Controlling Historical Information Dissemination in Publish/Subscribe

Jatinder Singh David M. Eyers Jean Bacon
Computer Laboratory
University of Cambridge
{first.last}@cl.cam.ac.uk

ABSTRACT

Application environments dealing with sensitive information require mechanisms to define the circumstances for data disclosure. In event-based environments, access control typically concerns messages (events) as they occur. However, scenarios exist in which the retrieval of historical information is required. The publish/subscribe paradigm decouples producers from consumers, where information from numerous sources can satisfy an information request (subscription). These sources may be unknown to subscribers.

This paper describes a unified approach for managing the disclosure of both historical and future events. We show, with the aid of healthcare scenarios, how context and access mechanisms can be used for fine-grained control over the circumstances for information disclosure.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Security, Management

Keywords

Data Control, Middleware, Event Replay, Publish/Subscribe

1. INTRODUCTION

Many application environments require control over data access. In some application domains, such as healthcare, data is particularly sensitive; stringent control mechanisms are required to meet the strict access requirements.

Access control decisions are circumstantial, referencing context to determine whether to allow access to information. Context may include details of the requester (credentials), current environmental state (an emergency situation, step in a workflow procedure), aspects/details of the request or the information itself—or some combination thereof. This decision may be binary: permit or deny; or may customise the information to the particular situation (see §2.2).

Publish/subscribe is a well-established asynchronous event dissemination paradigm. It is suitable for large-scale information distribution in data-driven application scenarios. Publish/subscribe delivers information to those interested as it occurs. However, certain situations require the delivery of *historical* information—events that have previously occurred. This is where an incident makes previous information relevant; perhaps due to mobility (events missed during disconnection) or where a change in context requires past information. In distributed environments, there may be multiple sources supplying a particular event, making it impractical for a principal to directly query (in a point-to-point manner) each datastore for the historical information [8].

There are various methods for controlling data access in a publish/subscribe paradigm [1], such as imposing restrictions on subscription requests and/or notifications (information delivery). Encryption can be used to secure event attributes and types. Event transformation allows event information to be tailored to the particular circumstances.

Event replay mechanisms require control, allowing policy to define the circumstances in which historical information is released. An important consideration is the effect of context on event replay. As access mechanisms are context-dependent, a replayed event might be subject to different controls from the original, due to a change in circumstance. For example, an event originally subject to a restriction filter, may now, instead, be subject to a transformation. Further, as all events use the same infrastructure, it is possible that the replayed events may impact on system state. It might be the case that an event (e.g. an emergency) must be delivered to a certain number of recipients to meet a particular obligation. Replayed events are relevant to the monitoring of this condition. The historical request itself may also be of significance, e.g. to audit the auditor.

This paper describes mechanisms for authorising and controlling historical event republication. We use real-world healthcare examples to describe how policy administrators can exploit context-aware access mechanisms to control data disclosure. Our approach unifies delivery for both live and historical events, providing a common interface for the administration of disclosure policy, while facilitating similar event-processing actions for both types of events.

2. BACKGROUND

2.1 The Publish/Subscribe Paradigm

Publish/subscribe [6] is an asynchronous message delivery paradigm suited to data-driven environments. A principal may be a publisher and/or a subscriber. A publisher

produces an *event*—a data-rich encapsulation of a particular semantic. Subscribers register their interest in receiving particular events (information) through a subscription. Communication between principals occurs through the publish/subscribe middleware, which delivers events from publishers to subscribers with matching subscriptions. Principals connect to a *broker*, which through its interconnection with other brokers, provides the middleware functionality.

A key feature of the paradigm is that information producers and consumers are decoupled, i.e. those producing information are not burdened with the addressing specifics of the potential end-points. Instead, the middleware is responsible for information delivery, routing events according to interest *as they occur*. As all communication passes through the middleware, it is an appropriate point for the enforcement of information control policy. Another characteristic of publish/subscribe is that its ability to perform distributed content-based routing can assist in load-balancing.

