A Policy-Based Framework for e-Health Applications

Giovanni Russello, Changyu Dong, Naranker Dulay
Imperial College London
{g.russello, changyu.dong, n.dulay}@imperial.ac.uk

Jatinder Singh, Jean Bacon, Ken Moody
University of Cambridge
{firstname.lastname}@cl.cam.ac.uk

Abstract

In this paper we present a policy-based framework for the specification and enforcement of policies for e-Health applications. In the framework, authorisation policies are used to express the rights that entities have in the system, while obligation policies are used to define event-condition-action rules as well as the responsibilities that carers and patients have in the system. We show how to this approach can be used for Edema treatment and how fine-grained authorisation and obligation policies are expressed. Authorisation will often depend on the context in which patient care takes place, and our policies support predicates that reflect the environment.

1 Introduction

In the CareGrid project we are developing methods for enhancing the development and deployment of healthcare applications. Such applications comprise small-scale systems, such as body sensors carried by a patient, up to large-scale distributed systems, such as the network of a hospital managed by the national health service. Further, healthcare applications are often very dynamic presenting scenarios where a group of carers are assembled to take responsibility in a healthcare situation, such as a traffic accident; in technology assisted living, etc.

In this paper we focus on technology-assisted living, where the domains to be administered are units of personal living. Here, someone may be living alone, perhaps in sheltered housing, perhaps post-operative, perhaps with one or more disabilities [6], perhaps elderly and infirm. Healthcare applications are composed of several services to monitor the patient condition, and to assist the patient or the carer in performing the appropriate treatments.

Monitoring the physical condition of a patient is carried out by means of body sensors [7]. Several types of sensors are commercially available to measure blood pressure, blood sugar, pulse rate, etc. There are different aspects that must be taken into account when dealing with data gathered from sensors, such as gathering and logging the sensor data streams persistently, validate the data values, correlate them for the various different sources. Another form of monitoring can be achieved using infra-red cameras that avoid the invasiveness of video surveillance. What can be detected is the number of people (or animals such as guide dogs) that are present. If a person is motionless this could indicate a faint or a fall. An example is Irisys technology [4], where a system includes an infra-red camera and motion detection software. The system can also make a note of any visitors, in order to find out whether carers are visiting according to schedule. The data gathered from sensors could be used to detect critical conditions for the patient and raise an alarm to summon some help. The appropriate actions could be determined from stored policy.

Policies can also be used for assisting patients and carers for executing their tasks. For example, a carer may set the schedule for the various forms of medication a patient must take. This can be complex and confusing when someone comes home from hospital with a large array of new pills. If the person regularly uses a mobile phone, this could be used for executing a policy that sends an alert that a pill is due to be taken. Also, carers visiting in a day several patients with different conditions must be assisted for performing correctly their tasks. In this respect, a carer can be equipped with a portable PC that provides information for each patient she must visit: i.e., patient addresses, the types of treatment and procedures, etc. By executing the appropriate set of policies, the device can provide an on-line feedback to the carer during she is performing her duties and gathering medical data from each patient that can be analysed back in the hospital.

There is a great deal of commercial interest in providing technology for assisted living. Indeed, companies are beginning to market them, but as vertically isolated, ad-hoc products. What we are developing in the CareGrid project is a uniform framework supporting healthcare applications. In our framework each individual assisted housing unit, may have a computer with local persistent storage for sensor-data logging, policy storage etc. The patient and carers may be comfortable with mobile phones; client and server software may be written to run on these as well as on mobile or home computers. Technology for body-sensing must be integrated with the computer systems, utilising wireless and wired communication as appropriate.

Our framework is policy based. We provide a policy language for expressing the rights and obligations of carers as well as threshold values on sensed data and patterns of sensed data values that indicate that medical attention is needed. The rights are typically to control access to the patient and associated, (possibly sensitive) medical data. Obligations make clear what each party "contracts" to do. For example, a patient agrees to take medication before acknowledging that this has been done. A doctor agrees to authenticate on arrival and, before leaving, to record any treatment given. In this paper, we illustrate the feasibility of our approach by means of an implementation of a healthcare application based on our framework.

