Kenneth J. Turner, Gemma A. Campbell and Feng Wang.
Policies for Sensor Networks and Home Care Networks, in Mohammed Erradi (ed.),
Proc. 7th. Int. Conf. on New Technologies for Distributed Systems, pp. 273-284,
ISBN 9981-9704-7-6, June 2007

Policies for Sensor Networks and Home Care Networks

Kenneth J. Turner — Gemma A. Campbell — Feng Wang

Computing Science and Mathematics, University of Stirling, Scotland FK9 4LA
Email: kjt@cs.stir.ac.uk, gca@cs.stir.ac.uk and fw@cs.stir.ac.uk

ABSTRACT. *This paper describes how a policy-based management system for call control has been generalised and extended, making it applicable to sensor networks and home care networks. The background to these domains is introduced. The previous ACCENT policy system is described, along with the restrictions that previously limited its suitability for other domains. The generalisations needed to the policy system are described. This now relies on layered ontologies that describe common and domain-specific features of the policy system. Conflicts among policies are now handled more uniformly, including tool support for semi-automated detection of policy conflicts. The structure of policies for sensor and home care networks is outlined, together with sample goals and policies that illustrate the new policy languages.*

RÉSUMÉ. *Cet article décrit la généralisation et l'extension d'un système de gestion d'appels par politiques. Nous présentons l'adaptation de ce système à deux domaines : les réseaux de capteurs et les réseaux de télésoins à domicile. Nous présentons les concepts fondamentaux de ces deux domaines, le système de gestion de politiques ACCENT, ainsi que les restrictions qui avaient précédemment limité son utilisation à d'autres domaines. Nous décrivons les généralisations dont le système de politiques avait besoin. Le système utilise dorénavant des ontologies en couches pour la description des fonctionnalités communes et orientées-domaine. Les conflits entre politiques sont maintenant traités de manière plus uniforme avec un outil de détection semi-automatique. La structure des politiques pour ces nouveaux types de réseaux est présentée. Des exemples de buts et de politiques sont aussi fournis.*

KEYWORDS: *home networks, policy-based management, sensor networks, telecare, wind farms*

MOTS-CLÉS : *gestion par politiques, parcs éoliens, réseaux à domicile, réseaux de capteurs*

1. Introduction

1.1. *Motivation*

The principal author and his colleagues have previously developed an approach to policy-based management on the ACCENT project (Advanced Call Control Enhanc-

ing Network Technologies, *http://www.cs.stir.ac.uk/accent*). This project developed a policy language called APPEL (ACCENT Project Policy Environment/Language). The work of ACCENT was focused on policies for call control in (Internet) telephony, although it was believed that the approach would apply to other kinds of systems.

This paper reports further development of the ACCENT work for novel application domains. Specifically, it investigates the changes for policy-based management of sensor networks and home care networks. Although these are distributed systems like call control, it will be seen that significant generalisation of the approach has been necessary. Both new domains have their own unique challenges and requirements.

1.2. *Context*

Policy-Based Management Policies express how the behaviour of a system is governed. Policy-based management is typically used in areas such as access control, network management and system management. Policies express higher-level objectives that are automatically enforced. Predefined policies allow a system to dynamically adjust its behaviour without requiring user intervention at run-time. Numerous policy languages have been defined. Ponder (Damianou *et al.*, 2000) is a well-known example that has been applied in areas such as system management. KAoS (*http://www.ihmc.us/research/projects/KAoS*) and Rei (*http://www.cs.umbc.edu/~lkagal1/rei*) are policy languages with applications in access control and the semantic web. Neither applies directly to the monitoring and control required for sensor and home networks.

Many policy languages fall into the category called ECA (Event-Condition-Action). In this approach, policies are activated by events (triggers). If the conditions on a policy are satisfied, the policy causes the specified actions.

There are many variations on these basic approaches. Most policy languages are able to express authorisation, interdiction, obligation and permission. Some languages deal with the subject of a policy (that performs an action) and the target of a policy (that is acted upon). The notion of a policy domain is also supported by some languages, allowing policies to be defined hierarchically in nested or overlapping domain.