2.2 Access-control in Publish/Subscribe

There are various ways to restrict information flow in a publish/subscribe framework [1]. Basic control concerns access to the middleware, describing who may connect, subscribe to an event type or publish an event instance. Filters may qualify access, restricting on various properties—typically event content. Encryption and key-management can be used to control the visibility of event types and the attributes of event instances [12]. Some control mechanisms allow publishers to define the circumstances for event delivery, either through filters [16] or by authorising specific subscribers [11]. Wun et al. [20] describe a generic policy model for publish/subscribe frameworks; directions are towards middleware enforcement of application-level policy.

Interaction Control

Interaction control provides the means for customising data access to circumstance [1]. Information is controlled through policy rule definitions that are loaded and enforced in a broker to control the data it releases. These rules set the bounds for data transmission.

Our model uses three types of data control rules [1, 15]. *Subscription authorisation rules* define the circumstances in which a subscription channel is allowed. *Restrictions* are silently imposed on a subscription to filter the messages delivered to a particular subscriber. Finally, *transformation rules* allow an event instance to be transformed into another. This might serve to enrich or perturb the information of the event instance, or perhaps convert an event into another type to encapsulate a different semantic. Transformations provide more than binary access control (permit/deny) by allowing data to be tailored to circumstance.

The power of interaction control stems from the fact that rules are context-sensitive. Rule definitions can access messaging system information (e.g. broker/principal information and message content), credentials (i.e. the identity, qualifications, roles and certificates held by the principals) as well as environmental state (e.g. emergency situations), stored data and external services. As policy rules are context-aware, unlike typical access control mechanisms, restrictions can be unrelated to the credentials of the principals, or event content, e.g. "nobody may access my heart-rate data unless I'm in cardiac arrest". They are also reactive to context; for instance, a subscription authorisation rule enforcing a relationship between a doctor and a patient can cause the subscription to be deactivated if the doctor no longer treats that patient. Transformations, which alter information, respond to context by applying only in certain situations.

These interaction control mechanisms were developed for application environments where information crosses boundaries of administrative control. Each administrative body maintains its own—possibly differing—policies that define the situations for data release, which are loaded into the broker(s) they control. This gives the ability to manage and control data by setting the bounds for release. Subscribers may define their interests, though these are subject to interaction control mechanisms.

Fine-grained access control mechanisms depend on context, thus a change in circumstance can alter information flow. Previous work in controlling publish/subscribe focuses on events as they occur within the system. When handling historical events, it becomes necessary to consider (1) the circumstances in which replay is appropriate, and (2) how a contextual change alters information flow.

2.3 DB Publish/Subscribe Infrastructure

We build on PostgreSQL-PS, an integrated content-based publish/subscribe and database system [19]. This means a database instance, in addition to storing information, operates as a broker routing information from publishers to subscribers. It also has the ability to, itself, publish and subscribe to (process) particular events. This coupling brings a common type system and API to both the messaging and data storage substrates. Importantly, this means that a broker has a rich representation of context, with the ability to access stored data, or internal or external functions through database languages. Events, subscription requests, publications and policies are represented in XML.

We have built interaction control mechanisms into this infrastructure [15] to control the flow of events as they occur within the system. Control concerns data transmission, thus a database-messaging system is the obvious point to enforce access policy. As a broker is a database, with rich access to state and the capability to process (subscribe, publish and analyse) events, it means that events themselves serve to alter context, which in turn effects access control mechanisms[1]. Here we extend this control to manage historical event dissemination, where context-aware data replay rules can be defined to appropriately protect historical data.

3. HEALTHCARE BACKGROUND

Healthcare is the motivating scenario for our research into publish/subscribe information control. Care providers must share information to afford proper care. At the same time, health data is sensitive, and must be protected. Those collecting and holding information as part of the care process are responsible for its protection, with serious repercussions for mismanagement or misuse [17]. To balance these concerns, health data must be shared when appropriate, in line with patient consent, which may be explicit, but is typically implied in the interests of the patient's health [2].

