Centralized Authorization Service (CAuthS) or Authorization as a Service (AuthaaS)—A Conceptual Architecture

Pranab Das
Principal Architect, Monocept
Cyber Pearl, Hitech City
Hyderabad, India

Abhinav Das
Student, Manipal University Jaipur
Dehmi Kalan, Sanganer
Jaipur, Rajasthan, India

ABSTRACT
Absence of architecture to describe how to implement authorization[1] as a centralized service, in a way similar to authentication[2], has been causing redundant deployment of computing resources, lack of standard practices, and never-ending learning curve in maintaining proprietary or ad hoc authorization solutions. The paper develops an architecture, which focuses on centralization of authorization, to be called Centralized Authorization Service (CAuthS) or Authorization as a Service (AuthaaS), when deployed as a service, and is targeted to substitute platform-based ad hoc authorization solutions.

General Terms
XACML: eXtensible Access Control Markup Language is a standard maintained by OASIS [1].

ACL, *RBAC*, and *ABAC:* Access Control List, Role-Based Access Control, and Attribute-Based Access Control are commonly known patterns in the domain of authorization ([2]; [3]; [4]]).

User Access Control (UAC): An enterprise or federation-scoped authorization (access control) service discussed in the paper. In addition to access control, validation of action by the principal (user) is included in the meaning of authorization.

UUID: Universally Unique Identifier is defined by the Open Software Foundation [5]. It should guarantee uniqueness of principal within an enterprise-scoped or a federation-scoped UAC service.

Identification, *Identification Provider (IdP):* Identification a.k.a., authentication, validates credentials of principals and returns results as Boolean responses. Issuing secured tokens is often included in the task-list of the IdP [6]; however, the latter responsibility may be gainfully shared by the SP particularly if per-transaction tokens[3] are warranted for security reasons.

Core Concern, *Crosscutting Concern:* Core business gives rise to logic known as the core concern; other, supportive logic are known as crosscutting concern [7]. However, the paper will take a relative view on this: if individual concerns in the set $C = \{c_1, c_2, \ldots\}$ containing all of them are implemented separately then it will be assumed that the relation, crosscutting X into C will be deemed reflexive, i.e. $c_i X c_j \Leftrightarrow c_j X c_i \forall c_i, c_j \in C, c_i \neq c_j$. It practically means that authorization being a crosscutting concern for some business service also means and is meant by the business service (or a part of it) acting as crosscutting concern for authorization.

Service Provider (SP): UAC resources are controlled by the Service Provider (SP), which provides access to identified principals, subject to nature and extent determined by the UAC.

Principal: Entity seeking access over UAC resources. Also vide infra.

Subject: Entity allowed access over UAC resources. Also vide infra.

Keywords
Computer security, access control, authorization, context type mapping, strategy

1. INTRODUCTION
Since authorization is not currently being treated as enterprise-wide, cross-domain, federated (cross-enterprise), or cloud-based service, it necessitated deployment of redundant computing resources to run multiple services in parallel. Generally, authorization includes following component-services: (a) collaboration with authentication to ascertain identity of a principal; (b) manage data necessary to perform authorization and providing a service-interface to the SP; and (c) make use of such patterns as ACL, RBAC, or ABAC, etc., which would cover current and foreseeable future requirements of managing the core problem of authorization without breaking codebases[4]. Many of such functions as are currently performed by individual services, can be performed by a centralized service that would scale computing resources better.

It is noteworthy that XACML evolved as a standard of authorization rather than to facilitate development of a centralized authorization service. Some criticized it for its verbosity and complexity [8]. Nevertheless, a standard is needed to build CAuthS or AuthaaS. Rest of the paper focuses on a conceptual, rather than a descriptive, standard. As far as the following architecture is concerned, ad hoc JSON or

[1] Authorization has been defined differently by many ([20]; [21]; [22]; etc.). In this paper, it is, primarily, granting access to a principal over a view of a UAC resource. However, it may also include validation of user actions.

[2] Similar to authorization, authentication (for the purpose of this paper it is same as or close to assertion of identity) has many definitions ([23]; [24]; etc.) Here, the term means asserting identity of a principal. Single Sign-On (SSO), Cross-Domain SSO, and Federated SSO ([25]; [26]; [27]; [28]; [24]) have contributed to centralization of this service.

[3] Vide DUKPT [29], which is a way to address vulnerability of sessions that use a single token for all transactions associated with it.

[4] Brittleness of codebase is defined as property that requires disproportionately extensive modification required by a change in the requirement, cf. Kremenek et al. [30]. The current usage is more in line with sense used by Keenan and Steele [31] though.

ASCII strings ([9]; [10]; [11]) may also be used in place of XML.

2. ELEMENTS OF DESIGN

Essential elements of CAuthS, as depicted by Figure 1, are following.

