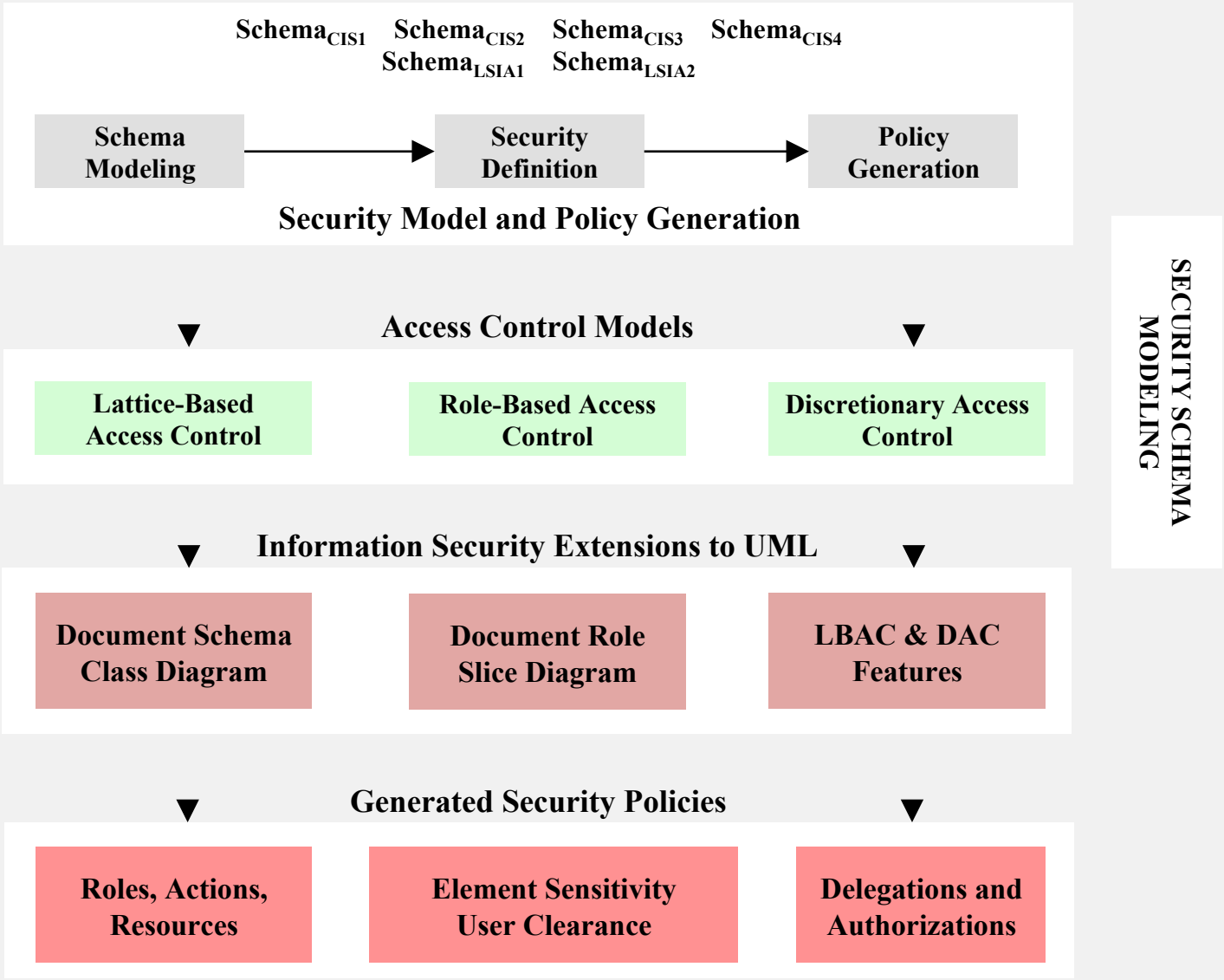
Attaining Security in Tree-Structured Documents

- Given an Application of Schemas and Associated Instances, can we:
  - Define Schemas for Security Levels, Roles, User-Role Authorizations, and Delegation
  - Augment Application's Schemas/Instances with LBAC Security Classifications (if Needed)
- Instances are Dynamically Filtered to Suit a User's Needs for an Application:
  - Based on User's Role, MAC, Delegation
  - Deliver Filtered Instance(s) to User
- 'Exploit' Security Policy Languages for Policy Generation

Why Multiple Access Control Models?

- Filter Documents (Instances) based on:
  - RBAC: Limit what Portions of Document can be Read and/or Written
  - LBAC: Security Level may Limit Portions of a Medical Record
  - DAC: Delegation of Authority for Emergent Situations
- Provide a Breath of Access Control Alternatives for Multiple Domains
  - Healthcare
  - E-commerce
  - National Security

What is the Big Picture?



Remainder of Presentation

- Brief Background
  - UML, Access Control, XML, XACML (policy)
- Security Model for Tree-Structured Data
  - RBAC, LBAC and DAC Support
- UML Diagram Extensions and Metamodel
  - DSCD, DRSD, SID, LSID, UD, DD, AD
- Security Policy Generation
- Conclusion
- Ongoing Research and Future Directions

Unified Modeling Language

- UML Diagrams Exhibit Two Views of a System's Model
  - Structural View
    - Objects, Attributes, Operations, Relationships
  - Behavioral View
    - Collaboration Among Objects and Changes to Internal States
- Different Kinds of Diagrams for System Modeling
  - Structure, Representing Components in the System
  - Behavior, Representing Series of Events that Must Happen
  - Interaction, Representing Data and Control-Flow between Components

Access Control Models

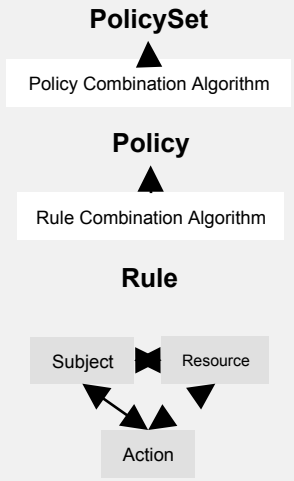
- Role-Based Access Control (RBAC)
  - Permissions assigned to Roles, Roles assigned to Users
- Lattice-Based Access Control (LBAC)
  - Policies defined and set by a Security Administrator
  - Users are not able to change their Security Attributes
    - MAC is the prime example!
- Discretionary Access Control (DAC)
  - Access to Objects is Permitted or Denied based on the Subject's Identity
  - Users are capable of passing Permissions to other Users

eXtensible Markup Language (XML)

- Defacto Standard for Information Exchange
  - Follows a tree-structure
- Provides a Common, Structured Language
  - Independent of Systems
- Data Hierarchically Structured and Tagged
  - Tags can Offer Semantics
- XML schemas
  - Blueprints for new Instances
  - Validation Agents
  - Achieved with
    - XML Schema Definition (XSD)
    - XML Schema Language (XSL)

eXtensible Access Control Markup Language

- Aims to Define a Common Language and Processing Model
  - Permits a Level of Security Interoperability
- XACML schema Provides Several Structures and Elements to Represent Policies
  - PolicySet, Policy, Rule
- PolicySets and Rules Combined by Policy/Rule Combination Algorithm
  - Permit-overrides
  - Deny-overrides
  - First-applicable
  - Only-one-applicable



Remainder of Presentation

- Brief Background
  - UML, Access Control, XML, XACML (policy)
- Security Model for Tree-Structured Data
  - RBAC, LBAC and DAC Support
- UML Diagram Extensions and Metamodel
  - DSCD, DRSD, SID, LSID, UD, DD, AD
- Security Policy Generation
- Conclusion
- Ongoing Research and Future Directions

Security Model

- Support any document format that follows a tree-structure for representation
  - XML, JSON, RDF, OWL, etc.
- Support of major NIST RBAC capabilities
  - Roles, Permissions, Assignments, Mutual Exclusion, etc.
- Support for LBAC capabilities
  - Classifications to all application schemas and their elements and define clearances for users.
- Ability to support DAC
  - Delegation of role from user to user and the ability to pass on the delegation.