Policy conflict is almost inevitable when using policy-based management. Such conflicts may arise at different levels. The policies of just one user may interfere with each other if the user has conflicting objectives, e.g. low cost as well as high quality. The policies of peer users may disagree, e.g. two call parties may have different views on the use of video. Policies defined in different hierarchical domains may also conflict, e.g. end user and professional carer may differ.

Ontologies An ontology classifies the terms and their relationships in some domain of discourse. As well as providing semantic interconnections between concepts, it also allows inferences to be drawn about properties and relationships.

OWL (Web Ontology Language (World Wide Web Consortium, 2004)) is a widely used language for defining ontologies. OWL provides a larger range of capabilities than any comparable language to date. It defines ontology classes (domain concepts), properties (class relationships) and individuals (class members, usually real data). OWL supports ontology inheritance and re-use.

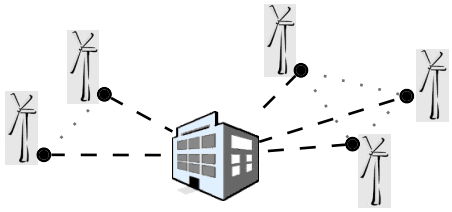

Figure 1. *Wind Farm Network*

1.3. *New Policy Domains*

The original work on ACCENT was conducted on call control for (Internet) telephony. The authors have extended and adapted the ACCENT approach to new domains: sensor networks and home care networks. The use of policy-based management techniques in these domains is novel. Policies for sensor networks are completely new, while only the UbiCare project (*http://www.ubicare.org*) has considered policies for healthcare monitoring.

Sensor Networks Sensor networks are distributed collections of communicating devices that monitor environmental conditions. They are used to collect a wide variety of data such as electrical, meteorological or mechanical conditions. Although sensor networks may be wired, they are typically wireless since this gives greater flexibility in device deployment and networking. The Berkeley Mote and its successors such as Mica 2 (*http://www.xbow.com*) are typical examples of sensor nodes.

Condition monitoring is a widely used in industry to verify the continuing performance of a system. The equipment being monitored is typically mechanical or electrical in nature, such as a bearing or a generator. Sensor networks for condition monitoring are attractive, as they exploit low-cost sensors in flexible configurations.

The authors are part of the PROSEN project (Proactive Condition Monitoring of Sensor Networks, *http://www.prosen.org.uk*). This is developing general techniques for condition monitoring, using wind farms to evaluate the approach.

Figure 1 shows a wind farm network schematically. A processing node (shown as a blob) is situated at the base of each wind turbine. This node is linked to a variety of local sensors: weather conditions, blade vibration, gearbox oil condition, etc. Nearby processing nodes may correlate their data by communicating with each other (dotted lines in the figure). Processing nodes are able to compress and filter data before communicating it to a control centre (dashed lines in the figure). The processing nodes generate triggers for significant events (e.g. wind shear, gearbox overheating). These are sent to the policy system located in the control centre. Policies for the wind farm respond with appropriate actions (e.g. feather turbine blades or shut down the turbine).

The policy system is also responsible for goal-directed configuration. This allows high-level goals (e.g. minimise equipment wear) to be translated into operational policies (e.g. operate equipment below maximum levels, monitor blade and gearbox vibration, match power generation to power demand).

Home Care Networks Many technologies have been developed for home networks, mostly for home automation. Smart homes are largely focused on technology for

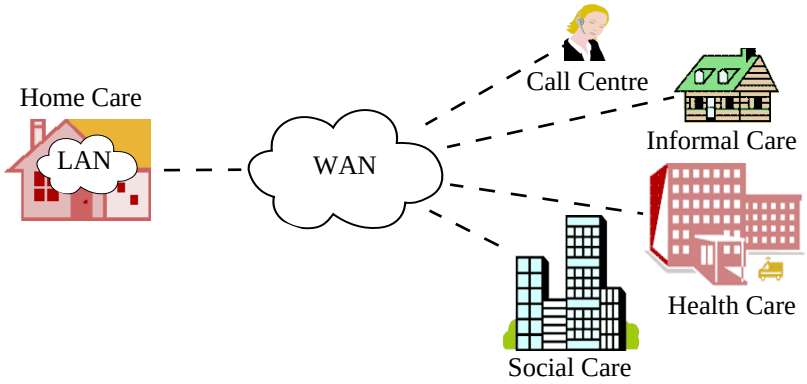

Figure 2. *Home Care Network*

the consumer. However, home networks are increasingly being used for telecare. Examples include the Millennium Homes (Perry *et al.*, 2004), the House_n project (*http://architecture.mit.edu/house_n*), and the SAPHE project (*http://ubimon.doc.ic.ac.uk/saphe*). Standards are being developed for telecare by the Continua Alliance (*http://www.continuaalliance.org*) and by ETSI committee STF 264.