The outline of this paper is as follows. In Section 2 we discuss the fundamentals of our event-based middleware. In Section 3 we describe how entities are grouped and managed. Section 4 describes our policy language. Section 5 we provided a scenario for the Edema treatment realised with out approach. We conclude in Section 7.

2 Event-driven Middleware

Middleware lies at the core of the patient care framework, as it is responsible for the interactions between entities in a system. A goal of CareGrid is to design a middleware framework enabling communication whilst providing the capability to enforce constraints and obligations according to system policy. At University of Cambridge, we have worked on event-based middleware for some time [2], focussing on event-driven systems.

Events are used to drive the interactions between entities, firing upon actions and/or changes in context. As such, the middleware must provide event propagation mechanisms that are reliable, secure and correspond to circumstance and system constraints. A healthcare domain poses extra constraints upon an event-messaging framework, as there are situations resulting in grave consequences if incorrectly handled. For instance, an event flow might involve routing an emergency-response event to an ambulance team when monitoring equipment detects a cardiac-arrest. Such an event requires rapid transmission, and might be prioritised for processing over other event types deemed less urgent, and so forth. Thus, the event infrastructure must guarantee a level of service appropriate to the scenarios involved in a pervasive healthcare environment. An event framework provides the means for enforcing obligations (discussed in Section 4) within the system. By actively monitoring events, it is possible to evaluate behaviour, through a comparison of observed actions and system policy, as well as react to situations, for example, by performing compensatory actions when an obligation is left unfulfilled.

3 Managing Healthcare Entities

An assisted living scenario is highly dynamic, as entities, including medical professionals, carers and sensor equipment, may frequently enter and leave a particular care domain. This dynamism means the middleware must autonomously manage these entities, by identifying them, assigning the requisite privileges and triggering various events and/or obligation flows. Another aspect of entity management, particularly important in the healthcare domain, concerns the performance of an entity. Actively monitoring and evaluating the behaviour of an entity allows for a higher quality of service, by allowing active intervention whilst functioning as a deterrent against misbehaviour.

In our approach, the carers associated with a unit of sheltered housing are grounded in a number of domains. A domain structure is created for each unit; this structure manages the carers that enter the unit, together with the authorisation policies for system services that are available from the unit. Appropriate policies for healthcare applications have been described in [5]. Here we are concerned with enforcement of the policies. Entities are authenticated by means of credentials (appointment certificates) issued by the domains in which the carers are employed as proof of their qualifications and fitness for performing tasks. We envisage that such credentials might be carried on an electronically readable identification token, to be used for personal authentication on entry to a unit. There are several methods than can be used for verifying credentials presented by the carers. As described in [3], the certificate can be checked by call-back to the home domain (such as the hospitals). An alternative for peripatetic carers is to use a decentralised network of certification authorities and using the notion of *trust* for authenticating only trusted entities. At Imperial College we currently investigating methods for integrating this authentication service within our framework.

4 Managing Healthcare Applications by means of Policies

Policy-based management is an effective solution for the distribution, automation and dynamic adaptation of systems used for supporting medical environments.

In this section we introduce our policy-based framework and the types of policies that it supports.

4.1 Policy Interpreter

The policy interpreter that we use for supporting carers in home-healthcare is based on the Ponder2 policy language developed at Imperial College London [8]. The language supports the specification of policies that are rules governing the choices in the behavior of a system [11]. Policies that can be specified are obligation policies (event-condition-actions) and authorisation policies written in XML. The interpreter organises the entities on which policies operate in hierarchical domains of managed objects. A managed object has a management interface that the object has to implement in order to be managed by the interpreter. Domains allow the classification and grouping of managed objects in a hierarchy. Furthermore, domain paths can be used to address managed objects in policy specifications. Managed objects can be used to represent resources (e.g., data repositories, printers, X-ray machines, etc.), devices (e.g., sensors), and people (i.e., nurses, doctors, GPs, etc).