Homecare services (including mobile and remote care) are acknowledged to be the future of healthcare. Such environments benefit both patients and the health service in terms

[1]In addition to other context-reactive processes, such as triggers and workflows.

of improved (preventative) care and resource allocation [4]. Homecare environments are created to cater for a particular aspect of a patient's care [13]. Existing outside of a central environment, they are particularly amenable to collaboration across administrative domains, where different entities provide particular services as part of the care process. Sensors and monitoring technologies also form an integral part of the homecare interaction-mix. Health incidents, be they actions (of a patient or carer), observations (e.g. sensor readings) or state transitions (e.g. cardiac arrest) are relevant to various care providers. However, the information that an entity requires from an incident will depend on the service they provide, in addition to current circumstances.

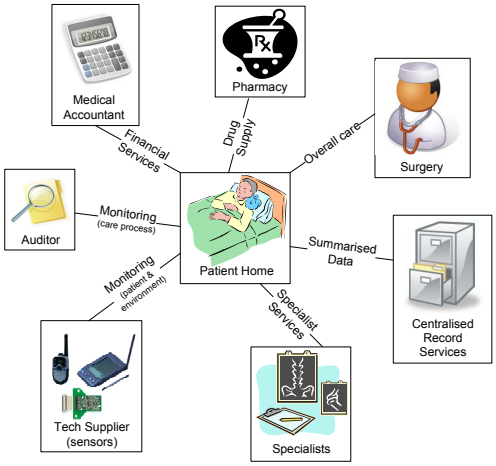

Figure 1: Home healthcare involves interactions between entities, managed in different administrative domains. Each has different data requirements, depending on the health services they provide.

The healthcare space is heterogeneous, where each administrative domain maintains a degree of control over their systems, processes and services [9]. As the environment is highly data-driven, event-based infrastructure is appropriate for managing interactions between entities, which may be across administrative boundaries. Our work in securing publish/subscribe systems is in line with this, providing administrative domains (care providers) with fine-grained control over the circumstances in which data is released from their local brokers. This allows providers to meet their data management responsibilities. Previous work concerned managing events as they occurred. Here we present a framework to unify the control of historical and current events, considering context and access control specifics.

3.1 Historical Event Replay: Drug Allocation

This scenario concerns the supply of controlled drugs. Nurses are authorised to prescribe drugs, which includes classes of controlled drugs (e.g. morphine), in certain situations [5]. Legislation states that the use of controlled drugs must be monitored, though the audit is prescriber focused [18]. Patient specifics should not be shared unless necessary, e.g. when a prescriber is under investigation.

We have used this example to demonstrate the power of our data control mechanisms in restricting information flow [15, 14], where the auditor receives prescription information for controlled drugs with patient specifics removed. If the prescriber is under investigation, the restrictions are removed so that the auditor receives patient specifics to assist in the investigation. Historical information is useful for an investigation as it provides details of the patients who received controlled drugs from this prescriber.

Consider a homecare nurse employed by a nursing clinic. Her employer contracts with many surgeries that manage home environments to provide daily homecare services. Currently, the nurse treats 15 patients a week, from 4 different surgeries; though throughout her employment contract, she has treated patients from over 50 institutions. It comes to light that a number of her patients have been falling ill, and that this nurse has been prescribing controlled drugs far more frequently than other nurses. As such, she is officially placed under investigation by the NHS. The auditor requires detailed information regarding the circumstances, including patient specifics, regarding past and future prescriptions authorised by the nurse. Although under investigation, she is not accused of anything, thus she continues to practice. However, it is in the interest of the public and patient safety that the auditor investigate the matter[2].

3.2 Sensor Obfuscation

Homecare environments are increasingly using monitoring technologies [4]. Sensors exist that measure various aspects of physiological state. These are often coupled with location sensors, to monitor position and mobility. As sensor readings may contain sensitive information, transformation functions can serve to protect privacy. This may involve fuzzifying values, e.g. bucketing respiratory readings into a 'dashboard' representation [e.g. Stable, Concern or Emergency states], and location coordinates into [Home or Not Home] as opposed to providing the precise GPS or room coordinates. In an emergency situation, the perturbing transformations do not apply, thus the complete details are propagated to facilitate better care. However, in an emergency, information of prior readings might not only be relevant, but vital to treatment. As such, appropriate historical events should be replayed, but with detailed (unperturbed) information. This provides the appropriate granularity of information to the care manager, A&E, etc., in the circumstances required.