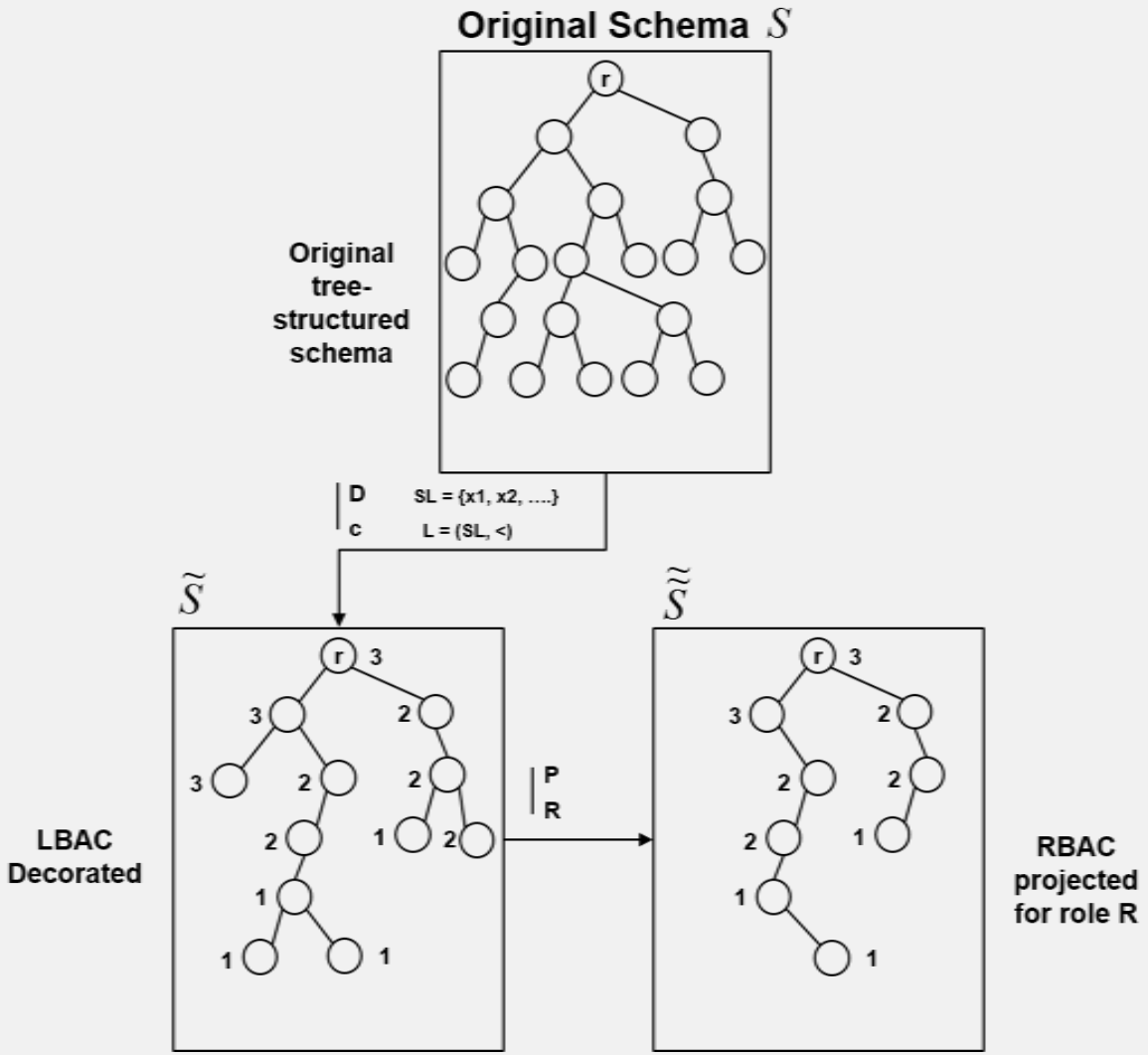
Model: Application, Schema, Instances, and Users

- Information and data is represented in a tree structure
  - Root node
  - Non-leaf nodes provide context, Leaf nodes provide value
- Schemas organize structure and content
  - Schemas can be instantiated with data
- RBAC and LBAC are orthogonal
  - Security assurance for applications that need either or both
- The User Object Contains the Security Definitions that are Relevant
  - Self-Contained Security

Model: Schema Operations for RBAC, LBAC, and DAC

- Schemas can be Altered via Specialized Operations
  - Schema Projection Operation (SPO)
    - Filters the tree into a proper subtree
    - Elements, Roles, LBAC Classifications
  - Schema Decoration Operation (SDO)
    - Extends Elements of a Tree with New Data
    - LBAC Classifications
- Altered Schemas mean Altered Blueprints
  - Instances Follow Suit
  - Effective Filtering or Decoration of Instances via Security Applied to Schemas

Example Process of SPO and SDO



Model: RBAC Security

- Schemas contain sets of elements
  - For each of the elements, a set of allowable operations are defined
    - Read, Aggregate, Insert, Update, Delete
- Permissions are operations on an element
  - P = <op, e>
- Roles are assigned operations
  - RPA = $<p_1, p_2, \ldots, p_n>$
- An SPO can be Defined over a Role
  - Identifies Elements of each Schema that the Role has Permissions Defined

Model: LBAC Security

- Lattice of Sensitivity Labels
  - Covers Mandatory Access Control (MAC)
  - Sensitivity Labels are Classifications Assigned to Elements
- Security Model's Lattice Specialized on an Ordered Set
  - SL = $\{x_1, x_2, \ldots, x_q\}$, where $x_1 < x_2 < \ldots < x_q$
    - X_q Represents the Most Secure Sensitivity
    - X_1 Represents the Least Secure Sensitivity
- Elements are Assigned Classification
  - Context Nodes Set the Minimum Security their Children

Model: DAC Delegations

- There Exists a Set of Original Users
  - Roles Assigned and Ability to Delegate
- There Exists a Set of Delegable Users
  - Can Receive Roles from Original Users
- Pass-On Delegation (PoD) Possible
  - Original Users can Determine if the Role can be Delegated One Step Further
- Delegation of Roles Only Allowed to Delegable Users
  - Starting Party is either an Original User or Delegable User (if PoD)
  - Receiving Party is a Delegable User

Remainder of Presentation

- Brief Background
  - UML, Access Control, XML, XACML (policy)
- Security Model for Tree-Structured Data
  - RBAC, LBAC and DAC Support
- UML Diagram Extensions and Metamodel
  - DSCD, DRSD, SID, LSID, UD, DD, AD
- Security Policy Generation
- Conclusion
- Ongoing Research and Future Directions

Securing Schemas with our Framework

- UML provides diagrams to model applications
  - Lack of diagrams for Security
    - Pavlich-Mariscal defined new UML diagrams for RBAC in the Metamodel layer
- Document Schema Class Diagram (DSCD)
  - UML Representation of the schema
- For RBAC, Document Role Slice Diagram (DRSD)
  - Representation of Elements, Roles, etc.
- For LBAC, LBAC Secure Information Diagram (LSID)
  - Representation of classification levels
- For DAC and Authorizations, the Delegation and Authorization Diagrams (DD and AD)