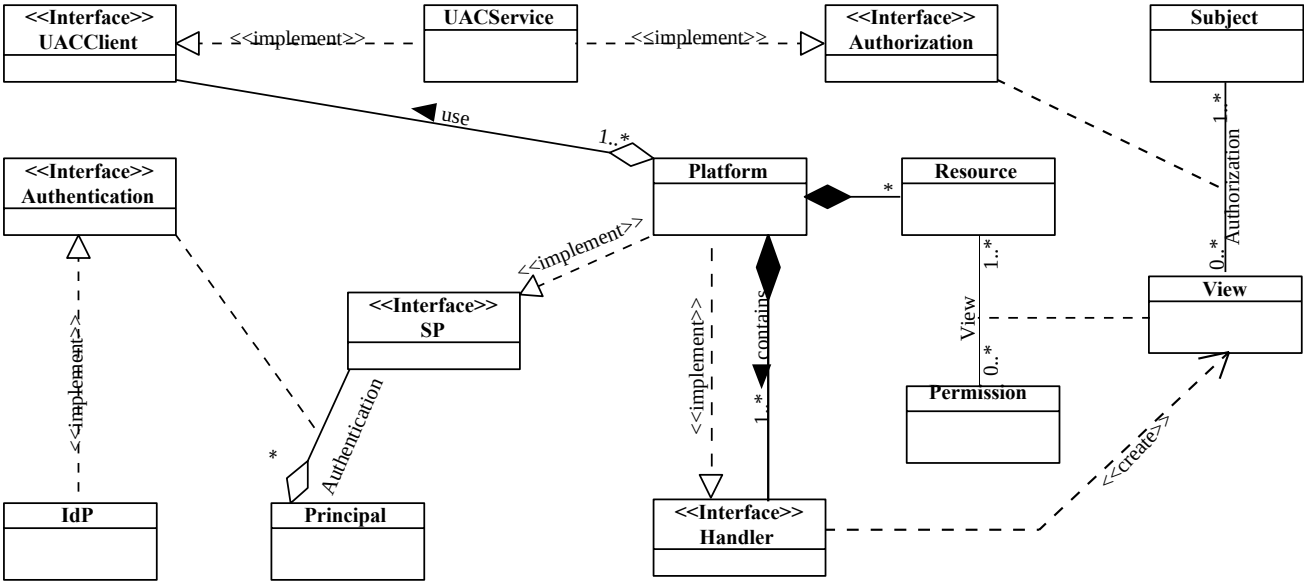

Figure 1: UMLclass diagram showing static relationship between CAuthS and its ecosystem

2.1 CAUTHS OR AUTHAAS

CAuthS or AuthaaS is described by the following classes.

2.1.1 Principal

A principal is an actor that requires access to a UAC resource by means of the SP interface provided by the platform containing such resource. It, therefore, needs authentication and authorization.

2.1.2 Platforms and UAC Resources

Platforms implement SP for the benefit of principals; they also contain UAC resources, views of which are provided to principals via SP. Such resources are components of platforms that would normally consociate with permissions, including none, thereby defining views[5] of it that are made accessible to principals. To create views, platforms would implement handlers. The set views Ψ may be conceived as the binary relation[6] [12] 'for the purpose of'[7] from the set of UAC resources Υ to the set of permissions Ξ, which is described by (1) below.

(1) $\Psi \subseteq \Upsilon \times \Xi$

Here, the first component of each vector[8] $\nu \in \Psi$ is arbitrarily taken to be a UAC resource and the second component is, permission.

2.1.3 IdP (Identity Provider)

IdP authenticates a principal to the platform.

2.1.4 UAC

UAC implements a client interface to be used by the platform. It also implements an authorization interface that determines a subjects' accessibility over a view.

2.1.5 Subject

Subject (a.k.a. role in RBAC) represents a set containing all smallest grains of assignees that can be authorized and a convenient indirection that allows loose coupling between principals and views. Subjects S and views Ψ would define relation'is authorized to' or$A$, also to be called the authorization-set.

(2) $A \subseteq S \times \Psi$

A relation 'may act as' orRo would exist from principals P into subjects S, defined by (3) below[9].

(3) $Ro \subseteq P \times S$

The relation (3) means that one or more principals can be mapped to one or more subjects (and vice versa) while (2) would determine corresponding views in the following steps:

(a) Given $p \in P$, find $Ro_p \subseteq Ro$ such that $(\{p\} \times S) \cap Ro = Ro_p$.
(b) For all $s_j \in Ro_p, j \in S$ find $\{a_k\} \subseteq A$ such that $(\{s_j\} \times \Psi) \cap A = \{a_k\}$.
(c) Views $\{\psi_p\}$ assignable to p would be given by $\{\psi_p\} = \{\psi | \{(s, \psi)\} = \{a_k\}, s \in S \wedge \psi \in \Psi\}$.

2.1.6 Extension

Infrequently though, subjects and views may be conditionally associated; for example, a subject and a view may be connected via environmental conditions[10] ([13]; [3]). It may be trivially observed that such conditions would affect dimensions of vectors represented by (2) above. Thus, if T is

[5] A view may be taken as such aspect or aspects of a UAC resource as can be made accessible to a principal.
[6] In this paper all future relations will be deemed to be binary ones.
[7] For example, if $\upsilon \in \Upsilon$ represents a 'chair' and $\xi_1, \xi_2 \in \Xi$ represents 'sitting on' and 'standing on' respectively then $\upsilon A \xi_1$ would mean 'chair for the purpose of sitting on', etc.
[8] These are *n*-tuples, including ordered pairs [33].

[9] By not narrowing down the relation from P into S to a function, Ro would fulfill both many-to-many, which is typical with RBAC ([34]; [35]), and many-to-one, typical with ABAC ([36]; [37]), schemata.
[10] Use case 1: An employee may enter the server room only between 8am and 5pm. Use case 2: A user may access their web-based account only when a request originates from Princeton, NJ.

the set of time intervals such as[11]{[8,17], ... } then A may be redefined by(4) below.

(4) $A \subseteq S \times \Psi \times T$

The relation (4) above may be observed to address requirement of use case 1, vide footnote 10.

2.1.7 Sequencing

Sequence of calls, in view of which the architecture has been defined, is following.

(a) A principal requests for a UAC resource to the platform via SP.
(b) The request is asserted by the IdP to have been originated from an identified principal[12].
(c) Authorization-profile of the principal is requested and fulfilled via UAC client that may comprise of data related to zero or more subjects. UAC may create profile from persistent data; it may also obtain data from other processes that make such data available to it on the fly.
(d) Based on profile-data, the platforms would select handlers and parameterize them in order to create views that are authorized for the principal. Thereafter platforms would return such authorized views to principals via SP.