These scenarios are used to show how replay mechanisms interplay with context to control access to sensitive data.

4. EVENT REPLAY

Event-based middleware focuses on the delivery of events as they occur within a system. However, there are situations where principals require historical information. This requirement typically stems from some incident: perhaps to update an information store after a period of disconnection (mobility); perhaps the occurrence renders previously 'uninteresting' events (e.g. those restricted by filters) relevant, or perhaps the incident triggers some application-level need for historical data, such as the investigation of a prescriber.

One potential approach is for principals to query the datastores of the information producers directly for the required historical data. However, in a distributed, event-driven environment with content-based routing [8] the potential decoupling of producers from consumers may preclude reliably discovering the complete set of producers. By handling historical requests in the middleware, the requesting principal

[2]The auditor is responsible for the information received.

is not burdened with the (perhaps impossible) task of uncovering and directly querying every potential information source. Although each broker in our PostgreSQL-PS deployments provides a database API, distributed communication occurs through the publish/subscribe API.

There is some work concerning historical events in publish/subscribe systems. Cilia et al. [3] describe the use of history buffers to allow event republication to deal with bootstrapping and disconnection in mobile systems. Muehl et al. [10] also deal with mobility, mentioning that event histories can be managed either by producers or brokers. Li et al. [8] take a general approach, where databases are connected to various brokers, each associated with a filter to store particular information. The database holding the relevant information republishes historical events on receipt of a subscription query with a historic time-based parameter.

Previous work demonstrates the value of event replay in publish/subscribe, but tends not to address security concerns. Access control policies are context sensitive, which means the content of, and/or flow (restrictions) on, a historical event may differ due to a change in circumstance. As such, it is necessary to consider the impact of *context* on historical republication, and its effects on information disclosure. Replay control need not only restrict, but may also enrich information flows. For example, where a particular event (emergency) triggers the release of a set of unperturbed historical events (sensor readings).

5. EVENT-REPLAY ACCESS CONTROL INFRASTRUCTURE

Replay semantics are defined in broker-specific policy rules which control the circumstances for data release.

5.1 Fluents

A *fluent*, as defined in Event Calculus [7], is a reified, half-open interval in some given time domain. It represents a particular state of affairs holding (non-exclusively). The simplest canonical forms of Kowalski's Event Calculus provide a predicate `holds-at`, that reports whether a particular fluent holds at a particular point in time.

Fluents are a useful representation of context; for instance, a fluent can be defined to refer to a particular patient as being in a critical situation, or a prescriber being under investigation. Fluents can potentially encapsulate complex representations of state (i.e. composite events), in that a number of events might serve to alter the state of a fluent.

Replay semantics are defined with reference to time ranges (i.e. the range in which events are republished), and thus also potentially with reference to fluents. From a policy perspective, generally it is an occurrence, or change in context (fluent state) that defines an interest, not the value of the time-point at which the triggering event occurs. In our implementation, fluents are used to facilitate more natural policy expressions: policy rules can include a named fluent instead of a time-stamp range. As for event definitions, fluent definitions may range in scope from local to global.

5.2 Replay Request

Principals require the ability to request access to historical information. Such a request is defined as a `REPLAY` message, which at the very least has an attribute indicating the `Event Type` being replayed. Most replay messages will limit the scope of replay[3]. `From` and `To` attributes can provide timestamps that bound the time line on one or both sides. Also the `During` attribute can filter events that do not occur when the named fluent is active. `Not during` inverts the fluent match. Finally, a `filter` attribute can provide a conditional clause akin to a subscription filter.

The filter must encapsulate any *mandatory attributes* specified by the policy author, which are attributes that the requester must include as part of their filter. For example, policy might specify that any replay request for a `prescribe` event must include the ID of the prescriber. This allows reasoning about the motivation behind a request—to make more informed authorisation decisions—while ensuring that requests are properly constrained.