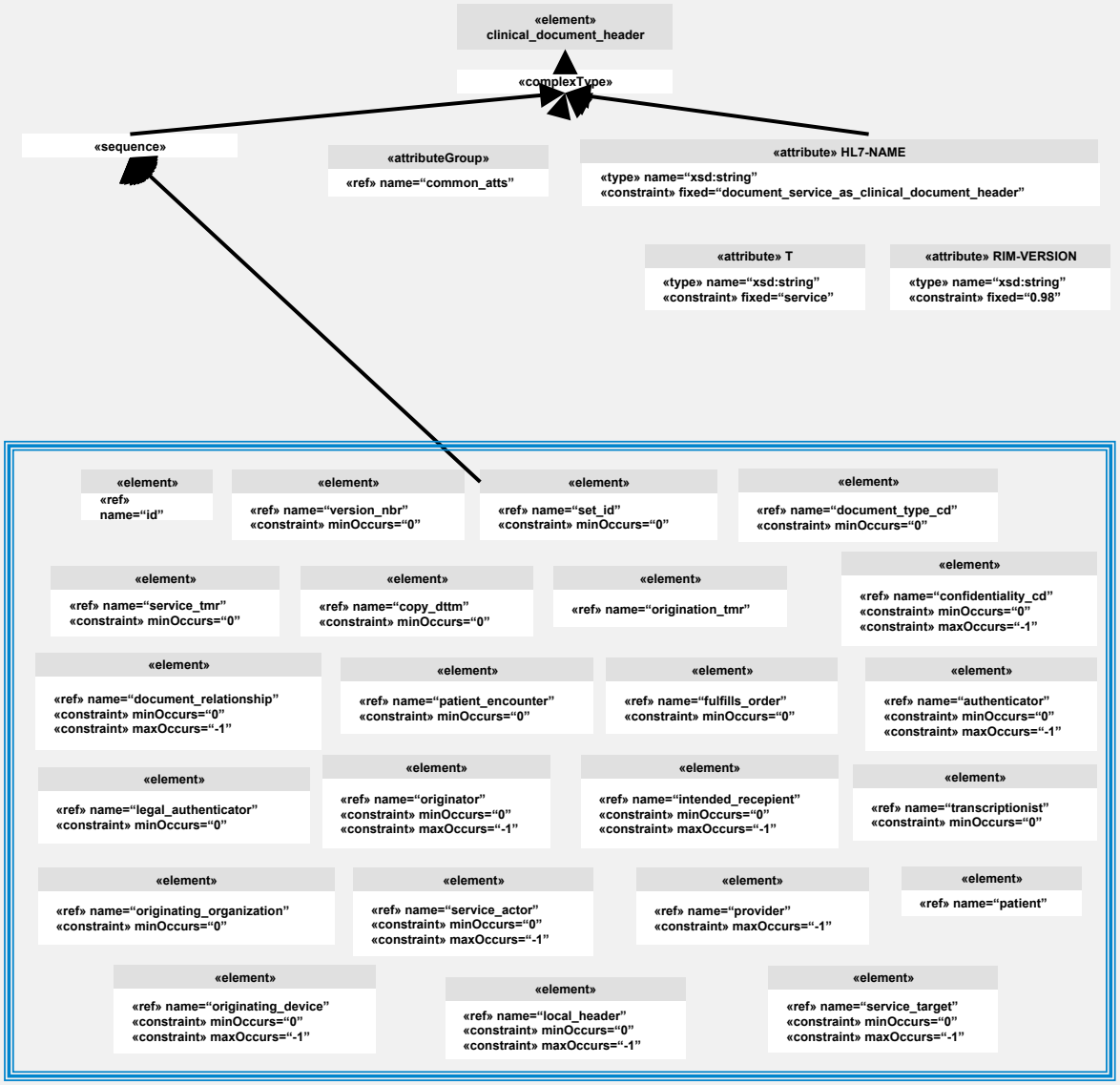
Document Schema Class Diagram (DSCD)

- An artifact that holds all the characteristics of an schema
  - Structure, Data Type, Value Constraints
- Hierarchical nature of schemas is modeled via a UML Profile
  - xs:complexType, xs:element, xs:sequence
  - Child Relations (xs:element, xs:sequene, xs:simpleType)
  - xs:extension
  - Data-type Cardinality Requirements and Constraints; type

