Jean Bacon and Ken Moody

Toward Open, Secure, Widely Distributed Services

The OASIS open architecture controls the interoperation of independent services in distributed environments, including the constant monitoring of security conditions, as illustrated by a U.K. application in health-record management.

The public is aware of computer networking because of the Internet-based Web and the multitude of services provided via Web browsers, irrespective of national boundaries. This awareness profoundly influences everyone's expectations, including those of officials who set policy on behalf of public institutions and businesses. They assume that the process of designing and deploying large-scale, network-based distributed systems is straightforward. But many do not realize that earlier complex systems like the worldwide telephone network had time to evolve, growing by increments and incorporating new technology as it became available. Today, however, the rapid and continuing evolution of computer technology, and its ubiquitous, networked deployment, has created a situation beyond the experience of even well-versed system designers.

The Opera research group at the University of Cambridge Computer Laboratory works on the design and deployment of large-scale, widely distributed systems, modeling them as domains of services. An example is a national Electronic Health Record (EHR) service for the U.K., with a central EHR management domain and thousands of client domains (including hospitals, clinics, research institutes, and primary care practices) serving millions of patients throughout the U.K.

TERRY MIURA

Computer science research has made good progress toward the design of complex, large-scale, distributed systems. Over the past 20 years, we've learned how to design naming schemes for components and participants and how to perform name-to-location mapping. We know how to replicate services to achieve 24-hour availability, and we understand the trade-offs between strong and weak consistency of replicated data. We can select a suitable, secure communications infrastructure. What remains is to incorporate these results into the design of total systems and provide tools for managers to configure and adapt them.

Heterogeneity must also be possible to allow independently developed and legacy software to work together. We support diversity with service-level agreements (SLAs) between these components. Incremental deployment of new components is also essential for large-scale systems; no such system begins atomically with a big bang.

Our research focuses on two remaining areas of distributed systems development: timely interaction and access control. The latter includes enabling nonexpert administrators to use tools to express and modify access-control policies, so such policies can be automatically checked for consistency and enforced. Here, we outline our work on event-based middleware, then describe OASIS (Open Architecture for Secure, Interworking Services), along with related work on access-control policy expression and enforcement.

Event-based Middleware

Asynchronous, event-based middleware is needed for many existing and emerging large-scale distributed applications, including the management of mobile objects in sensor-rich environments (an aspect of pervasive computing), to provide timely response to fault and alarm conditions and manage multimedia and group communication. Since the early 1990s we have sought to extend standard middleware platforms with asynchronous communication in the form of events [1].

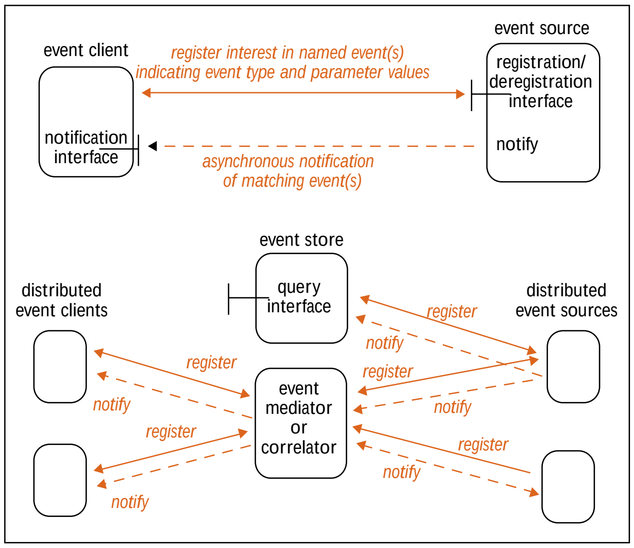

Figure 1. The Cambridge Event Architecture.

Figure 1 shows the components of the Cambridge Event Architecture (CEA). Objects that are event sources publish the event types they are prepared to notify. Clients invoke the register method at an event source indicating the event name of interest and the required parameter values or wild cards. Subsequently, when events occur, the source matches each event occurrence against client registrations and notifies interested parties. Note that because filtering is at the source side only events of interest are notified.

Along with direct communication between event sources and sinks, the architecture includes event mediators, event correlators (also known as composite event services), event gateways, and event stores. An event mediator is useful for presenting the information of a primitive event source, such as a simple sensor, through a higher-level interface than the source alone could offer. For example, a mediator may support an event *seen(person, room),* whereas a sensor may work in terms of lower-level sensor and badge identifiers.