2.2 Profile

Not all mutually exclusive, subject-profiles[13] Π_j's are necessarily subsets of the authorization-set A and/or are indexed by the set S of subjects, vide(5) and (6) below[14].

(5) $\bigcup_{j \in S} \Pi_j = A \Rightarrow \Pi_j \in \mathcal{P}(A) \forall j \in S$

(6) $\Pi_m \cap \Pi_n \supseteq \phi \forall m \neq n \wedge m, n \in S$

Thus, all subject-profiles, together, constitute the authorization-set; additionally, distinct subjects may not always have distinct subject-profiles. One or more subject-profiles would aggregate in a profile, which relates to a principal, vide supra (under Subject) and Figure 2.

Profiles may be conceived as mementoes [14], each holding a complex message from UAC to undifferentiated platforms—any participating platform should be able to process profiles to obtain authorization-related data. Such messages should have semantics that need be understood only by the recipient, not even UAC, in general (only an administrator of UAC data related to an SP may need to understand semantics of data related to such SP). A minimalist schema of profile is depicted by Figure 2.

The schema need not be the same for all platforms or UAC resources; each platform or resource may have their own schema or reuse a common schema. Details of the sample schema in Figure 2 are following.

2.2.1 Principal Id

Principal Id uniquely identifies a principal.

2.2.2 Profile

A profile is a comprehensive set of authorization-data related to a principal. Notably, principals P and the set $\mathcal{P}(A)$ would have a functional relation [15] h given by (7); also, vide §2.2. Thus, given a principal p, an element Π_p of $\mathcal{P}(A)$can be uniquely determined. This is because a power set would contain all subsets, including possible unions.

(7) $h: P \to \mathcal{P}(A) \Leftrightarrow \exists \Pi_p \in \mathcal{P}(A) \forall p \in P | h(p) = \Pi_p$

Clearly, the profile Π_p would be union of zero or more subject-profiles s, vide (8) below.

(8) $\Pi_p = \bigcup_{s \in p} \Pi_s$

[11][8,17] represents a closed interval, being a subset of the real continuum $\mathbb{R}$, representing an interval between clock times in 24 hour format.
[12] Such assertion may be performed by the IdP at the outset of a session, to be carried through tokens exchanged between a principal and the SP until expiry of the same session.
[13] Subject profiles are profiles related to one subject.
[14] $\mathcal{P}(X)$ denotes the power set [38] of the set X.

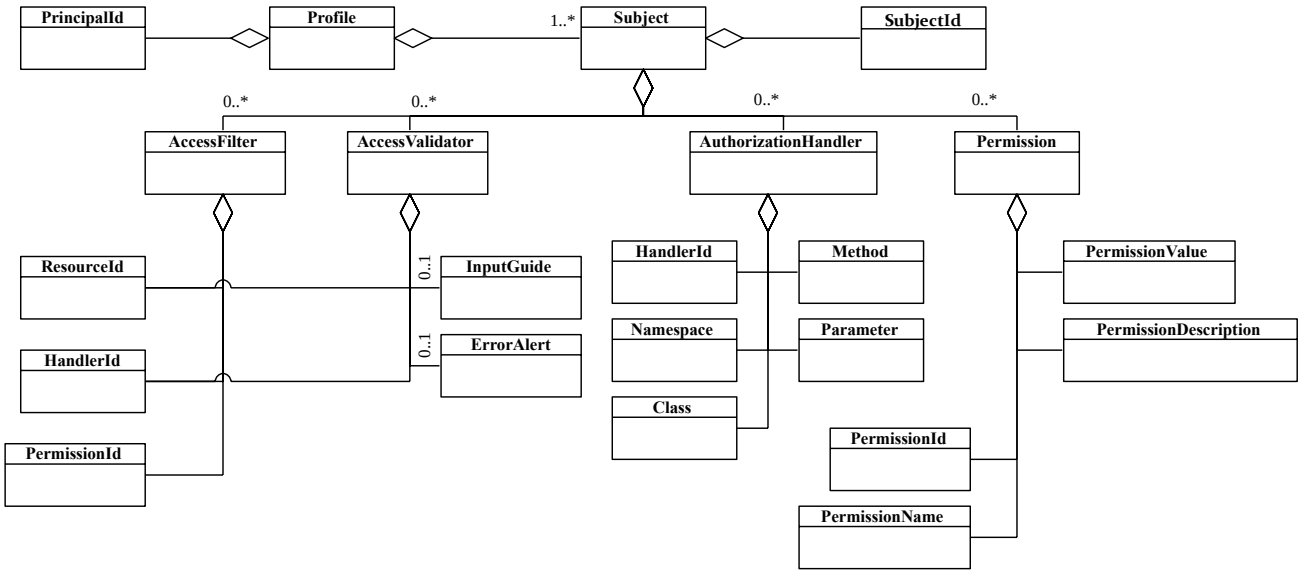

Figure 2: A sample schema of profile (i.e. of a principal)

2.2.3 Subject Id

Uniquely identifying a subject, subject id I_s'swould be derivable from principal id I_p, because associated subjects can be identified from the identity of a principal, vide supra (under Subject). Subject-profiles are aggregated over subjects related to a principal in order to constitute a comprehensive set of authorization-data related to the latter.

2.2.4 Access Filter

An access filter includes data related to a set of permissions, applicable to a subject, over a subset of UAC resources and how authorized views of such resources may be created. Thus, an access filter$j$is a unique vector $\left(I_j^v, I_j^h, I_j^\xi\right)$ ofa UAC resource id I_j^v, a handler id I_j^h, and a permission idI_j^ξ, describing which handler should be applied against which UAC resource and with which permission. Generically, the set of access filters comprise a sub-set of the set of three-tuples,$I^v \times I^h \times I^\xi$, of UAC resource id's, handler id's, and permission id's.