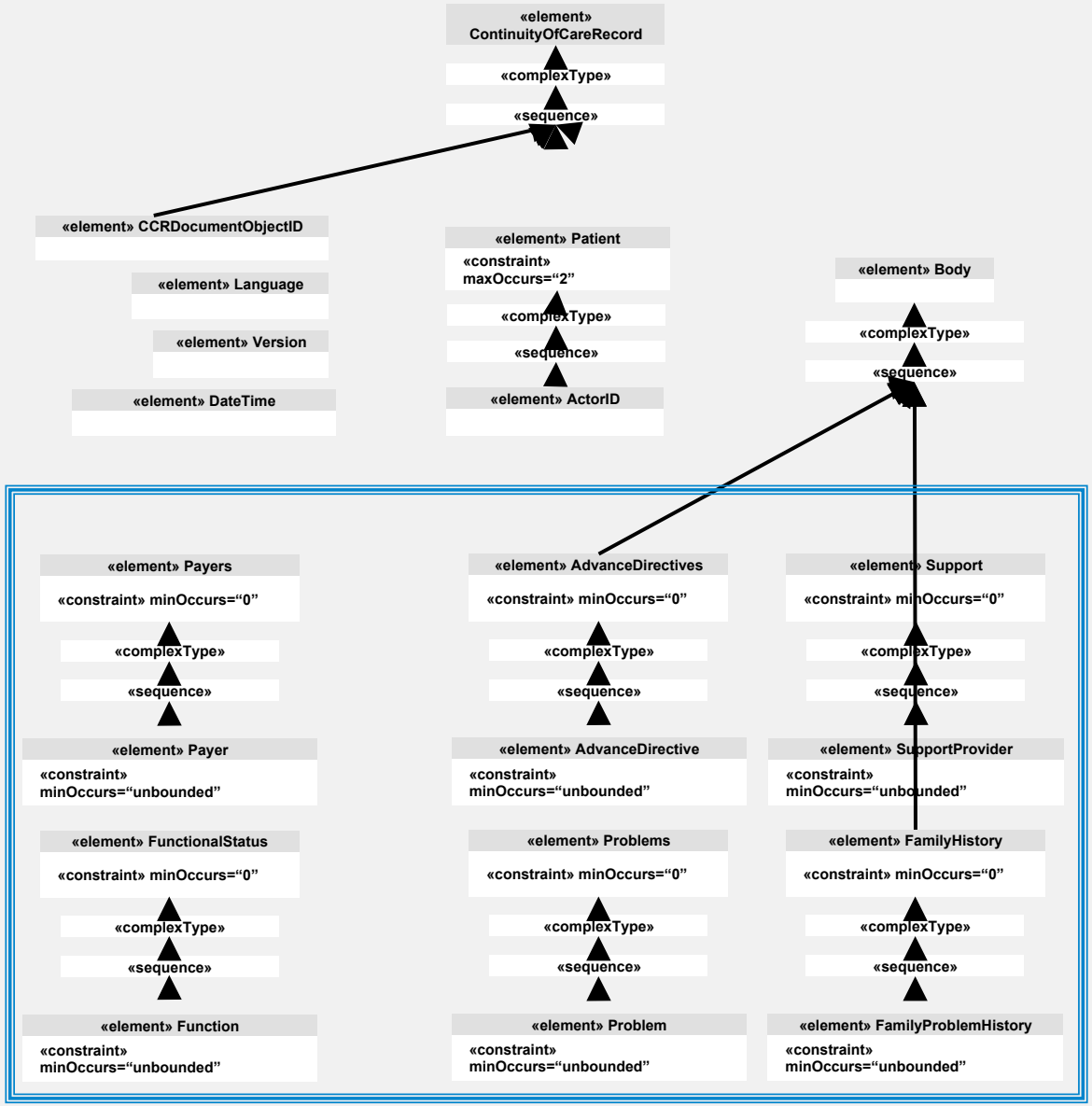
UML Profile for DSCD

Tree-Structured Document Component	XML Analog	DSCD Component
General Element	Element	UML Class
General Element Name	Element Name	UML Class Name
General Element Attribute	Generic Attribute	Stereotyped Attribute
Parent – child relationship of a schema (tree-subtree)	Tree - Subtree	UML Dependency Relationship
Complex Type of Elements and/or Attributes	XML xs:complexType	Stereotyped «complexType» UML Class
Sequential Element Order	XML xs:sequence	Stereotyped «sequence» UML Class
Aggregation of Attributes	XML xs:attributeGroup	Stereotyped «attributeGroup» UML Class
Grouping of Elements to form a complex type	XML xs:group	Stereotyped «group» UML Class
Acceptable Values for Elements	XML constraints via minOccurs, maxOccurs	Stereotyped «constraint» class member
Indirect Reference of Elements	XML ref	Stereotyped «ref» name class member
Parent – child relationship of non-named Elements	XML Element – non-named child element	UML Directed Association Relationship

Example DSCD for the HL7 CDA XML Schema



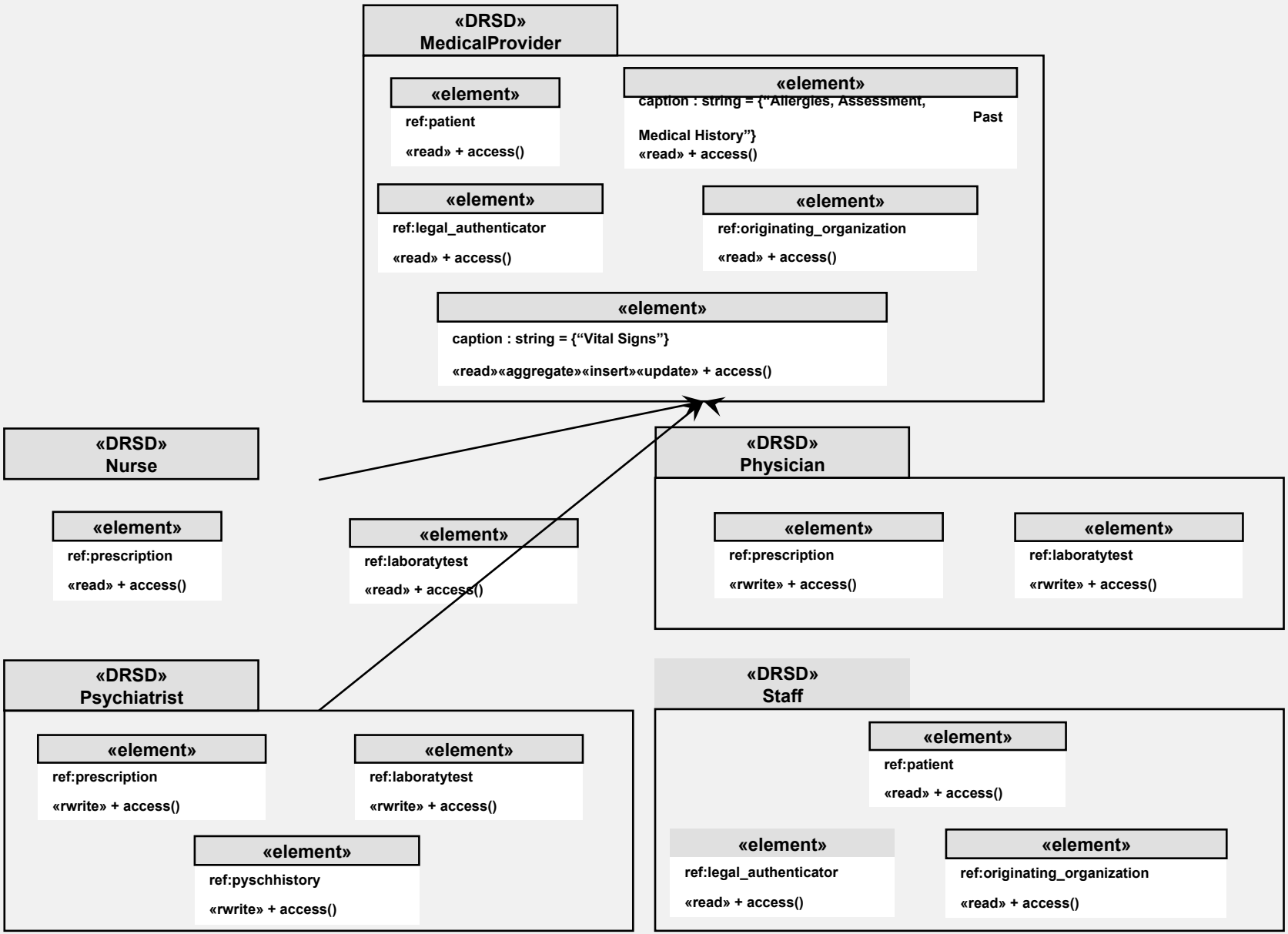
Example DSCD for the CCR XML Schema



Document Role Slice Diagram (DRSD)

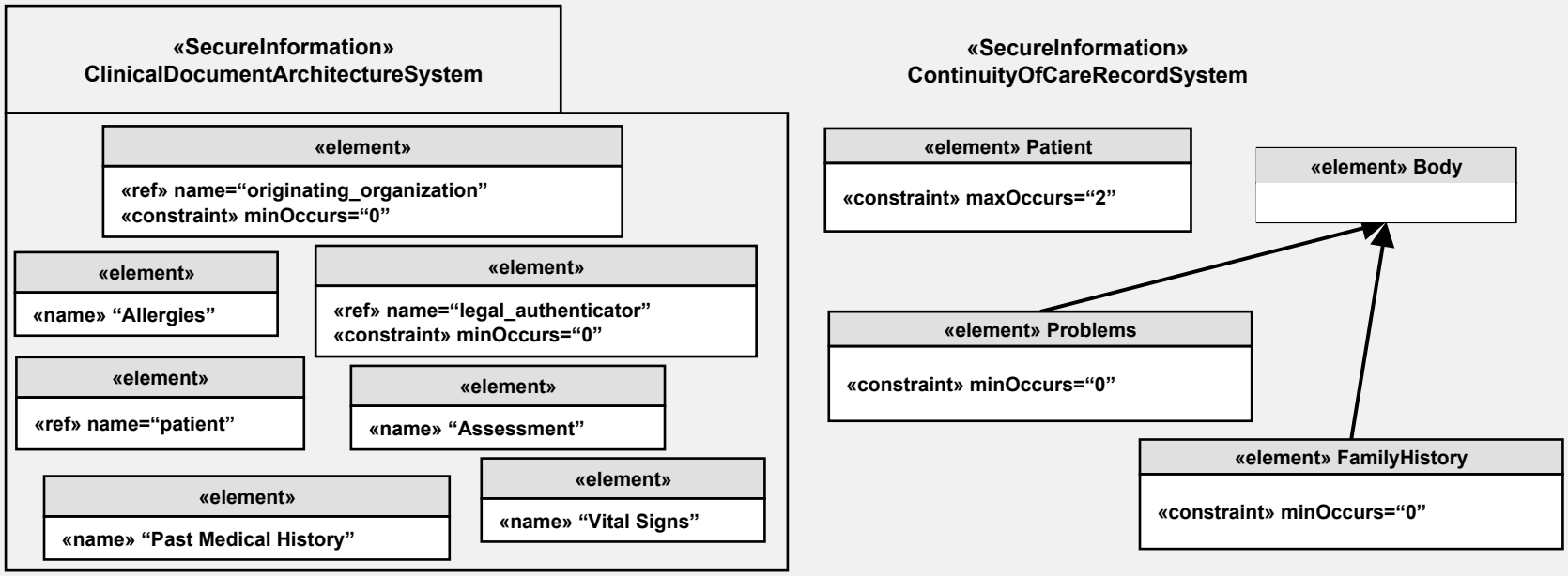
- Represents Access Control Definitions on DSCD Attributes for RBAC
  - Fine Grained Control through Security Policies and Definitions to the DSCD
    - Permissions on Documents with operations
      - Read, Aggregate, Insert, Update, Delete
- Represented in the DRSD with Stereotypes:
  - On a access() method for the class
    - «read» (non-destructive)
    - «aggregate» (non-destructive)
    - «insert» (destructive)
    - «update» (destructive)
    - «delete» (destructive)