Note that a replay request is essentially a subscription to past information, while a general subscription is a query over future events. As such, subscriptions and replay requests can be unified, where a general subscription request includes a `From` tag to signify that the subscription should replay prior events as well as delivering new events as they occur.

5.3 Interaction Control & Event Replay

Interaction control policies are enforced in a broker, giving the management domain fine-grained control over the information released. Replay requests are logically equivalent to subscription requests, except that they refer to prior events. As such, the same controls can be used for both.

Here we consider the details of authorisation and restriction rules. Authorisations tend to define general privilege. These may be qualified by one-or-more restriction rules, which cater for particular situations, e.g. to satisfy the request of a particular patient. This separation brings flexibility, as there may be a many-to-many mapping between the rules types. Authorisation rules take the following form[4]:

```
ALLOW [SUBSCRIPTION TO | REPLAY OF] (event type)
 WHERE (validation conditions)
```

In this rule, the choice of `subscription to` or `replay of` is merely a lexical concern, since they are equivalent in terms of implementation. The rule is specified over a particular `event type`. Finally, a set of `validation conditions` is included. This conditional clause can access parameters of the request, which can be used to verify that the principal holds particular credentials for the rule to apply, that fluent values and mandatory attributes were appropriately specified, that the time range is valid, etc. As this condition is a SQL statement, it may access a variety of functions, operators and services, allowing for rigorous validation.

Restrictions may be imposed that filter the delivery of events. Restrictions take the following form:

```
RESTRICT EVENT DELIVERY OF (event type)
 WHERE (validation conditions) FILTER (filters)
```

The same definitions exist for event type and validation conditions, however `filters` specify the restrictions that are imposed (enforced) on the delivery of an event instance.

Transformations convert an event instance into the specified output type, either on publication or notification (deliv-

[3]The requested scope may be limited by authorisation and restriction rules—see §5.3.

[4]Our implementation represents policy rules in XML, however we are moving toward a syntax usable from the database console—as presented here.

ery) of the instance. Transformation rules take the following form:

```
TRANSFORM EVENT (event type) TO (output event type)
 ON (publish | notify) EXECUTE (function)
 WHERE (validation conditions)
```

The transform occurs through the specified `function` when the `validation conditions` are met. Unless otherwise defined (through validation conditions), transformation rules are triggered by both general and replayed events[5].

Authorisation rules are evaluated on receipt of a replay or subscription request. Multiple policy rules can apply to a particular request: enforcement and conflict resolution involves overriding rules or aggregating restrictions (see [15] for details). If the request, which satisfies an authorisation rule, encompasses historical information, any prior events matching the requester's query and set of applicable restriction (policy) filters are republished. If the request also concerns future events, then the subscription channel is established with the appropriate restriction filters imposed [15]. Events are delivered, subject to any applicable (and resolved) transformations.

Republished events are transmitted with some meta-data, including the time of original publication.

5.4 Automatic Replay

There may be situations where a broker should automatically republish historical events. This might be to assure a quality of service, e.g. to minimise loss in cases of disconnection, or in situations where a change in context interacts with the access control mechanisms to alter the flow of information. That is, a change in context might cause a certain restriction or transformation to no longer apply, and thus prior events should be republished without restriction.

The definition of an automatic replay rule is similar to a request, except that it defines the triggering condition and the target. Rules are specified as follows:

```
AUTO REPLAY EVENT TYPE (event type)
 (FROM|TO) (timestamp)
 ((NOT) DURING) (fluent)
 WHERE (validation conditions)
 FILTER (filter)
 ON EVENT (event) WHERE (execution conditions)
```

The event type, time, fluent, validation conditions and filters have been described previously. The final line of this listing describes the event instance, and associated conditions, that trigger the event replay. Replayed events will not trigger an automatic replay rule, unless explicitly specified in its execution conditions. These rules are loaded on a subscription request (for future events), where it meets the validation criteria specified in the rule. Historical events in the defined range are delivered upon the occurrence of a triggering event matching the defined conditions.