2.2.5 Access Validator

Access may need validation ex post facto, i.e. after access has been provided to a UAC resource; in the same way as access filters do before providing access. The vector $\left(I_j^v, I_j^h, M_j^t, M_j^e\right)$ representing an access validatorwould include UAC resource id I_j^v, handler id I_j^h, pre-access hint (hint-message) M_j^t, and exception alert (exception-message) M_j^e. One or more of the last two dimensions of the vector may be undefined.

2.2.6 Authorization Handlers

Handlers have been supererogatorily designed as reusable logic implemented with identical syntax but distinct semantics. Semantically differentiated but syntactically identical interfaces are not new[15]. With such interfaces one can effectively dissociate contexts from strategies in order to facilitate reuse by discovery of types at runtime, which may also be achieved, albeit in a limited way, with conditional logic. Let, for example, a handler $\hbar$ be known by the following generic (functional) signature:

(9) $$\hbar: \mathcal{I} \to \mathcal{O}$$

Here $\mathcal{I}$ represents input-type and $\mathcal{O}$ output-type. If $\mathcal{I}, \mathcal{O}$ are replaced by members of the context-type $\mathcal{C}$ then, given the function, discovery d, vide footnote 15, it is possible to bijectively ($\cong$) [16] map contexts into types, vide (10) below.

(10) $$d: \mathcal{C} \cong \mathcal{I} \cup \mathcal{O} \cup \ldots \Rightarrow \exists c_i, c_o \in \mathcal{C} \forall i \in \mathcal{I} \wedge o \in \mathcal{O} | \left(\left(d(c_i) = i \wedge d(c_o) = o \right) \wedge \left(i = j \Rightarrow c_i = c_o \right) \right)$$

The first part of (10) implies that the discovery function maps, equivalently[16], contexts into the union of different types, including input-type and output type. This implies that for all members of input-type or output-type one can always find corresponding context-type such that discovery of context yields types such as input-type or output-type. Further, distinct contexts always map to distinct types.

Therefore, it would be possible to discover syntactically correct types by contexts that are supplied at runtime, which would allow (9) being re-designed as (11) below.

(11) $$\hbar \circ d: \mathcal{C} \to \mathcal{O}$$

Thus, it is possible to map contexts into syntactically correct output-types. This is possible because d, vide (10), enables converting contexts into types[17]. Access filters can, thus, covert contexts into type-driven handlers (strategies) and apply them to control access over UAC resources.

2.2.6.1 Overcoming Platform Barriers

The scheme described above uses discovery in order to avoid serialization of types, too, some of which could be disparate across platforms. Thus, it may help if all types are serialized as universally recognizable context-type[18].

[15] Discovery of types at runtime in high-level languages such as through 'generics' or 'reflection' [39] are in vogue, whereby given context, it is possible to discover and construct a instance of a type. Discussion on a similar line can be found in the strategy pattern [14]. With non-object-oriented languages, e.g. UNIX and SQL, 'eval' and 'dynamic SQL' work in a similar fashion.

[16] Equivalency is a consequence of bijection [40].

[17] It may also be possible to change syntactic signature based on context. Thus $c_d \in \mathcal{C}$ may represent the context of a delimiter-type $, \in \mathcal{D}$ separating distinct types expressed as contexts. For example, if $c_a \in \mathcal{C} | d(c_a) = a \in \mathcal{A}$ and $c_b \in \mathcal{C} | d(c_b) = b \in \mathcal{B}$ hold then $\hbar\left(d(c_a)\right) = b$ and $\hbar\left(d(c_a c_d c_b)\right) = e \in \mathcal{E}$ may have same syntax. However, the delimiter-context c_d, in function $\hbar$ makes it bivariate by making the latter equivalent to $\hbar(c_a, c_b) = e \in \mathcal{E}$.

[18] XML [41] and JSON [42] can be two such types.

Since implementation of handlers would be specific to the platform of each access-controlled UAC resource, a generic function, e.g., (9) above, between any platform and UAC may return a profile as a memento that would contain all necessary contexts of types for controlling access. These are mapped to different types and are specific to platforms. Using taxonomy of major object-oriented languages, a typical approach[19]to invoke a method was chosen in the sample.

2.2.7 Permissions

Permissions Ξ combine with UAC resources Υ to give the superset of all views Ψ, vide (1) supra (under Platforms and UAC Resources)[20].

3. COMPONENTIZATION

How UAC stands in relation to other components is depicted in Figure 3 as well as described below. It may be noted that functions to be performed by platforms are included in UAC Resource Managers.

3.1 Interfaces

3.1.1 UAC Resource<n>SP (UAC Resource Service Provider)

The interface allows principals *P* to make use of the Principal Interface[21]I_p to request from UAC resources Υ and obtain amongst views Ψ determined by their profiles $\{\Pi_p \forall p \in P\}$. Earlier (vide Subject supra) it was discussed how id[22] of a principal, which ought to be part of a request, could be logically processed to obtain the set of views $\{\psi_p\}$ permissible to it. If all possible views corresponding to UAC resource$\upsilon \in \Upsilon$ be given by $\{\psi_\upsilon\}$ then $\Psi' = \{\psi_p\} \cap \{\psi_\upsilon\}$ should determine the view to be returned to the principal, if unique. However, if cardinality of Ψ' is greater than one then some other policy or condition e.g., 'union of permissible views if more than one' would determine the final view to be authorized. If composite views are unwarranted, principals may explicitly indicate which subject they represent to avoid exceptions [17].

3.1.2 Identity Service

Identity service is implemented by an IdP and used by platforms to identify principals at the outset of sessions. However, instead of the IdP, tokens or assertions (also, vide footnote 12), may be generated by platforms, which would facilitate per-transaction tokens, vide footnote 3, without overburdening the IdP.