Document Role Slice Diagram (DRSD)



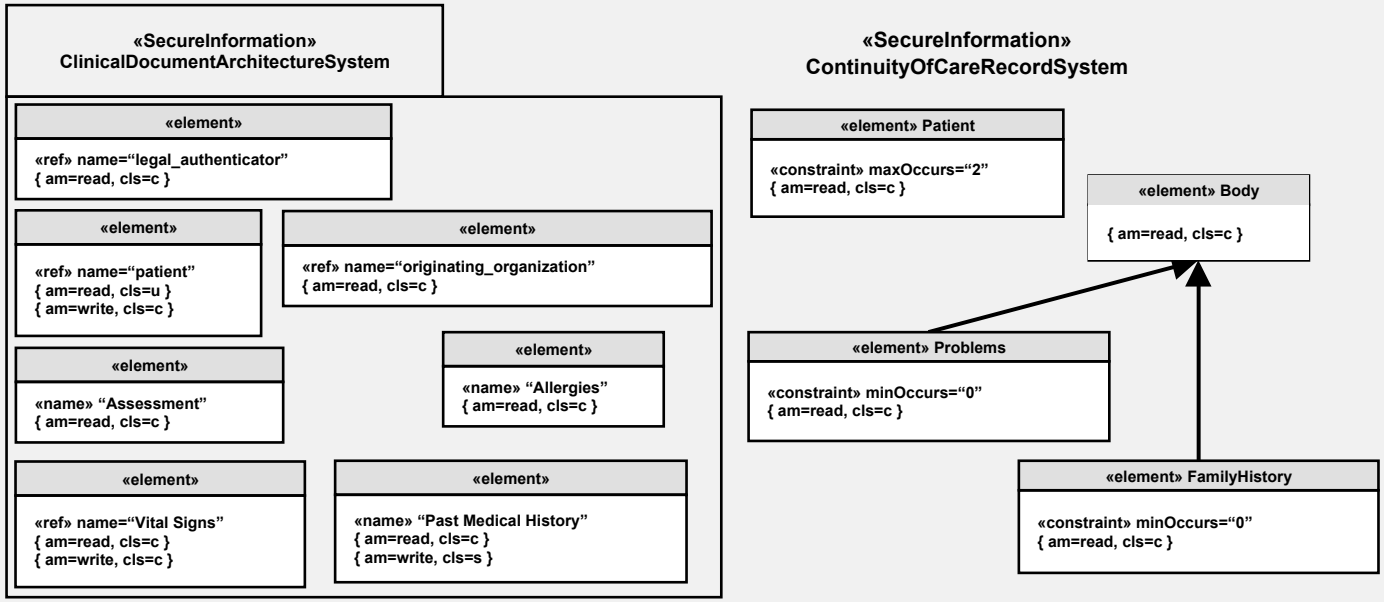
Secure Information Diagram (SID)

- Represents those elements from the DSCD that require some type of security
  - RBAC permissions, LBAC classification
- Results from the projection operation over the original schema diagram
  - Truncates the original schema by some criteria
    - Elements, Roles, Classification



LBAC Secure Information Diagram (LSID)

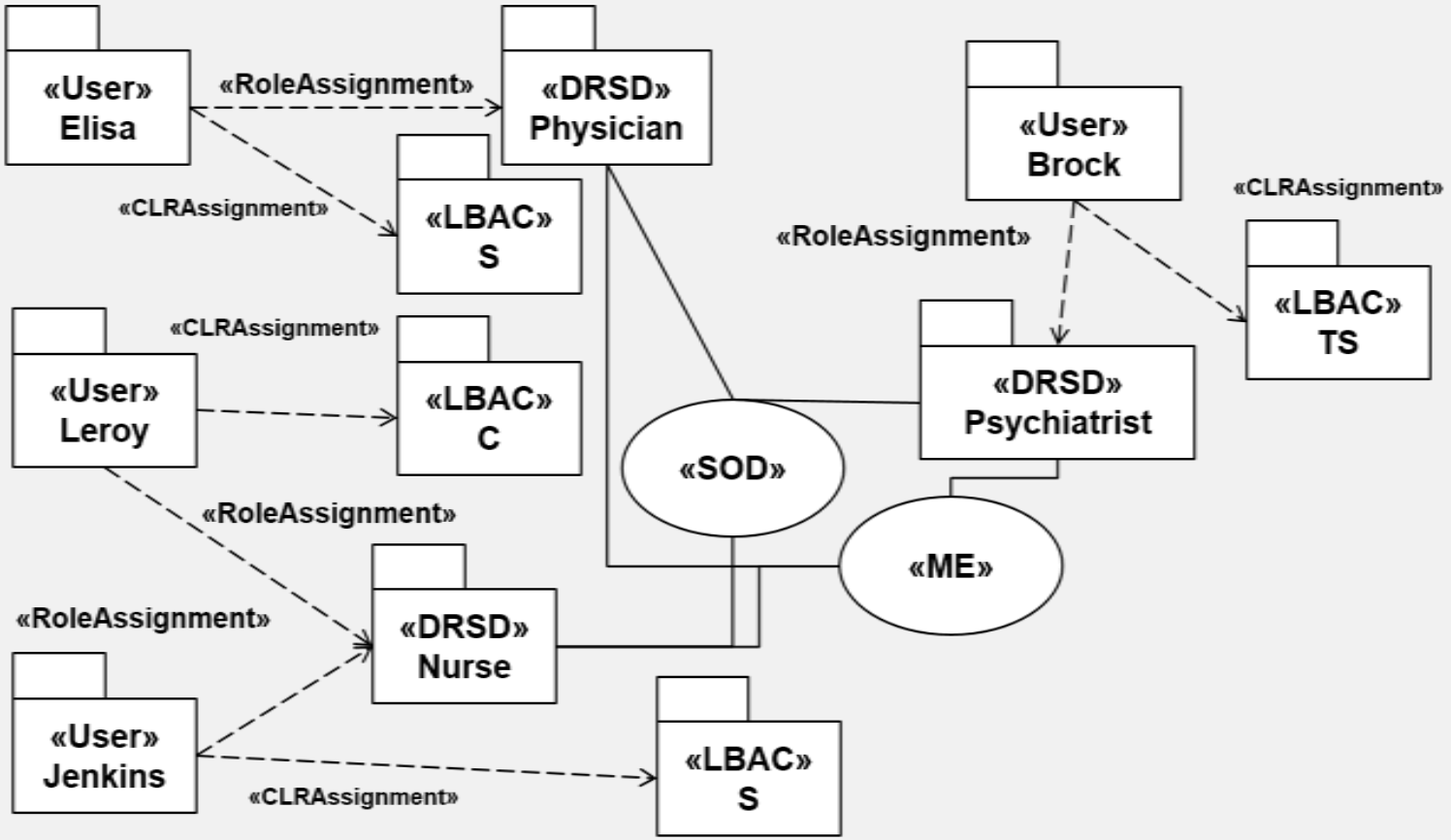
- Similar to SID
  - Represents elements that require LBAC
- UML package with the stereotype «SecureInformation» that decorates the
  - Contains all of the respective classes of elements from the schema to be secured
    - Access modes (ams), Classifications (cls)