Managed objects can be associated with a set of data that the interpreter uses for carrying out management operations, such as authentication, authorisation and obligations. In our system, we group such information in a *Managed Object Template* (MOT). The MOT defines the type of managed objects to which it must be applied as shown in Figure 1-(a). In this example, a MOT is specified for managed objects of type `Doctor`. The MOT defines the set of credentials that an entity has to provide in order to be authenticated and identified as a doctor for the sheltered home. The MOT may define the set of authorisation and obligation policies that have been specified for its managed object type. In the example, authorisation policies are used to control the access rights of doctors in the domain structure where they are operating. Obligation policies are used to enforce to a certain extend the responsibilities that doctors have in the system.

When an entity wants to access resources of a domain structure it has to present to the interpreter's authentication service its set of credentials, as shown in step 1 of Figure 1-(b). The authentication service validates the credentials and finds the MOT that captures the set of presented credentials. Once the MOT is found (2), a corresponding *Managed Object Instance* (MOI) is created in the domain structure (3). When a

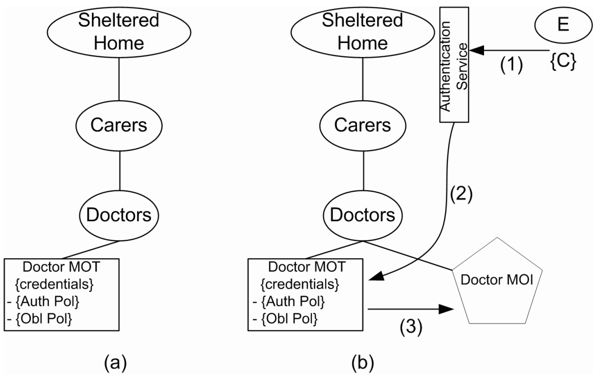

Figure 1. (a) MOT specification in a domain structure for sheltered home. (b) Instantiation of a Doctor MOI.

MOI is created the system activates all the policies defined in the MOT.

The use of MOTs caters for a more dynamic management of entities and policies. In fact, the set of credentials and policies that should be associated with a managed object can be dynamically retrieved from well-known repositories. For instance, in our scenario of sheltered home if the MOT of the doctor is not defined locally or is out-of-date it can be retrieved from a NHS repository. A domain structure can also import part of a MOT. Again as an example in the scenario in Figure 1, the domain structure could allow the device used by the doctor to provide the obligation policies that the doctor has to satisfy when visiting the patient. Such obligation policies are imported in the Doctor MOT of the domain structure of the sheltered home and activated when the doctor is authenticated.

4.2 Authorisation Policies

Authorisation policies are used for controlling the rights that entities have on the resources managed in a domain structure. In the case of a sheltered home these resources could be represented by the patient's medical data stored in the home PC, the body sensors that the patient wears, and any other application/devices that could be used to assist the patient.

Authorisation policies are defined on (subject,target,action)-triples, where the subject is the managed object that wants to execute the action on the target. In our framework, we also support negative authorisation policies, that when applied negate the execution of the defined action. Moreover, we support a fine-grained authorisation model that allows the specification of authorisation policies for independently controlling the subject and the target of an action [9]. Our framework is also able to autonomously resolve conflicts that could happen when authorisation policies of different signs apply to the same triple.

4.3 Obligation Policies

In Ponder, obligation policies are used to capture event-condition-action rules. Such policies are used to dynamically adapt the system to changes of either context or behaviour of applications. Events are trigged by such changes and are propagated using our event-based middleware. Obligations polices capture such events and execute actions for adapting the system. In a medical scenario, such policies can be used for adapting the environment of the patient's home, i.e. rising or lowering the temperature of the rooms. Obligation policies can also be used for sending out alarms in case some critical condition is detected. For example, let us consider a patient that wears a heart-rate sensor that can detect cardiac anomalies. If the anomaly is critical an ambulance could be dispatched to help the patient to overcome the crisis. We can define an obligation policy for such an event as follows:

Policy 1 *shows an obligation policy that captures the* `HrtAnomalyCritEvt` *sent by the heart-rate sensor when it detects a critical* *anomaly. In this case, the policy sends a request to a hospital for dispatching an ambulance, providing the patient's address and the type of condition.*

```
oblig AmbulanceDispatchPolicy
   on HrtAnomalyCritEvt
   do hospital.dispatchTo(patient.addr,
                               condition.CRITICAL);
```

However, another important aspect of obligation policies is to encode the responsibilities of people in the system. In this respect, obligations are particularly relevant in a healthcare applications, where failure to perform an action may result in serious consequences. Obligations work to ensure that the relevant entities are aware of their duties and act accordingly; and also provide a safety-net by allowing for provision of contingencies for a failure in responsibility (i.e. an unfulfilled obligation).