3.1.3 Access Control Service

Platforms would make use of the interface to request access control data with respect to an identified principal. It may also include which UAC resource(s) the principal wanted access to and conditions, if any. In response, the service would return a memento containing authorization profile of the principal. If details such as UAC resources or conditions are supplied, the profile can be trimmed to include only relevant information.

3.1.4 Invoker

Platforms use the interface for decorating [14] UAC resources in order to create views, where handlers would provide strategies [14] and profiles contexts.

3.1.5 View Handling

UAC resources would implement the interface for the benefit of handlers in order to decorate them. If necessary, order of handlers may be specified in the profile.

3.1.6 View Generation

The interface, implemented by the platform, allows a UAC resource to pass a final and decorated view of it, thereby completing processing-cycle of authorization.

3.2 Components

3.2.1 Platform<n>

Platforms are containers of all other components. One platform differs from another in implementation details of its components—types in relation to components being generally incompatible across platforms. However, context-types need be undifferentiated to be understood in the same way universally.

3.2.2 UAC Resource<n>

UAC resources are components, views of which are accessed by principals. To facilitate the process, they would implement view handling and associate with view generation interfaces.

3.2.3 UAC Resource Manager<n>

These components coordinate the processes of authentication, authorization, and managing views. Thereby, they present a façade [14] to the principal interface. Internally, they may manage session-tokens, process profiles of principals to determine strategies in order to invoke appropriate handlers as strategies, and receive final views of UAC resources before passing to the principal.

3.2.4 Handler<n>

These components process specific authorization tasks, which may be generic or applicable to a specific type of UAC resources.

3.2.5 Principal Interface

These are components implemented by clients of the UAC resource service provider interface—they maintain an interface between principals and platforms.

3.2.6 IdP

This component implements the identity service.

3.2.7 UAC

This component implements access control service.

[19] These are as follows: handler id, namespace, class, method, parameters (separated by delimiter).

[20] In sample, permissions were described with a unique id, name, value, and description. The former would be used to index permissions, name and value would describe a property, description is optional.

[21] This may be a human-computer interface or one between two systems, depending on nature of the principal.

[22] Ids of principals may be taken as indirection of principals themselves.

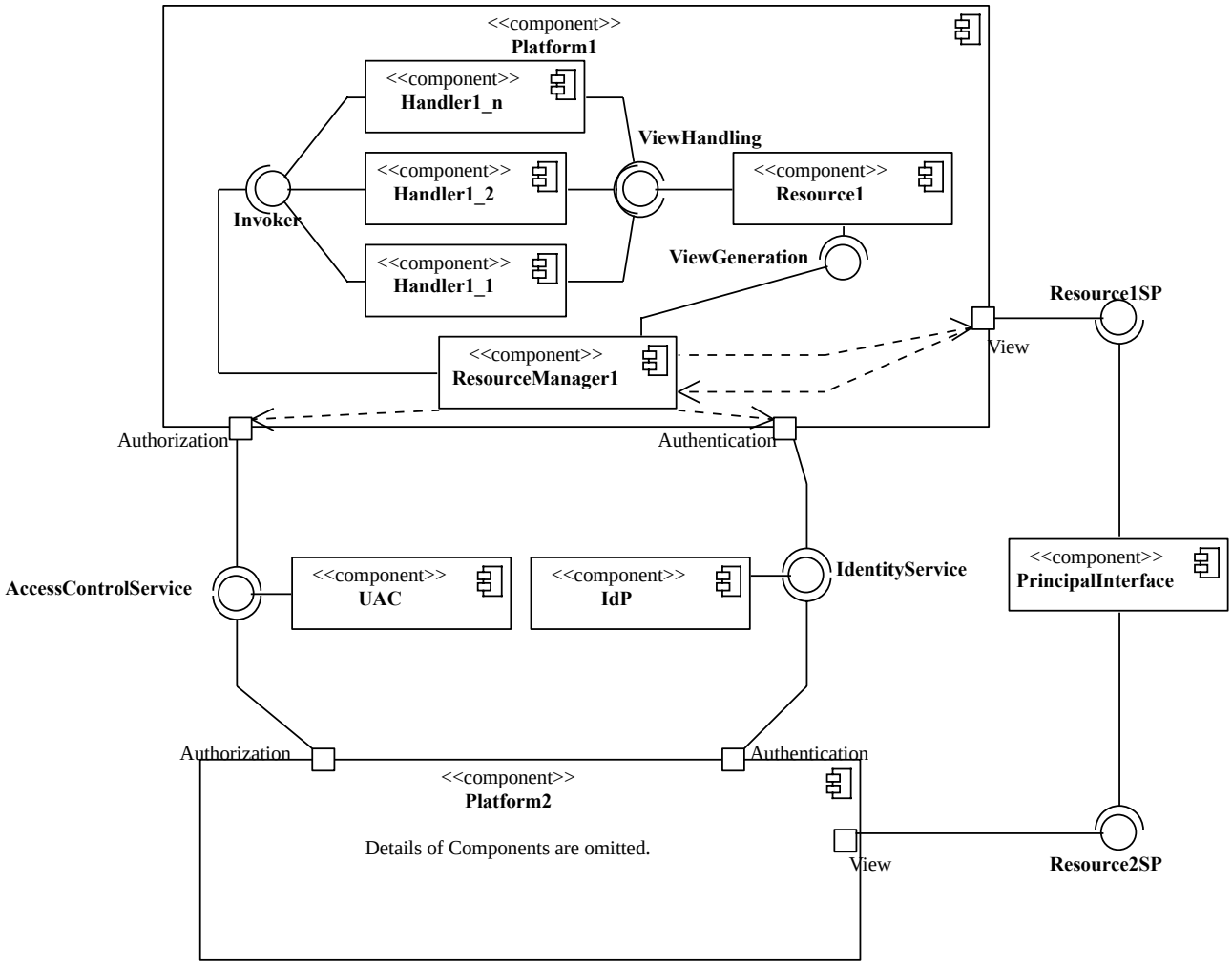

Figure 3: UML component diagram of CAuthS in relation to its ecosystem

4. ADMINISTRATION

Administration relates to the maintaining static relationship between UAC and other components. For such purpose, a maintenance interface would be provided by UAC (not in the diagrams).