5.5 Propagation

These rules allow definition of the circumstances for information propagation from the perspective of a local broker. Clearly, such rules can also apply to centralised, point-to-point architectures. However, in a distributed publish/subscribe system, a replay request is routed through the network in the same manner as a subscription. Historical information is released by brokers serving such events, if authorised by their disclosure policy. In order to avoid duplicates, only a broker hosting an information publisher should propagate the historical information of that publisher. Similarly, various brokers might load replay rules for a particular subscription, providing the subscriber with information, possibly from multiple sources.

[5]Note that transformed events (outputs) do not trigger other transformation rules at the same interaction point [15].

6. SCENARIO APPLICATION

We return to the previously described healthcare examples and consider the event replay rules and conditions used in such scenarios.

Drug auditing. When the nurse falls under investigation, the auditor requires historical information of the patients to whom she supplied controlled substances. The access control rules involve a transformation, which removes patient details from the event delivered to the auditor. However, this is conditional, applying only to situations where the prescriber is not under investigation[6]. When the nurse is under investigation, her future prescription events should contain patient details. To obtain historical information, the auditor can issue a replay request defined using the `underInvestigation(nhsid)` fluent, as shown in Figure 2(a). This request propagates through the network, where each domain (broker) republishes all relevant (controlled drugs issued by the prescriber) `prescribe` events. The authorisation rule shown in Figure 2(b) ensures that a request is for a specific prescriber[7], while the restriction rules ensure that the auditor only receives events pertaining to restricted drugs, regardless of whether they are historical. The system combines the request and restriction filters, determining the events to release (if any). In this case, events where a drug is supplied will be replayed, if the requester is an auditor and specifies the prescriber of interest. As the nurse (prescriber) is under investigation, the replayed events pass through unperturbed; thus the auditor receives information from all domains (in this case, a number of surgeries) on patients to whom the nurse prescribed controlled drugs.

Sensor obfuscation. For reasons of privacy, transformation functions alter the granularity of sensor readings received by subscribers. However, in emergency situations, it is important that those subscribed to sensor streams receive complete, detailed (unperturbed) information. As the transformation is performed on notification, the database stores the unperturbed event instances. Given the imperative nature of an emergency situation, a domain might decide it important to automatically replay the sensor events from several hours before the patient reached a critical state. When an emergency situation is detected, an `emergency` event for the patient triggers event republication. This replay mechanism provides care staff with detailed sensor readings in the period leading up to the emergency situation, to help improve patient care. Figure 2(c) shows the rule definition that replays readings to all subscribers to the patient's sensor stream from two hours before the emergency.

[6]This is generally subject to consent. Though omitted due to space, consent can be verified through rule conditionals.

[7]Although policy is domain specific, it is likely that policies will be consistent given the legislative basis.

```
<replay_request>
  <event_type>prescribe</event_type>
  <during not="T">underInvestigation(nurse_8821)</during>
  <filter>prescriber_id = nurse_8821</filter>
</replay_request>
```

(a) Request policy for prior prescribe events.

```
ALLOW REPLAY OF prescribe
  WHERE sub.prescriber_id <> NULL AND credentials(user, auditor)
RESTRICT DELIVERY OF prescribe
  WHERE credentials(user, auditor)
  FILTER isControlled(drug_id)
```

(b) The replay authorisation and restriction rules concerning prescribe events.

```
AUTO REPLAY EVENT TYPE physical_status FROM
  now() - interval '2 hours' TO now()
  WHERE sub.patient_id = nhs_patient_4122
  FILTER physical_status.patient_id = nhs_patient_4412
  ON emergency WHERE emergency.patient_id = nhs_patient_4412
```

(c) The automatic replay rule for emergency situations.

Figure 2: Policy fragments to drive the scenarios

7. FUTURE WORK AND CONCLUSION

Event replay is a useful feature of an event-based infrastructure, particularly in situations where changes in context alter data visibility and/or where information comes from multiple sources. In application environments with stringent data control requirements, mechanisms are required to control the circumstances of event replay. Such mechanisms not only control the situations for data release, but as shown in the two scenarios, serve to encode application-level semantics into the middleware—bringing about automatic enforcement and compliance checking.