User Diagram (UD)

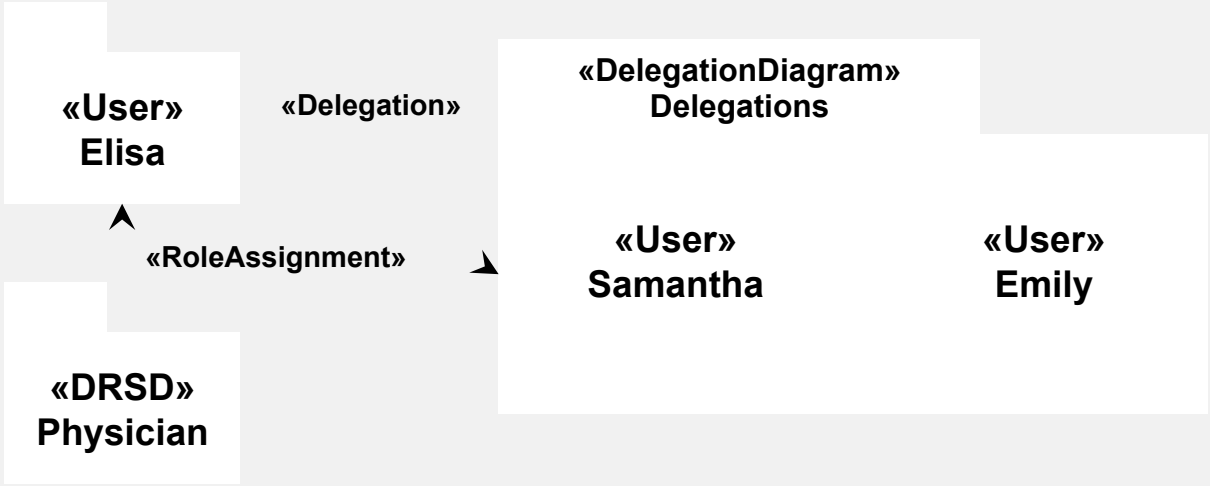
- Fulfills the need to quantify different users of the system
  - Their requirements and constraints
    - Define the users of the system whose information is to be secured.
- The interplay of users, roles and delegation permissions, clearance levels, and authorization permissions
  - Pavlich-Mariscal proposed a UML extension for users via a User Diagram.
    - We build upon it for information security

User Diagram (UD)



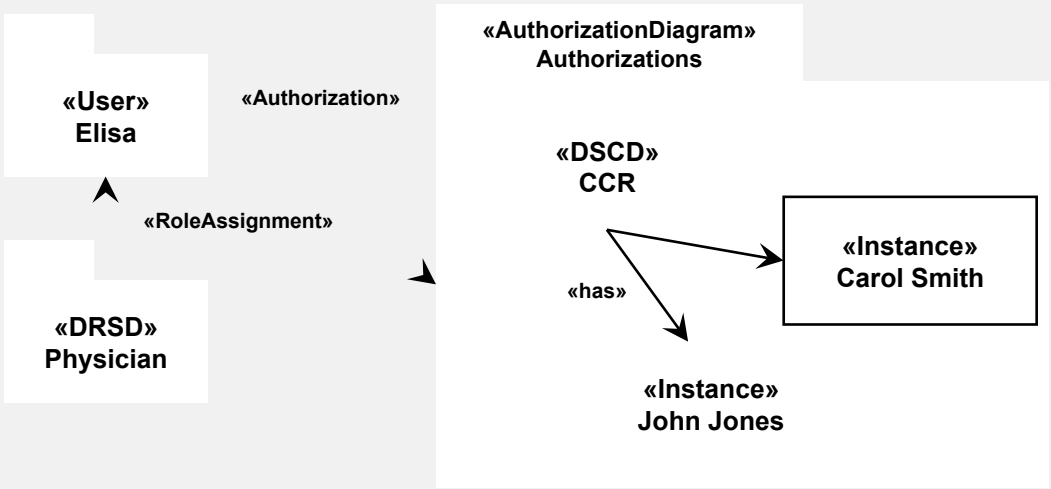
Delegation Diagram (DD)

- Captures the information of the security model's delegation
  - Mechanisms as a new UML diagram extension
- Meant to capture the concepts
  - Original user
  - Role assigned
  - Delegable users
  - Role delegation