5. ADAPTING TO AD HOC AUTHORIZATION SOLUTIONS

If ad hoc authorization solutions need be adapted, it may be achieved in on of the following two ways.

5.1 Adapting CAuthS to Ad Hoc Authorization

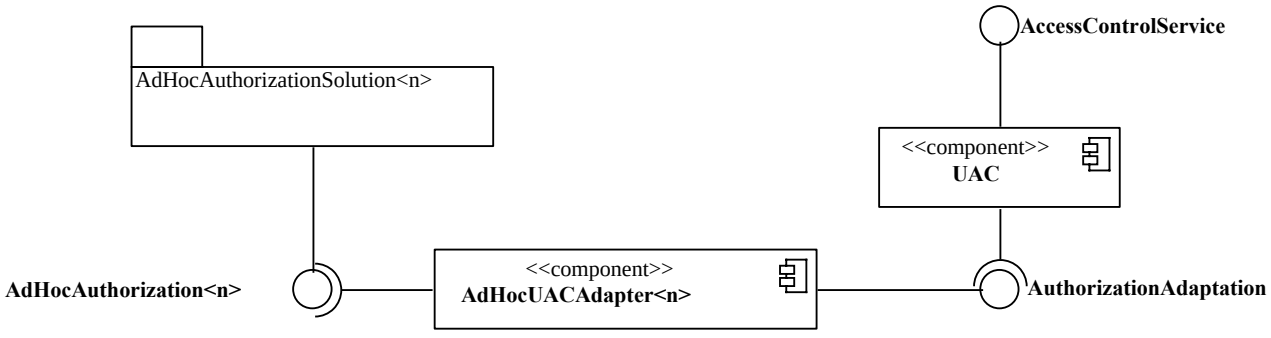

Figure 4: Adapting CAuthS to Ad Hoc Authorization

Firstly, adapter may convert calls to the access control service into those of the ad hoc authorization interface. Thus, there would be as many ad hoc UAC Adapters as there are ad hoc authorization interfaces. The scheme does not use a view-handling interface.

5.2 Adapting Ad Hoc Authorization to CAuthS

Alternatively, ad hoc authorization interface may allow unrestricted access to handlers over UAC resources vide

Figure 5. Unless view creation process is tightly coupled with the ad hoc authorization solution, this way more granular access control could be exercised by UAC.

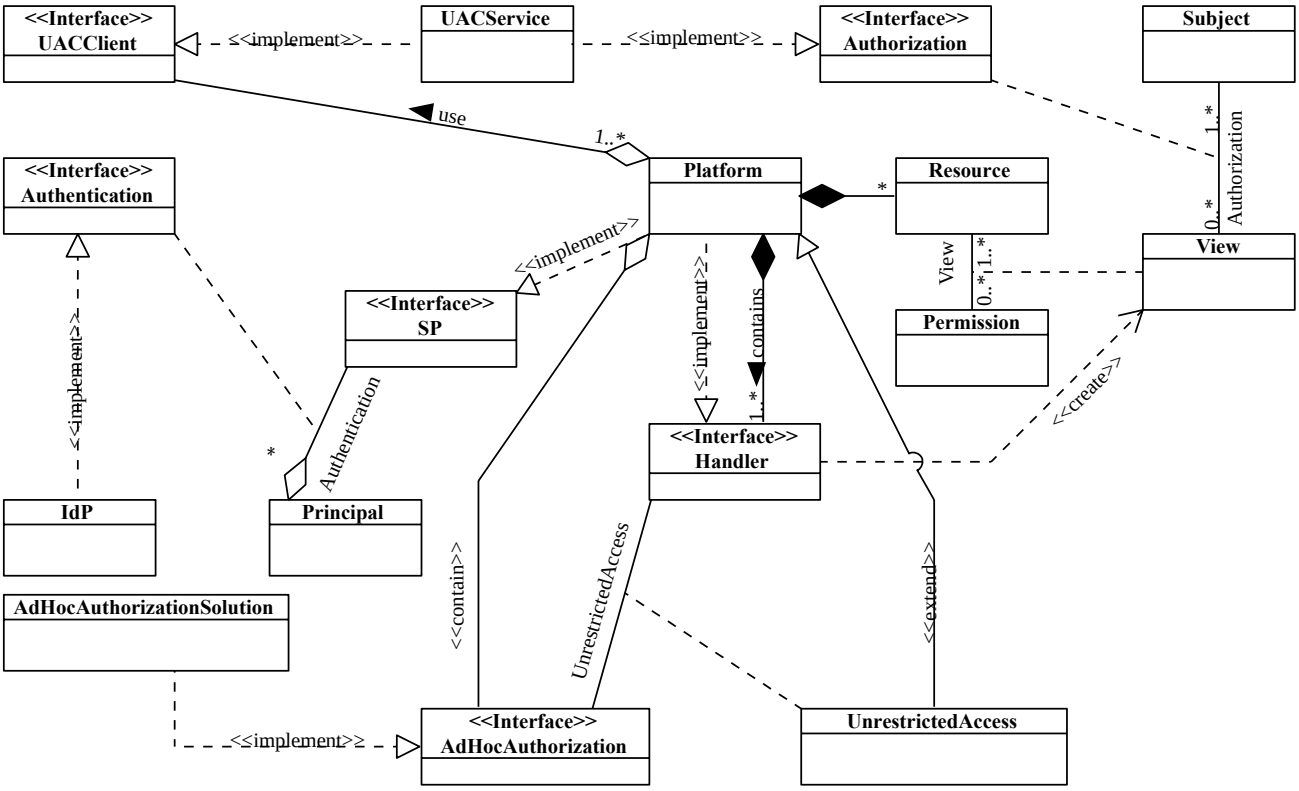

Figure 5: Adapting ad hoc authorization to CAuthS

6. CONCLUSION

The architecture facilitates centralizing authorization service in the following way.