A mediator is concerned with basic events, whereas a correlator is concerned with the live detection of patterns of events that might indicate some alarm condition or suspicious behavior. Composite events are created from streams of basic or composite events, combined using event operators. Event gateways are concerned with federating event systems. Event stores are necessary for logging and audit. An event store component may register interest in selected events using the standard CEA mechanism and receive streams of notifications. A query interface allows interrogation to find occurrences and patterns of interest; a promising commercial application is fraud detection in financial systems.

CEA is middleware-platform-independent; our work shows how any widely used platform, including the Object Management Group's Common Object Request Broker Architecture (CORBA), Sun Microsystems' Java Remote Method Invocation (RMI), and Microsoft's Distributed Component Object Model (DCOM), can be extended to support asynchronous operation.

Our first implementation of CEA in 1997 used CORBA's standard type system, namely the CORBA Interface Definition Language (IDL). The advantage was that commonly used programming languages have bindings to CORBA IDL, so heterogeneous operation is supported. In order to handle transmitted and persistent (stored) events consistently we used the Object Data Management Group's (ODMG) Object Definition Language (ODL), which is a superset of IDL and supports the Object Query Language (OQL) for querying event stores.

If the OMG/ODMG standardization approach had gained wide acceptance, applications would have been able to work with an efficient implementation of a well-defined, rich, type system supporting heterogeneous interoperation. Instead, the advent and enormous worldwide popularity of the Web transformed the nature of access to distributed information. Object-oriented middleware evolved from a programming language focus on distributed-system design via early remote procedure call systems like OSF's DCE, ISO's RM-ODP, and Xerox's Courier, as well as those from a number of research teams, including Amoeba and Mayflower.

The alternative approach was to build distributed systems bottom-up from the communication of packets of bytes, an effort that led to message-oriented middleware, or MOM. MOM, such as that from IBM (MQseries) and Tibco Software (Rendezvous) was developed in parallel with object-oriented middleware but employed a less well-defined type system and an asynchronous communication model. More recently, the widespread use of the Web-services model has led to the remarkable acceptance of a middleware technology derived from document definition.

The Extensible Markup Language (XML) is a subset of the Standard Generalized Markup Language (SGML) document standard of 1985. The Simple Object Access Protocol (SOAP) defines object invocation using XML for transferring typed data (see the World Wide Web Consortium; www.w3.org). MOM today is being redesigned to use XML to define the structure of messages, including types of arguments and presentation format.

More recently, we have produced an XML-based implementation of CEA, generalized for greater scalability and wide-area operation. Like ODL, XML may be generated automatically from programming language types for transmission in messages, though its use in message communication was not the intention of its original designers. Not surprisingly, XML's type system is weaker than that of programming languages, and transmitting and storing string-based data is inefficient and at present gives poor support for querying. Despite these shortcomings, XML's wide acceptance makes it a reasonable choice for large-scale interoperation.

OASIS Access Control

OASIS is a role-based access control (RBAC) architecture used to deliver secure interoperation of independently managed services in open, distributed environments [3, 7]. Associating privileges with roles, RBAC provides a means of expressing access control requirements that is scalable to large numbers of principals and objects within a system. A notable advantage is the avoidance of the detailed management of large numbers of access control lists when people change their employment or function. Decentralized, domain-level RBAC is an effective approach for large-scale, widely distributed systems. For example, it is not necessary for the U.K. National Health Service (NHS) to register all doctors nationally; a hospital domain may define a role *doctor* for its employees, with corresponding privileges. The NHS can then recognize the local role under an appropriate SLA.