The authors are part of the MATCH project (Mobilising Advanced Technologies for Care at Home, *http://www.match-project.org.uk*). This is developing general techniques for use of technology to deliver social and health care to the home. The aim is to prolong independent living for those who would otherwise have to go into a nursing home or sheltered housing. The authors are applying policy-based management techniques in this domain.

Figure 2 shows the home care network schematically. A LAN in the home links a variety of sensors and actuators to a residential gateway. The nature of the LAN is unimportant as many different technologies can be supported (e.g. wired and wireless, various home network standards). The sensors include movement detectors, pressure mats, smoke alarms and door switches. The actuators include appliance switches, mains supply shutoff, alert messaging and lighting control. The residential gateway supports OSGi (Open Services Gateway initiative, *http://www.osgi.org*). This is a service-oriented platform that is being used to develop a range of care services.

The residential gateway is linked to other locations via a WAN. The home is typically linked to call centres, social work departments, healthcare centres, and informal carers (e.g. family, neighbours). The nature of the WAN is unimportant, as many technologies can be supported (e.g. Internet broadband, GPRS, UMTS).

The policy system is also responsible for goal-directed configuration. This allows high-level goals (e.g. ensure the user is active) to be translated into operational policies (e.g. record movement around the house, verify the user shops regularly, check the user spends time outside the house).

Policies for home care reflect the views of a number of stakeholders, e.g. the user, the family, community nurses, social workers, and housing wardens. As these are all non-technical users, a user-friendly interface is vital. The multiple sources of policies

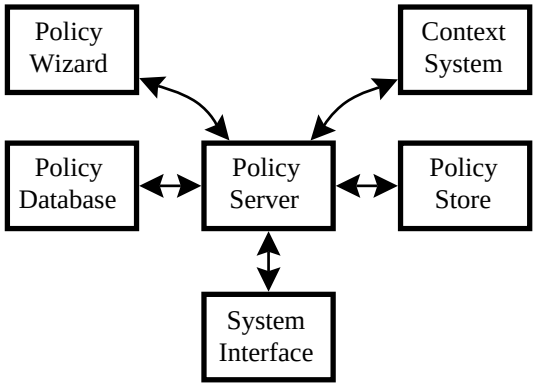

Figure 3. *Policy System Architecture*

also make policy conflict very likely. The support of conflict resolution is therefore particularly critical in this domain.

2. The ACCENT Approach

2.1. *System Architecture*

The ACCENT system supports the APPEL policy language. For generality, this has a core that defines the structure of the language. The core is then specialised for each application by defining the specific triggers, conditions and actions that apply in this domain.

As shown in figure 3, the ACCENT architecture is divided into three layers. This allows policy-based management to be kept independent of the system being controlled, and therefore supports applications in many domains. (Turner *et al.*, 2006) describes the general approach, while (Turner *et al.*, 2007) describes conflict resolution.

User Layer This layer contains components that interface directly with users. Many policy approaches require policies to be programmed in some language. Although this is appropriate for engineers and technical staff, it is unsuitable for ordinary end users. The ACCENT policy wizard is instead designed to allow non-computing users to create and edit policies. The wizard is web-based, which allows policies to be defined anywhere using a familiar interface. The wizard uses structured natural language and is multilingual. A number of aids facilitate policy definition by non-technical users.

As discussed in section 3, the work reported in this paper has addressed some significant weaknesses of the policy wizard developed by ACCENT. The latter was specific to call control, and could not easily be used in other applications. In addition, conflict resolution policies had to be created manually (in XML) and not by the wizard.