(i) It separates the concern related to maintenance and management of authorization-related data from application of the same that, in a way, distinguishes between access-control "core" and "crosscutting"([18]; [19]).

(ii) The core concern of authorization has been separated from SP's by managing data related to them in platform-neutral, ubiquitous contexts, thereby facilitating centralization.

(iii) Handling of contexts forms a part of business logic of SP's that are core concern for them. However, handlers as strategies may introduce a standard way of handling, wherein contexts are provided by the UAC client.

(iv) Dynamic discovery of types, vide footnote 15,can scaled down maintenance of handlers.

(v) A profile, as a memento, may use a standard schema, e.g. one provided by XACML, or, an ad hoc schema. JSON may be used in addition to XML or an ASCII string. Thus, it is plausible that CAuthS uses different schemas for mementoes for different authorization clients.

(vi) It provides architectures to integrate ad hoc authorization solutions, thereby making coexistence possible among ad hoc solutions adopted by legacy applications and CAuthS.

Since a ubiquitous memento connects SP's with the UAC service, the latter may be deployed as a service, thereby justifying calling it as AuthaaS. Benefits of UAC as CAuthS or AuthaaS are following.

(i) Scaling productive computing resources, man or machine, would be easier. This is because, as a centralized service, one may load balance CAuthS independent of UAC resources that CAuthS authorizes.

(ii) Learning curve of maintaining authorization service would be flatter since one need not learn a multitude of solutions.

7. A Special Use Case[23]

A special use case of CAuthS would be to grant access to a subset $\Psi_{\subseteq p}$ of the accessible views Ψ_p by an authenticated principal $p \in P$ to another, possibly unauthenticated principal $p' \in P$, who normally would not have access to all views under such subset i.e., $\phi \subseteq \Psi_{p'} \cap \Psi_{\subset p} \subseteq \Psi_{\subseteq p}$. The process would be following.

(i) A permission grant $g \in \Xi$ may be created.

(ii) A subset Ψ_Y of views Ψ may be defined, which only would be under purview of 'grant'. Alternatively, let $\Psi_Y = \Psi$.

(iii) A subset P_Y of principals P may be defined, in a way similar to (ii) above. Alternatively, let $P_Y = P$. Another alternative would be P_Y may have

[23] The use case is similar to OAuth 2.0 standard ([43]; [44]). However, in the paper details of implementation are omitted.

principals all of which are not in P i.e., $\phi \subseteq P_\Upsilon \cap P \subseteq P_\Upsilon$[24].

(iv) A set Υ_g of composite resources may be defined, union of which with Υ may be assigned (:=) to Υ, vide(12) below.

(12) $\left(\Upsilon_g = (\Psi_\Upsilon \times P_\Upsilon) \wedge \Upsilon := \Upsilon \cup \Upsilon_g\right) \Rightarrow \Upsilon_g \left| \Upsilon_g \subseteq \Upsilon \cap (\Psi_\Upsilon \times P_\Upsilon)\right.$

(v) A subject $s_j^g \in S$ may be created that has grant g permission on a subset Υ_j of Υ_g. In other words, the subject has access to any view $\Psi_j = \Upsilon_j \times \{g\}$.

(vi) A principal $p \in P$ having been related to s_j^g by 'is authorized to' or A, i.e., pAs_j^g, would be able to grant permission on previously defined views to previously defined principals.

8. ACKNOWLEDGMENTS

The first author wishes to acknowledge encouragement and suggestions provided by Mr. Gangadhar Heralgi, Co-founder & CTO of Monocept and Mr. Ganesh Iyer, President of Vertex Computer Systems.

9. REFERENCES

[1] OASIS. (2013, Jan.) OASIS eXtensible Access Control Markup Language (XACML) TC. [Online]. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf

[2] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli, "Proposed NIST standard for role-based access control," ACM Transactions on Information and System Security (TISSEC), vol. 4, no. 3, pp. 224--274, 2001.

[3] Eric Yuan and Jin Tong, "Attributed based access control (ABAC) for Web services," in Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on, 2005.

[4] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in INFOCOM, 2010 Proceedings IEEE, 2010, pp. 1--9.

[5] Network Working Group. (2005, July) A Universally Unique IDentifier (UUID) URN Namespace. [Online]. http://www.ietf.org/rfc/rfc4122.txt

[6] Martin Gaedke, Johannes Meinecke, and Martin Nussbaumer, "A modeling approach to federated identity and access management," in Special interest tracks and posters of the 14th international conference on World Wide Web, 2005, pp. 1156--1157.

[7] Markus Lorch, Seth Proctor, Rebekah Lepro, Dennis Kafura, and Sumit Shah, "First Experiences Using XACML for Access Control in Distributed Systems," in XMLSEC '03 Proceedings of the 2003 ACM workshop on XML security, New York, NY, USA, 2003, pp. 25-37.

[8] Gregor Kiczales et al., "Aspect-Oriented Programming," in Proceedings of the European Conference on Object-Oriented Programming, vol. 1241, ECOOP, 1997, pp. 220–242.

[9] Ramon Lawrence, "The space efficiency of XML," Information and Software Technology, vol. 46, no. 11, pp. 753--759, 2004.

[10] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, and Clemente Izurieta, "Comparison of JSON and XML Data Interchange Formats: A Case Study.," Caine, pp. 157--162, 2009.