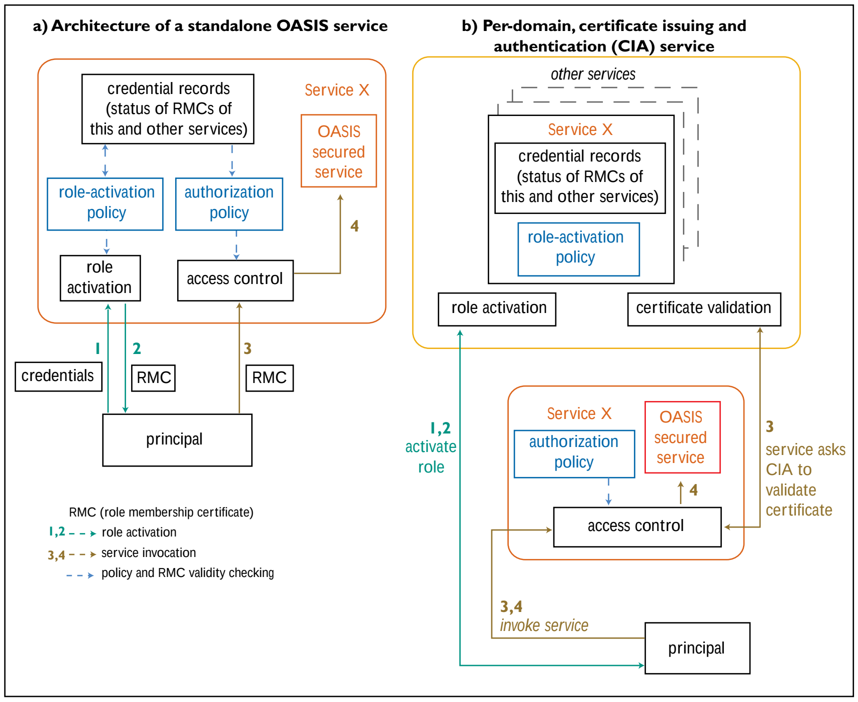

Figure 2. OASIS service architecture and per-domain engineering.

RBAC is also likely to correspond well with policy expressed in legislation. Such policy may also require the expression of relationships, such as the registration of a patient with a particular doctor. It may also allow individual exclusions; for example, my uncle may be a doctor allowed—by generic policy—access to my EHR, through under the Patient's Charter I can forbid that access. Pure RBAC associates privileges only with roles, whereas OASIS roles are parameterized so these additional requirements can be captured and checked.

OASIS services name their client roles and enforce policy for role activation and service invocation (see Figure 2a). The encryption-protected role membership certificate (RMC) returned to the user on successful role activation may be used as proof of authorization to use this and other services and as a credential for activating other roles, depending on the policy of each service.

Unlike many RBAC schemes, OASIS does not support the delegation of privileges. Instead, it uses "appointment," whereby a function of certain roles is to issue appointment certificates that may be used (with any other credentials required by policy) to activate one or more roles. There is no reason why the holder of the appointer role should be entitled to the privileges conferred by the certificates. For example, although hospital administrators need not be qualified to practice medicine they can issue appointment certificates in the form *doctor(hospital-id)* to indicate a doctor is employed at a particular hospital.

Appointment meets the requirement for long-lived credentials; in contrast, privileges are associated with roles, which are activated in the context of an authenticated session. The fact that sessions are of short duration promotes an active security environment in which role-membership conditions are monitored and breaches notified. OASIS is closely integrated with our active event-based middleware infrastructure (CEA), making it possible to ensure that security policy is satisfied at all times.

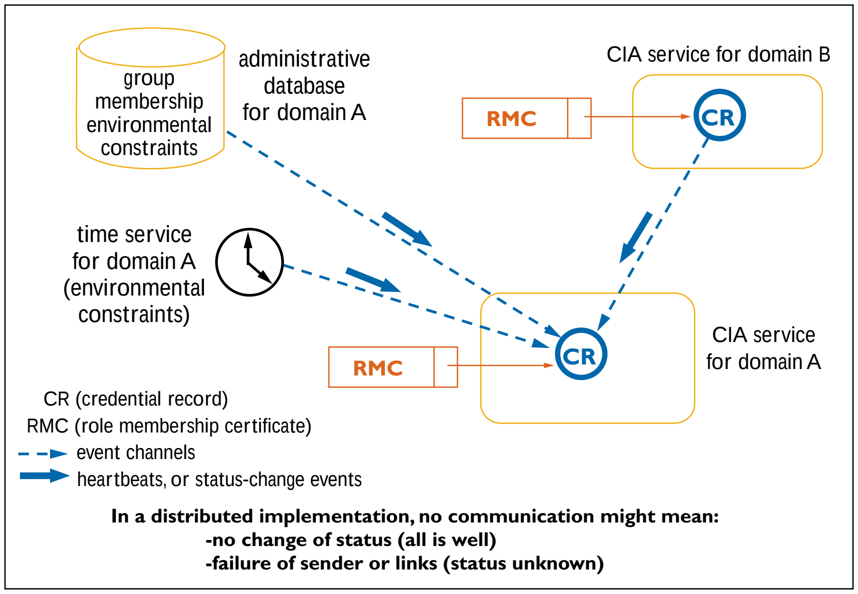
group membership environmental constraints
administrative database for domain A
CIA service for domain B
RMC
CR
time service for domain A (environmental constraints)
RMC
CR
CIA service for domain A
CR (credential record)
RMC (role membership certificate)
event channels
heartbeats, or status-change events
In a distributed implementation, no communication might mean:
-no change of status (all is well)
-failure of sender or links (status unknown)

Figure 3. Maintaining an active security environment.