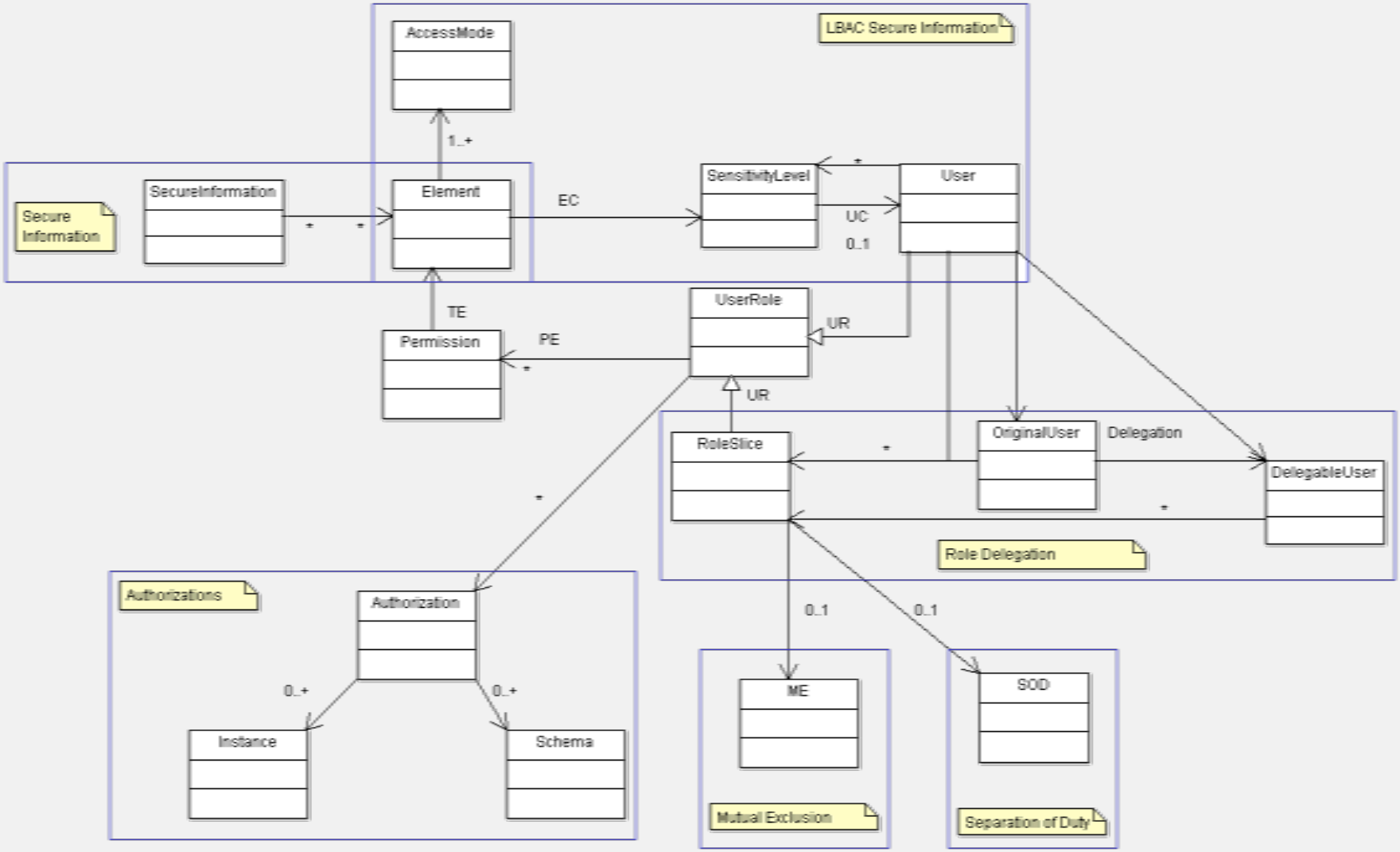


Authorization Diagram (AD)

- Illustrates a particular user/role combination
  - Connected to authorizations to particular schemas and/or their instances
- Authorizations are used to augment security by providing another layer of verification.
  - If a user has permissions defined over a specific schema, but is not authorized to it, then that user cannot perform any of the permissions.



UML Metamodel



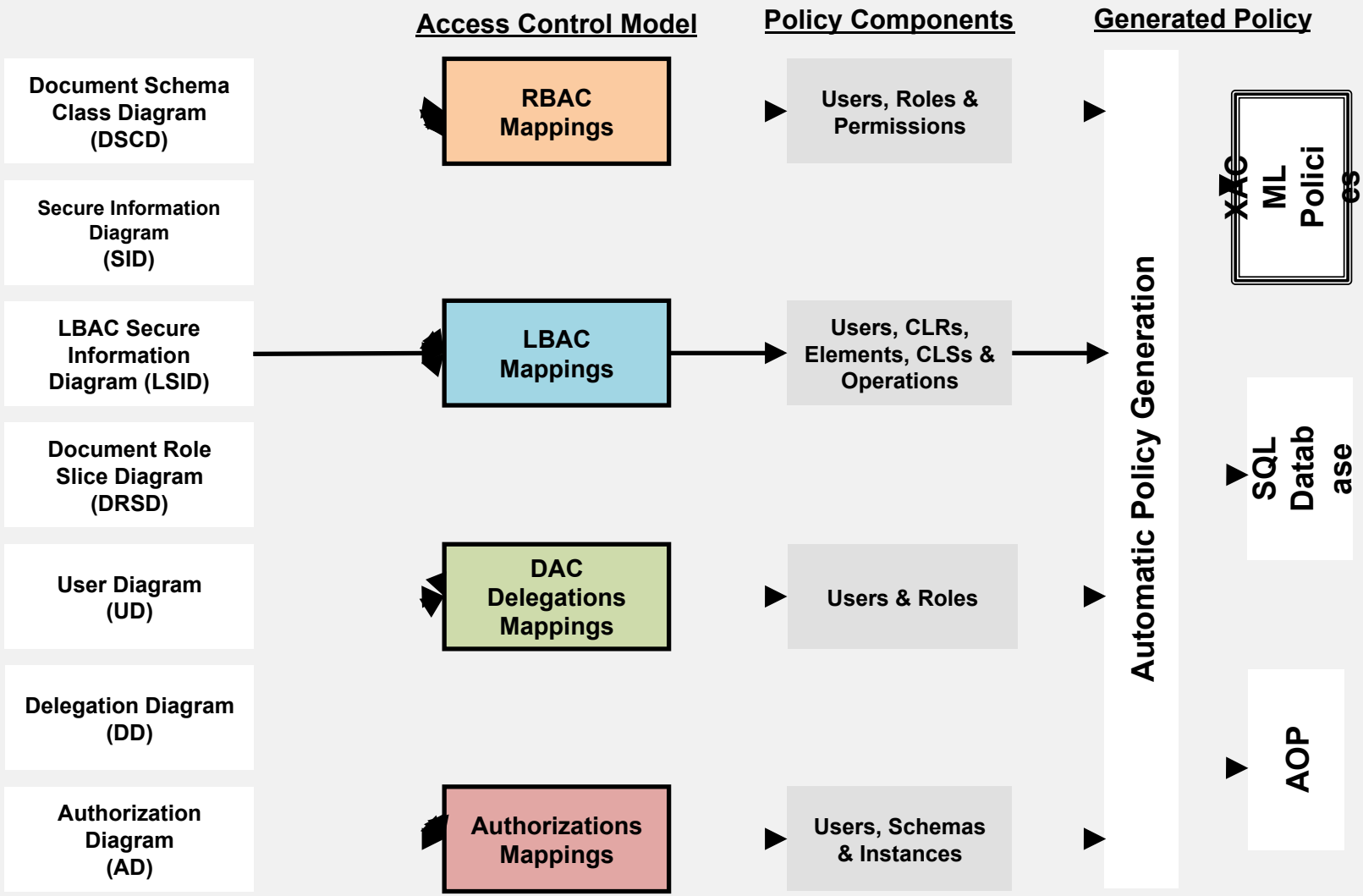
Remainder of Presentation

- Brief Background
  - UML, Access Control, XML, XACML (policy)
- Security Model for Tree-Structured Data
  - RBAC, LBAC and DAC Support
- UML Diagram Extensions and Metamodel
  - DSCD, DRSD, SID, LSID, UD, DD, AD
- Security Policy Generation
- Conclusion
- Ongoing Research and Future Directions