This work serves as a basis for further exploration into historical event republication. Clearly, the rate of data flow is an important concern [10], as replaying historical information involves the propagation of a number of events at once. While remaining an area for future work, it would appear that rate/flow-control requirements could be integrated into the conditional segments of the replay authorisation rules. Events are replayed for particular time ranges, optionally refined by the truth of given fluents. Both the requester and administrator can filter the event stream. While intuitively flexible, more application scenarios are required to test whether historical requests are sufficiently expressive to meet the requirements of the healthcare domain.

We have presented a framework that unifies the delivery of both past and current events. This common interface for defining information disclosure policies gives administrators fine-grained control over the information released. The same infrastructure can react to historical requests and both new and republished events; altering context (fluent state), triggering actions (e.g. workflow processes), etc. The replay of an event is more than just a snapshot, it can impact on system state. The infrastructure presented demonstrates mechanisms to control event republication, describing how context can be exploited by policy administrators to best manage their data disclosure responsibilities.

8. ACKNOWLEDGEMENTS

We acknowledge the support of the UK EPSRC grant CareGrid (EP/C53718X). We thank Luis Vargas and others from the Opera Research Group for their contributions.

9. REFERENCES

[1] J. Bacon, D. M. Eyers, J. Singh, and P. R. Pietzuch. Access Control in Publish/Subscribe Systems. In *Distributed Event Based Systems*, pages 23–34, 2008.

[2] British Medical Association. Confidentiality and disclosure of information to PCTs in primary care settings, 2007.

[3] M. Cilia, L. Fiege, C. Haul, A. Zeidler, and A. P. Buchmann. Looking into the past: enhancing mobile publish/subscribe middleware. In *Distributed Event Based Systems*, pages 1–8, 2003.

[4] Department of Health (UK). Building Telecare in England, 2005.

[5] Department of Health (UK). Safer management of Controlled Drugs, 2007.

[6] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[7] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[8] G. Li, A. Cheung, S. Hou, S. Hu, V. Muthusamy, R. Sherafat, A. Wun, H.-A. Jacobsen, and S. Manovski. Historic data access in publish/subscribe. In *Distributed Event Based Systems*, pages 80–84, 2007.

[9] Lord Darzi. High quality care for all: NHS Next Stage Review, 2008.

[10] G. Muehl, A. Ulbrich, K. Herrmann, and T. Weis. Disseminating information to mobile clients using publish-subscribe. *IEEE Internet Computing*, 8(3):46–53, 2004.

[11] L. Opyrchal, A. Prakash, and A. Agrawal. Supporting privacy policies in a publish-subscribe substrate for pervasive environments. *Journal of Networks*, 2(1):17–26, 2007.

[12] L. I. W. Pesonen, D. M. Eyers, and J. Bacon. Encryption-Enforced Access Control in Dynamic Multi-Domain Publish/Subscribe Networks. In *Distributed Event Based Systems*, pages 104–115, 2007.

[13] J. Singh, J. Bacon, and K. Moody. Dynamic trust domains for secure, private, technology-assisted living. In *ARES*, pages 27–34, 2007.

[14] J. Singh, L. Vargas, and J. Bacon. A model for controlling data flow in distributed healthcare environments. *Pervasive Health*, pages 188–191, 2008.

[15] J. Singh, L. Vargas, J. Bacon, and K. Moody. Policy-based information sharing in publish/subscribe middleware. In *POLICY*, pages 137–144, 2008.

[16] A. Tomasic, C. Garrod, and K. Popendorf. Symmetric publish/subscribe via constraint publication. Technical Report CMU-CS-06-129R, 2006.

[17] UK Crown. Data Protection Act (1998).

[18] UK Crown. The Controlled Drugs Regulations 2006.

[19] L. Vargas, J. Bacon, and K. Moody. Event-Driven Database Information Sharing. In *British National Conference on Databases*, pages 113–125, 2008.

[20] A. Wun and H.-A. Jacobsen. A policy management framework for content-based publish/subscribe. In *Middleware '07*, pages 368–388, 2007.