Usually, such responsibilities are bound with the notion of time, i.e. the time that the carer spent at the patient's home, or the time of the day that a medication should be taken. For this reason, we provide a timer service that can be programmed to set events after a given interval of time expires. Obligation policies can be expressed in terms of these events and can execute actions either to help entities to fulfill their responsibilities or to provide alternative solutions in case of a failure of responsibility.

Consider an example where a patient needs to take a medicine every 4 hours. The system could send an alert event to remember the patient that the medicine should be taken within a certain amount of time. To acknowledge that the medicine was taken the patient wears a swallowing motion sensor to detect the medicine intake. When the sensor detects that the patient swallowed his medicine fires a event. Here, the responsibility lies with the patient to take their medicine. However, if no event from the sensor is received after an interval of time from the alert, the system infers that the patient fails to take the medicine and it might send an alarm to the patient's carer, or the emergency services, depending upon the gravity of the situation. To correctly express an obligation policy that captures this type of situations we need to define *constraints* on the sequence of events. Moreover, for each event different actions need to be taken. For doing this, we use an extended syntax for defining obligation policies that allows us to define composite events and actions in a compact and readable format. The obligation policy is specified as follow:

Policy 2 *shows an obligation policy that captures the scenario presented above.* `MedSwallowedEvt` *is sent by the swallowing motion sensor [1] when the patient swallows the medicine. The action of the policy is to log the time when the medicine is taken. The policy also specifies the sending of two events,* `AlertEvt` *and* `AlarmEvt`*, using the "->" operator. The sending of these events is constrained by the occurrences of* `MedSwallowedEvt`*. In particular, the policy defines that* `AlertEvt` *can only be sent after 3 hours and 50 minutes from the last occurrence of* `MedSwallowedEvt`*, unless another occurrence of* `MedSwallowedEvt` *is seen. When* `AlertEvt` *is sent, the policy captures it and sends to the patient a message to alert him that in 10 minutes the medicine should be taken. A similar constraint is defined for* `AlarmEvt`*, however this event is sent after 4 hours and 10 minutes from the last occurrence of* `MedSwallowedEvt`*, unless an occurrence is already seen before. When the policy captures* `AlarmEvt` *it sends a message to the patient's GP to notify that the medicine was not taken within the specified interval.*

```
oblig MedicineInTakePolicy
   after 3h 50min from MedSwallowedEvt -> AlertEvt
                   unless MedSwallowedEvt
   after 4h 10min from MedSwallowedEvt -> AlarmEvt
                   unless MedSwallowedEvt
   on MedSwallowedEvt
     do log("medicine taken", time);
   on AlertEvent
     do alertPatient("Medicine in 10 min");
   on AlarmEvent
     do sendAlarmToGP("No medicine taken");
```

5 Implementation of a home care scenario for Edema

In this section, we present the implementation of a simple healthcare application for the Edema treatment based on our framework.

Edema refers to swelling caused by excess fluid in body tissues. Monitoring involves a carer taking various measurements/performing tests. The scenario comprises several steps:

1. Authentication of the carer when entering the home healthcare domain.
2. Careplan for Edema treatment assigned to the carer
3. Carrying out treatment procedures
4. Departure of the carer from the home environment

For each of these steps we provide details on how they are implemented in our framework and some examples of authorisation and obligation policies used in the scenario.

5.1 Deployment

Carers are provided with portable devices that help them in performing their tasks. In the scenario shown in Figure 2, a nurse carries with her PDA with its own domain structure where a `Nurse` MOI is already instantiated.

The domain structure for the sheltered home is instantiated on the home PC. In the sheltered home, resources (such as body sensors and records) are provided as MOIs that can be accessed by other authorised MOI. The structure also provides a MOT for `Nurse` managed objects that provides authorisation and obligation policies required for the treatment of Edema pathology.