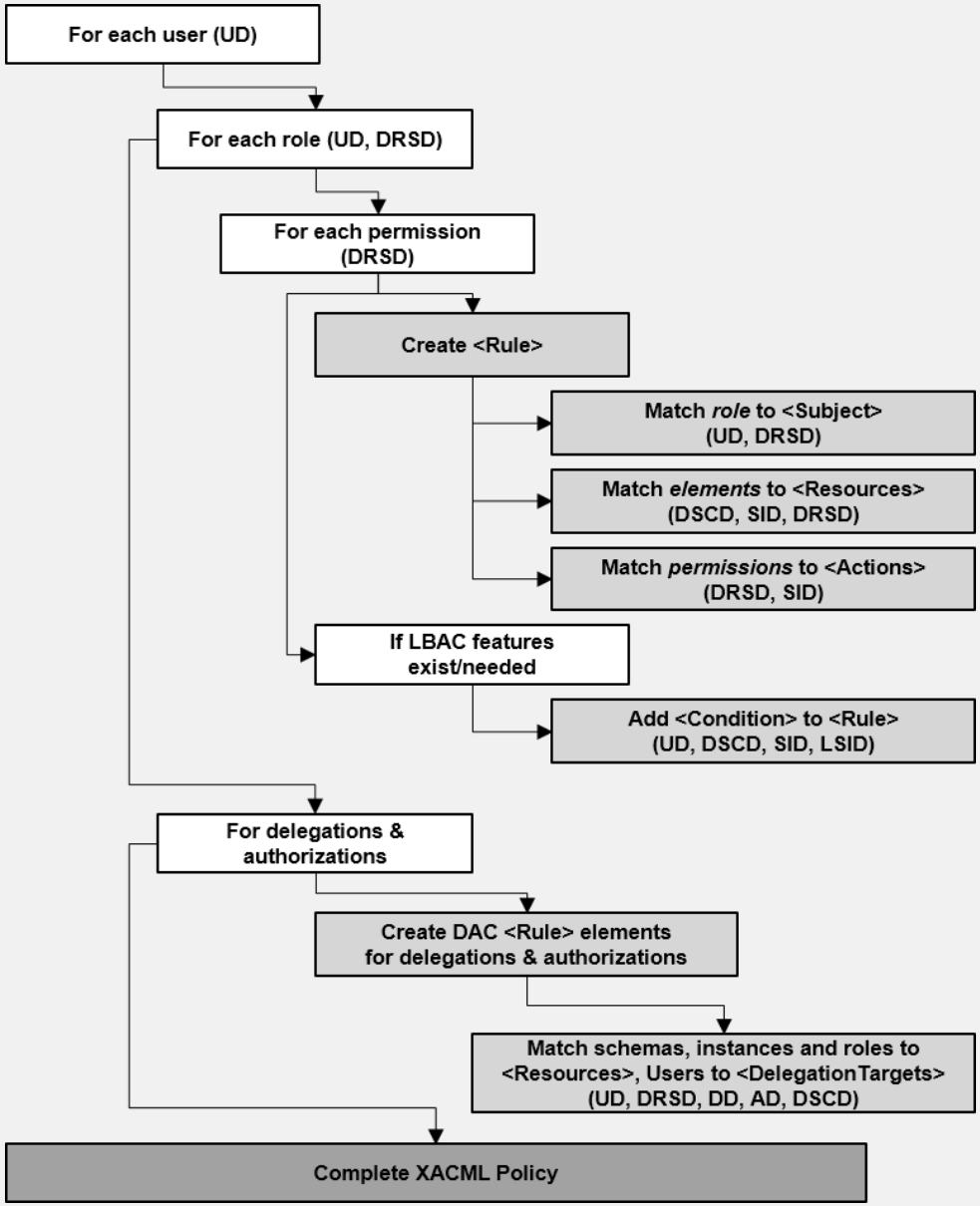
Generating Enforcement Policies

- UML has a long history for the automatic generation of code in varied languages
  - Our usage of our new UML diagrams to generate a security policy is consistent with this
- Define a set of *mapping statements (MSs)*
  - Utilized to define the conditions under which the combination of the various diagrams (DSCD, SID, DRSD, LSID, UD, DD, and AD)
  - Utilized to support the creation of respective policies for RBAC, LBAC, DAC, and authorization
- A mapping rule (MR) is defined to take the security model concepts and capabilities and the new the UML diagrams to yield a portion of the security policy

Generating Enforcement Policies



High-Level Mapping Algorithm



Mapping Algorithm Pseudo-Code

```
1.   RBAC_LBAC_DAC_XACML_generation(DSCD, SID, DRSD, LSID, UD, DD, AD)
2.   {
3.     Generate_XACML_Description_Header()
4.
5.     foreach(User as currentUser)
6.     {
7.       role_list = Find_Role(UD, DRSD);
8.
9.       foreach(role_list as currentRole)
10.      {
11.        permission_list = Find_permissions(DRSD);
12.
13.        foreach(permission_list as currentPermission)
14.        {
15.          XACML.createRule();
16.          XACML.mapSubject(UD,DRSD);
17.          XACML.mapResources(DSCD,SID,DRSD);
18.          XACML.mapActions(DRSD,SID);
19.
20.          if(LBAC)
21.          {
22.            XACML.createCondition(UD,DSCD,SID,LSID);
23.          }
24.        }
25.      }
26.    }
27.
28.    foreach(Delegation)
29.    {
30.      XACML.createRule();
31.      XACML.mapResources(DRSD,DD);
32.      XACML.mapTargets(UD,DRSD,DD);
33.    }
34.
35.    foreach(Authorization)
36.    {
37.      XACML.createRule();
38.      XACML.mapSubject(UD,AD);
39.      XACML.mapResources(AD,DSCD);
40.    }
41.  }
```

Resulting XACML Policy

```
<Policy PolicyId="ada-policy" RuleCombiningAlgId="deny-overrides">
<Description>Omitted due to length.</Description>

<Target>
 <Subjects>
 <user><id>6</id><name>Elisa</name></user>
 </Subjects>
 <Resources><AnyResource/></Resources>
 <Actions><AnyAction/></Actions>
</Target>

<Rule RuleId="simple-RBAC+LBAC-rule" Effect="Permit">
 <Target>
 <Subjects>
 <role><roleID>5</roleID><roleName>Physician</roleName></role>
 </Subjects>
 <Resources><element>
 <elementID>el-3</elementID>
 <elementName>Past Medical History</elementName>
 </element></Resources>
 <Actions><operation>
 <operationName>insert</operationName>
 <opAccessMode>write</opAccessMode>
 </operation></Actions>
 </Target>
 <Condition>
 <Apply FunctionId="…:integer-greater-than-or-equal">
 <Apply FunctionId="…:integer-one-and-only">
 <AttributeValue DataType="…#integer">Secret</AttributeValue>
 </Apply>
 <AttributeValue DataType="…#integer">Secret</AttributeValue>
 </Apply>
 </Condition>
</Rule>
```

```
<Rule RuleId="simple-delegation-rule" Effect="Permit">
 <Target>
 <Subjects>
 <user><id>6</id><name>Elisa</name></user>
 </Subjects>
 <Resources>
 <Roles><role>
 <roleID>2</roleID><roleName>Physician</roleName>
 </role></Roles>
 </Resources>
 <DelegationTargets>
 <user><id>30</id><name>Samantha</name></user>
 </DelegationTargets>
 </Target>
</Rule>

<Rule RuleId="simple-authorization-rule" Effect="Permit">
 <Target>
 <Subjects>
 <user><id>6</id><name>Elisa</name></user>
 </Subjects>
 <Resources><Schemas><schema>
 <schemaID>4</schemaID>
 <schemaName>Schema 4</schemaName>
 </schema></Schemas>
 <Instances><instance>
 <instanceID>4,2</instaneID>
 <instanceName>Carol Smith Health Record</instanceName>
 </instance></Instances></Resources>
 </Target>
</Rule>

</Policy>
```

Conclusion

- Presented a Security Framework
  - Addressed the Issue of Providing Information Security in Systems with Tree-Structured Documents
  - Utilize Security Policies defined after the different Access control models
    - Support for RBAC, LBAC and DAC
- Not enough to utilize the security requirements of the newly developed system
  - Security Definitions and Requirements of Constituent Systems must be Considered

Ongoing Research and Future Directions

- Non-orthogonal RBAC and LBAC
  - Clearance assigned to both users and roles
- Support of other access control models
  - ABAC (Attribute-Based Access Control)
  - Support of Compartments for RBAC
- UML Profile for other specialized document formats
  - JSON
  - RDF serializations
  - OWL
  - Automatic Creation of DSCD
- Policy generation in other languages and more efficient algorithm
  - Deployable to databases
  - Development Framework Policies
  - Decoupled systems from a security architecture