[11] Bruno Gil and Paulo Trezentos, "Impacts of data interchange formats on energy consumption and performance in smartphones," in Proceedings of the 2011 Workshop on Open Source and Design of Communication, 2011, pp. 1--6.

[12] Paul Moritz Cohn, Universal algebra.: Springer, 1981.

[13] Jaehong Park and Ravi Sandhu, "Towards usage control models: beyond traditional access control," in Proceedings of the seventh ACM symposium on Access control models and technologies, 2002, pp. 57--64.

[14] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Upper Saddle River, NJ: Addison-Wesley Professional computing Series, 1995.

[15] James F Gray, Sets,relations,and functions.: Holt,Rinehart & Winston, 1962.

[16] F. Borceux, Handbook of Categorical Algebra: Volume 2, Categories and Structures.: Cambridge University Press, 1994.

[17] Westley Weimer and George C Necula, "Exceptional situations and program reliability," ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 30, no. 2, p. 8, 2008.

[18] Kyo C Kang et al., "FORM: A feature-oriented reuse method with domain-specific reference architectures," Annals of Software Engineering, vol. 5, no. 1, pp. 143--168, 1998.

[19] Gregor Kiczales et al., "Getting started with AspectJ," Communications of the ACM, vol. 44, no. 10, pp. 59--65, 2001.

[20] Patricia P Griffiths and Bradford W Wade, "An authorization mechanism for a relational database system," ACM Transactions on Database Systems (TODS), vol. 1, no. 3, pp. 242--255, 1976.

[21] Sushil Jajodia, Pierangela Samarati, and VS Subrahmanian, "A logical language for expressing authorizations," in Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on, 1997, pp. 31--42.

[22] Michiharu Kudo and Satoshi Hada, "XML document security based on provisional authorization," in Proceedings of the 7th ACM conference on Computer and communications security, 2000, pp. 87--96.

[23] Jan De Clercq, "Single sign-on architectures," in Infrastructure Security.: Springer, 2002, pp. 40--58.

[24] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra, "Formal

[24] In the first two alternatives, IdP's as well as authentication interface corresponding to any member of P_Υ would be known to the platform; however, for the third alternative, such information—metadata relating to principals—may be added by the authorizing principal when they define a principal for the purpose of grant.

analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps," in Proceedings of the 6th ACM workshop on Formal methods in security engineering, 2008, pp. 1--10.

[25] Howard Barnum, Claude Cr'epeau, Daniel Gottesman, Adam Smith, and Alain Tapp, "Authentication of quantum messages," in Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on, 2002, pp. 449--458.

[26] Jingsha He, "System and method for single sign-on to a plurality of network elements," 5,944,824, Aug. 31, 1999.

[27] Timothy S Dare, Eric B Ek, and Gary L Luckenbaugh, "Method and system for authenticating users to multiple computer servers via a single sign-on," 5,684,950, Nov. 4, 1997.

[28] Andreas Pashalidis and Chris J Mitchell, "A taxonomy of single sign-on systems," in Information security and privacy, 2003, pp. 249--264.

[29] Dennis G Abraham and Richard K Hite, "Method and apparatus for initialization of cryptographic terminal," 5,745,576, Apr. 28, 1998.

[30] Ted Kremenek, Paul Twohey, Godmar Back, Andrew Ng, and Dawson Engler, "From uncertainty to belief: Inferring the specification within," in Proceedings of the 7th symposium on Operating systems design and implementation, 2006, pp. 161--176.

[31] Ed Keenan and Adam Steele, "Exploring game architecture best-practices with classic space invaders," in Proceedings of the 1st International Workshop on Games and Software Engineering, 2011, pp. 21--24.

[32] James Rumbaugh, Ivar Jacobson, and Grady Booch, Unified Modeling Language Reference Manual, The.: Pearson Higher Education, 2004.

[33] W Rudin, Principles of mathematical analysis, 3rd ed.: McGraw Hill, Inc., 1976.

[34] David F Ferraiolo and D Richard Kuhn, "Role-based access controls," arXiv preprint arXiv:0903.2171, 2009.

[35] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman, "Role-based access control models," Computer, vol. 29IEEE Computer Society, no. 2, pp. 38--47, 1996.

[36] Fawaz A and Miege, Alexandre and El Saddik, Abdulmotaleb Alsulaiman, "Threshold-based collaborative access control (T-CAC)," in Collaborative Technologies and Systems, 2007. CTS 2007. International Symposium on, 2007, pp. 46--56.

[37] Ruo-Fei Han, Hou-Xiang Wang, Qian Xiao, Xiao-Pei Jing, and Hui Li, "A united access control model for systems in collaborative commerce," Journal of Networks, vol. 4, no. 4, pp. 279--289, 2009.

[38] George A Gratzer, Universal algebra.: Springer Science & Business Media, 2008.

[39] Andrew Kennedy and Don Syme, "Design and implementation of generics for the. net common language runtime," in ACM SigPlan Notices, 2001, pp. 1--12.

[40] Steven Roman, Steven M Roman, and Steven M Roman, Advanced linear algebra.: Springer, 2005.

[41] Tim Berners-Lee, Dan Connolly, and Ralph R Swick, "Web architecture: Describing and exchanging data," WWW-address: http://www. w3. org/1999/04/WebData, 1999.

[42] Douglas Crockford. (2006, July) Network Working Group Request for Comments. [Online]. https://tools.ietf.org/html/rfc4627

[43] Whitfield Diffie, Paul C Van Oorschot, and Michael J Wiener, "Authentication and authenticated key exchanges," Designs, Codes and cryptography, vol. 2, no. 2, pp. 107--125, 1992.

[44] D. Ed. Hardt. (2010, Jan.) Internet Engineering Task Force. [Online]. https://tools.ietf.org/html/draft-hardt-oauth-01.