5.2 Entering the Home Domain and Assignment of Careplan

When the nurse enters the patient household, her PDA and the home PC discover each other. The discovery of the home domain will raise an event on the nurse device that triggers an obligation policy. This policy forces the local nurse MOI to retrieve the careplan from the `Treats` MOI in the sheltered home domain.

However, before this is achieved, the nurse MOI has to be authenticated. The steps that are executed are shown in Figure 2: (1) the nurse MOI presents its credential to the authentication service of the home domain; (2) after the credentials are verified the service finds the matching MOT; (3) the MOT creates the MOI and activates the set of obligation policies to follow the Edema treatment and the authorisation policies necessary to access the required resources; (4)

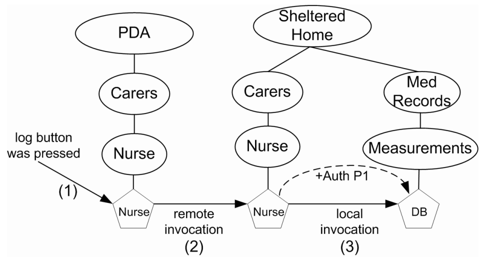

Figure 3. Controlling access to resources via authorisation policies.

once the nurse MOI is created locally the careplan is retrieved and (5) returned to the nurse MOI in the nurse device. Through her device, the nurse can follow the Edema treatment procedures for this particular patient.

The nurse MOI instantiated on the sheltered home domain acts as a proxy for the nurse MOI running on the nurse's device. This allows the specification and enforcement of policies locally to the domain structure where the resources are instantiated.

5.3 Treatment Procedures

Edema treatment involves the following procedures: take measurements and record results. The measurements include blood pressure, weight, pitting test, and body measurements. These measurement can be performed using devices and body sensors available as MOI in the home domain, such as the scale and the blood pressure sensor (BPS) in Figure 2. In this case, the recording of the measurement is done automatically by the device itself.

However, the results of other measurements need to be logged manually by the nurse (i.e. body measurements). For instance, the nurse can use her device to enter the result of the measurement and press a 'log measurement' button that will appear on the screen. As shown in Figure 3, once the button is pressed (1) the nurse MOI will try to connect to the database in the home domain. The invocation of the action on the database goes through the nurse MOI on the home domain (2). The local nurse MOI must be authorised to execute such an action. In other words, there must exist an active authorisation policy that allows the nurse MOI to access the

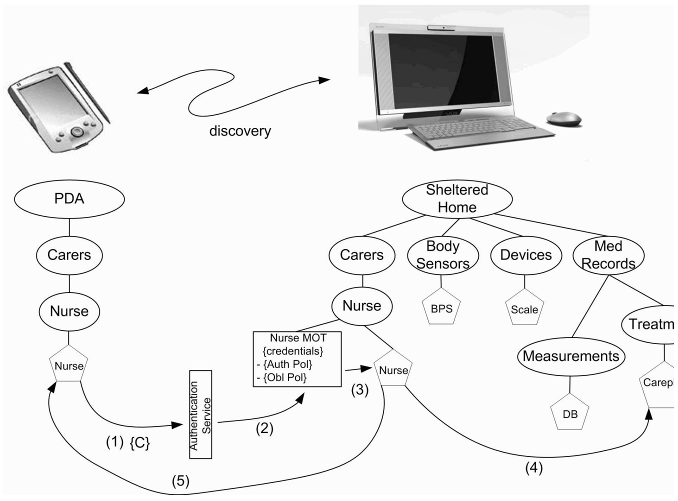

Figure 2. Domain structures for a carer device and the shelter home devices.

database for recording the measurements. As Figure 3 shows, such a policy exists (policy $P1$ shown as a dashed arch from the nurse MOI to the database MOI) and it is specified as part of the nurse MOT shown in Figure 2. Therefore the system will allow the action of the nurse MOI on the database MOI (3). The authorisation policy $P1$ is activated when the nurse MOI is instantiated and it is specified as follows:

Policy 3 *a positive authorisation for the nurse MOI to access the method* `recordMeasurement()` *of the database MOI.*

```
auth+ P1:
nurseMOI→databaseMOI.recordMeasurement()
```

To avoid the nurse leaving the patient's home without having completed the treatment procedures we can use obligation policies to help the nurse to fulfill her responsibilities. Obligation policies can be used to remind the nurse via alerts of the tasks that need to be completed. In this case, the alert should reach the nurse some time in advance before her expected departure time from the home domain. On the other hand, if the nurse left the home without completing all her expected tasks, then the unfulfillment of her task must be reported.