Within an OASIS session, it is often necessary to make cross-domain invocations of remote services. For example, if a doctor, away from his or her usual practice, has to perform an operation, he or she will need access to the patient's health records. This access is easily authorized through cross-domain SLAs—the agreement of the patient's local domain to recognize an appointment certificate issued by the doctor's base domain as a credential for activating a role, such as *visiting-doctor(home-id, away-id)*. The result is that although OASIS is role-based it involves important differences from other RBAC schemes, including the seminal work of [9]:

- Roles are service-specific; there is no need for globally centralized administration of role naming and privilege management.
- Roles may be parameterized, as required by applications.
- There is no explicit role hierarchy, hence no inheritance of privileges.
- Roles are activated by presenting the credentials specified by policy.
- Roles are activated within sessions, providing good security.
- As in traditional RBAC, all privileges are associated with roles. OASIS uses appointment instead of privilege delegation; activation conditions of roles, conveying whatever privileges are granted, may require appointment certificates, which can be persistent.
- OASIS provides an active security environment facilitated by session-limited privilege allocation. Conditions checked during role activation can include constraints on the context; the role is deactivated if any membership condition subsequently becomes false.

Figure 2a outlines the architecture of an OASIS service that defines roles. A client activates a role by presenting "credentials" to a service, thus enabling it to prove the client conforms to its policy for entry to that role (path 1). The service validates the credentials, possibly checking back with certificate-issuing services and establishing environmental constraints. If all checks succeed, an RMC is issued (path 2) and the service creates a credential record corresponding to it.

The RMC may be presented with subsequent requests to use that service (path 3). The service validates the RMC, checks any environmental constraints required by authorization policy, and, if all is well, lets the invocation proceed (path 4). Activation of any role in OASIS is controlled explicitly by a role-activation rule specifying, in Horn clause logic (a conjunctive formal notation), the conditions users must meet to activate the role. Conditions may include prerequisite roles, appointment certificates, and environmental constraints. For more on the OASIS model, including its formal semantics, see [5].

OASIS engineering. It is unlikely that certificates would be issued and validated by each individual service; for example, a hospital might offer, say, Accident, Emergency, Pharmacy, and X-ray services. Instead, a hospital domain contains one highly available service—the Certificate Issuing and Authentication (CIA) service [8]—as in Figure 2b. Role activation is carried out by the CIA service on behalf of all services in a domain; it also maintains a credential record for each issued RMC. Domainwide record administration allows efficient implementation of the management of role dependencies within a domain; knowledge of the status of any prerequisite roles within domain A is maintained within the CIA service and does not require an external event channel.

The active security environment outlined in Figure 3 involves monitoring the membership rule for an RMC within domain A. Its validity depends on a prerequisite role in domain B and two environmental constraints in domain A. When any service is invoked, the RMC supplied by the invoker is validated by the CIA service during authorization. An optimization involves caching the RMC; an event channel from the CIA service informs the service if the RMC subsequently becomes invalid. The next time the RMC is used it can be compared with the cached bit-pattern.

Despite its shortcomings, XML's wide acceptance makes it a reasonable choice for large-scale interoperation.

Integration with a public key infrastructure. Role membership certificates may not have a systemwide format, though there is likely to be a uniform format within each domain. The role-name and any parameters are recorded in the RMC. The owner's personal identification may or may not be one of the parameters, depending on application requirements. RMCs are encryption-protected to guard against tampering and principal-specific to guard against theft.

Because the design of X.509 certificates meets our requirements, we use them in the current implementation of OASIS. We have extended their normal use, for the purpose of authentication, using the extension fields for access control information, that is, for role names and parameters. A CIA server is therefore responsible for creating, signing, issuing, and validating these certificates. The presenter of the certificate may be authenticated (through a challenge-response protocol) as the owner of the private key corresponding to the public key in the certificate; for more on security threats to RMCs and the way they may be countered, see [4].

Policy Expression and Management

In practice, distributed systems contain many domains; for example, the health-care domain comprises the subdomains of public and private hospitals, primary care practices, research institutes, and clinics, as well as national services, including electronic health-record management. Access control policy may be dictated by national law and/or derive from organizational decisions. An early attempt at pseudo-natural language policy expression and its automatic translation into first-order logic was described in [2]. A common semantic representation is crucial for any large-scale deployment of policy; consistency must be maintained as policies evolve.