For instance, let us consider the body measurement task for checking body swelling. We can define in the nurse MOT the following obligation policy:

Policy 4 *shows an obligation policy for the body measurement task. The* `BMTakenEvt` *is sent when the nurse manually inserts the results of the patient measurements and presses the confirmation button. This event represents the fulfillment of the task. The policy can also alert the nurse that she has 30 minutes left to get the measurements before she must leave for the next patient. Assuming that the nurse needs 1 hour to complete the Edema treatment, the policy sends the alert event after 30 minutes that the nurse has been detected in the home domain (indicated by* `NurseInEvt`*). If no body measurements were taken after her departure (indicated by* `NurseOutEvt`*) then the policy will alert the nurse after 10 minutes that she left without taking the body measurements, unless she enters the home domain again. On the other hand, after 30 minutes that the nurse left the home domain without taking the body measurements the hospital is notified, unless the nurse left due to an emergency (indicated by* `EmergEvt`*).*

```
oblig BMPolicy
    after 30min from NurseInEvt -> AlertBMEvt
                    unless BMTakenEvt
    after 10min from NurseOutEvt -> Alert2BMEvt
                    unless NurseInEvt or BMTakenEvt
    after 30min from NurseOutEvt -> HNotifyBMEvt
                    unless EmergEvt or BMTakenEvt
    on BMTakenEvt
      do log("MB taken", time);
    on AlertBMEvt
      do alertNurse("BM within 30 min");
    on Alert2BMEvt
      do alertNurse("Departed without BM");
    on HNotifyBMEvt
      do sendNotToHosp("No BMtaken");
```

5.4 Leaving the Home Domain

When the nurse completes her visit and leaves the patient's home the resources allocated for her in the home domain must be relinquished. For instance, the nurse MOI should be removed from the home domain and the policies associated with it deactivated. These "garbaging" operations are executed by an obligation policy that is triggered by the event `NurseOutEvt`.

Policy 5 *shows an obligation policy that captures the* ***NurseOutEvt*** *sent when the nurse leaves the patient's home. The obligation policy removes the nurse MOI and disables the policies associated with it.*

```
oblig NurseLeavePolicy
on NurseOutEvt
do nurseMOI.disablePolicies;
    home.remove(nurseMOI);
```

The `NurseOutEvt` event can be sent in several ways:

- manually by the carer: the nurse presses a button to notify that the treatment is completed (according to her) and she is ready to leave.
- automatically by the system: when the nurse's device is out-of-range from the home PC or after a certain amount of time allocated for the carer expires.

These methods can be used in different situations. For instance, if the nurse has to leave before completing her procedures due to an emergency in another sheltered home, then either the manual disconnection or the out-of-range detection could allow the system to promptly react (i.e., the system could report via an alarm that some procedures need to be completed). On the other hand, the time-out disconnection can be used by the system to recover from erroneous situation (i.e. the nurse left behind her device without notifying the system when she departed from the patient's house).

6 Other Services

Ina addition to the policy framework, the CareGrid architecture includes:

- A trust framework for modelling and coordinating trust decisions a cross collaborating entities using autonomous domains. Trust domains can be federated and/or grouped in hierarchical or P2P fashion. This requires protocols for group-membership and trust negotiation, as well as an overarching architecture that is self-managing. Requests for trusted interactions are forwarded to the local trust domains, which is responsible for determining whether the interaction should succeed or fail. This can involve negotiation with the trust domain of the requester as well as other trust domains. The trust domain is also capable of providing a signed statement of the reasoning for success or failure. If a request succeeds, the system will typically establish a new secure channel and trigger any necessary security adaptations, e.g in the communications or access control systems.
- A monitoring and audit service for recording the actions of principals. Monitoring acts to regulate behaviour in and of a system, but in a proactive manner. By actively observing the actions (events) occurring within the system, it is possible to react in response to particular situations. Auditing is particularly important in a pervasive healthcare domain, where many entities, each with different motivations and purposes, interact as part of the healthcare process. System reliability and performance should also be monitored.
- Integration with the national electronic health record (EHR) service, as achieved via

a connection to an EHR metadata server, through which patient records may be read and updated. We have worked on the integration of event-based middleware with active databases [12]. Databases are components of our healthcare architecture and they can be accessed by means of a managed object interface presented in the domain structure. Databases provide logging sensor data locally within a sheltered housing domain. Our current work is concerned with processing these streams of sensor data, firstly for logging in a database component and secondly for detecting values of interest within a stream.

7 Conclusions and Future Work

In this paper, we have presented a policy-based framework for the specification and enforcement of policies for e-Health applications. In the framework, authorisation policies are used to express the rights that entities have in the system, while obligation policies are used to define event-condition-action rules as well as the responsibilities that carers and patients have in the system. We showed how to the approach could be used for Edema treatment and how fine-grained authorisation and obligation policies are expressed. Authorisation will often depend on the context in which patient care takes place, and our policies support predicates that reflect the environment. Obligations and responsibilities of entities must also be expressed precisely.

We are currently investigating how to extend the notion of obligation policies to use a workflow language to capture the more complex orchestrations needed in healthcare treatments. We are also investigating how the notion of trust can also be associated with the authentication process. Trust allows a more dynamic authentication mechanism where the entity being authenticated is not known to the system in advance.

Acknowledgments

This research was supported by the UK's EPSRC research grant EP/C537181/1 and forms part of the CareGrid, a collaborative project between the University of Cambridge and the Imperial College London. The authors would like to thank the members of the Policy Research Group at Imperial College for their support.

References

[1] O. Amft, G. Troster "Methods for Detection and Classification of Normal Swallowing from Muscle Activation and Sound." In *1st International Conference on Pervasive Computing Technologies for Healthcare*, December 2006.

[2] J. Bacon, D. Eyers, K. Moody, and L. Pesonen. "Securing publish/subscribe for multi-domain systems." In *Middleware 05*, Nov. 2005.

[3] J. Bacon, K. Moody, and W. Yao. "Access control and trust in the use of widely distributed services." In *Middleware 01*, IFIP/ACM International Conference on Distributed Systems Platforms, volume 2218 of LNCS, pages 295310. Springer, Nov. 2001.

[4] Irisys. http://www.irisys.co.uk

[5] Y. Liu and J. Bacon. "A practical synthesis of dynamic role settings in telecare services." In *First International Conference on the Digital Society (ICDS07)*, page 5. IEEE, Jan 2007.

[6] Y. Liu, J. Bacon, and R. Wilson-Hinds. "On smart-care services: Studies of visually impaired users in living contexts." In *First International Conference on the Digital Society (ICDS07)*, page 32. IEEE, Jan 2007.

[7] B. Lo and G. Z. Yang. "Key technical challenges and current implementations of body sensor networks." In *IEEE Second Workshop on Body Sensor Networks (BSN 2005)*, April 2005.

[8] The Ponder2 Project http://ponder2.net

[9] G. Russello, C. Dong, N. Dulay. "Authorisation and Conflict Resolution for Hierarchical Domains." In *proc. of Policy07*. June 2007.

[10] M. Sloman, J. Magee and K. Twidle and J. Kramer. "An Architecture for Managing Distributed Systems." In *Proc. 4th IEEE Workshop on Future Trends of Distributed Computing Systems*, pp. 40–46, 1993.

[11] M. Sloman and E. Lupu. "Security and Management Policy Specification." In *IEEE Network*, pp.10–19, Vol. 16, Issue 2, March, 2002.

[12] L. Vargas, J. Bacon, and K. Moody. "Integrating Databases with Publish/Subscribe." In *Proceedings of the 4th International Workshop in Distributed Event-Based Systems (DEBS05)*, pages 392397. IEEE Press, June 2005.