Since each individual health-care domain expresses its own access-control policy, coordinating policy across domains may not be straightforward. We have addressed this problem through meta-policies [6] expressing predefined constraints to which changes of policy are subject. Because meta-policies are long-lived, they provide to bodies external to a domain stable information about the domain's policy, thus forming a basis for interworking.

We use an object-relational database (PostgreSQL) to record per-domain access-control information (such as the structure of RMCs) and to store role-activation and authorization policies. We have used the PostgreSQL trigger facilities to build an active predicate store against which clients register their interest through templates. It is thus possible to monitor changes in a domain's policy to ensure conformance with meta-policy and so maintain agreements for interoperation between services automatically. We have closely integrated the CIA service for a domain with these PostgreSQL databases using event channels. The extensions to PostgreSQL are transparent to its use by an application as a conventional database management system.

Conclusion

Our early work on event-based systems has achieved general acceptance, and middleware platforms have come to include event services, albeit with a weaker type system than we would wish. Our use of such an active middleware platform to provide constant monitoring of security conditions is unique, to our knowledge.

Many authors have argued that RBAC is a realistic way to provide scalable policy expression for access control in large-scale systems, yet most work on RBAC has used a single organizationwide role-management model and privilege delegation. Our work began with the assumption of widely distributed systems with many independently managed domains needing to interoperate. The result is OASIS, which includes an architecture and a formal model; engineering in the context of several practical applications is under way, most notable a pilot project for an EHR service for the U.K. NHS. In the future, we shall collaborate with natural language processing experts to automate the expression of policy, translating it into a formal notation, so it can be enforced as implemented code. **C**

References

1. Bacon, J., Bates, J., Hayton, R., and Moody, K. Using events to build distributed applications. In *Proceedings of the 2nd Workshop on Services in Distributed and Networked Environments (SDNE'95)* (Whistler, BC, Canada, June 3–5). IEEE Computer Society Press, Los Alamitos, CA, 1995, 148–155.
2. Bacon, J., Lloyd, M., and Moody, K. Translating role-based access control policy within context. In *Proceedings of the 2nd International Workshop on Policies for Distributed Systems and Networks (Policy 2001), Lecture Notes in Computer Science, vol. 1995* (Bristol, U.K., Jan. 29–31). Springer-Verlag, Heidelberg and New York, 2001, 107–119.
3. Bacon, J., Moody, K., Bates, J., Hayton, R., Ma, C., McNeil, A., Seidel, O., and Spiteri, M. Generic support for distributed applications. *IEEE Comput. 33,* 3 (Mar. 2000), 68–76.
4. Bacon, J., Moody, K., and Yao, W. Access control and trust in the use of widely distributed services. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Lecture Notes in Computer Science, vol. 2218* (Heidelberg, Germany, Nov. 12–16). Springer-Verlag, Heidelberg and New York, 2001, 300–315.
5. Bacon, J., Moody, K., and Yao, W. A model of OASIS role-based access control and its support for active security. *ACM Transact. Info. Syst. Sec. 5,* 4 (2002).
6. Belokosztolszki, A. and Moody, K. Meta-policies for distributed role-based access control systems. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (Policy 2002)* (Monterey, CA, June 5–7). IEEE Computer Society Press, Los Alamitos, CA, 2002, 106–115.
7. Hayton, R., Bacon, J., and Moody, K. OASIS: Access control in an open, distributed environment. In *Proceedings of 1998 IEEE Symposium on Security and Privacy* (Oakland, CA, May 3–6). IEEE Computer Society Press, Los Alamitos, CA, 1998, 3–14.
8. Hine, J., Yao, W., Bacon, J., and Moody, K. An architecture for distributed OASIS services. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2000), Lecture Notes in Computer Science, vol. 1795.* (Palisades, NY, Apr. 4–8). Springer-Verlag, Heidelberg and New York, 2000, 104–120.
9. Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. Role-based access-control models. *IEEE Comput. 29,* 2 (Feb. 1996), 38–47.

Jean Bacon (Jean.Bacon@cl.cam.ac.uk) is a reader in distributed systems in the Computer Laboratory of the University of Cambridge, Cambridge, U.K.
Ken Moody (Ken.Moody@cl.cam.ac.uk) is a lecturer in the Computer Laboratory of the University of Cambridge, Cambridge, U.K.

The U.K. Engineering and Physical Sciences Research Council supports this work through grants and studentships.