Policies are triggered by system events that carry information specific to the event. A context system allows additional information to be provided. For example, the context system may indicate the role of a user in an organisation, the user's current schedule, or the capabilities of a user.

Policy Layer This layer contains components that enforce policies. The policy server is the heart of the approach. This receives triggers from the system being managed, including associated information such as the triggering entity and any trigger parameters. The entity identity is used to retrieve all policies that apply to this trigger: policies specifically for this entity plus those for its domain. For example, the policies of a user, department and employer may all be retrieved by the same trigger.

The retrieved policies are filtered for applicability. Basic tests include whether a policy is enabled, has been triggered during its period of validity, and belongs to the user's current profile (e.g. at home, on holiday). The policy conditions are then evaluated. If they hold, the policy actions are sent to the system under control (e.g. turn on a fan). Triggers, conditions and actions may be compound (e.g. two triggers combined, alternative conditions, parallel actions).

The policy server is designed to be application-independent. Static data is held in the policy database (a conventional relational database), while dynamic data is held in the policy store (an XML-based tuple space). The policy server does not require knowledge of particular triggers and actions: these are specified only by policies. The policy server also maps between system concepts and policy concepts. This allows triggers and actions with different names in different applications to be represented by a common policy vocabulary. Conditions are also handled flexibly by the policy server. Each kind of condition is managed by a Java class that is discovered at runtime. It is therefore easy to add new conditions for new application domains.

The policy server also handles conflicts. Once all applicable policies propose their actions, these are filtered for conflict. For example, trying to turn a fan on *and* off is obviously contradictory. Crucially, the detection and resolution of policy conflicts is *defined* instead of being built into the policy server. This is done by means of special resolution policies that deal with conflicts among regular policies. By externalising conflict handling, the approach can be applied in any application domain.

System Layer This layer contains components in the system that are under policy control. An interface is required between the policy server and the managed system. This interface is responsible for intercepting significant system events that require policy management. For example, temperature and stress warnings in a wind farm are controlled by policies; these are reported as triggers to the policy server. The system interface suspends processing of the trigger until the policy server determines the appropriate actions. These are then enforced by the system interface. For example, a turbine may be monitored more regularly or may be shut down.

3. Generalising Support of Policies

3.1. *Ontology Framework for Policies*

As noted earlier, most of the software developed by ACCENT was application independent. The notable exception was the policy wizard, which had hard-coded knowledge of call control. Significant effort has now been put into generalising the wizard, making it suitable for any application.

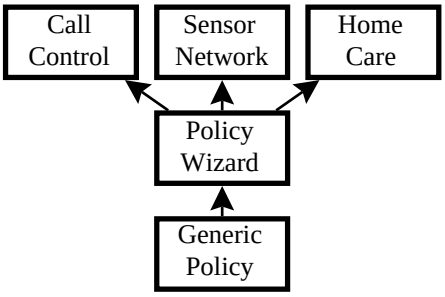

Figure 4. *Policy Ontology Stack*

Application-specific aspects of the wizard have been factored out into ontologies. Apart from generalising the wizard, it is beneficial to have a self-contained ontology for each application domain.

APPEL consists of a core language and its specialisations for each application domain. This is reflected in its ontological basis. At first sight, it would appear that a core ontology should be extended by domain-specific ontologies. However, the policy wizard supports concepts that are features of its interface rather than of APPEL. For example the wizard conforms to user skill level and understands units of measurement.

For this reason, two common ontologies were developed to support APPEL. The *genpol* ontology (generic policy aspects) supports the core of the language. This is extended by *wizpol* (policy wizard aspects) for the special features of the wizard. Both of these are domain-independent. A domain-specific ontology extends these by defining the triggers, conditions and actions that apply in some application. The resulting ontology stack is shown in figure 4. This exploits the ontology import feature of OWL.

Resolution policies have a very similar structure to to regular policies. A significant difference is that resolution policies are triggered by conflicting actions from regular policies. The triggers of resolution policies are therefore the actions of regular policies. This means that resolution policies have both common and domain-specific aspects. The conditions of resolution policies include those of regular policies and so also depend on the domain. Although resolution policies may have domain-specific actions, the approach also supports generic actions. If two policies conflict, priority might be given to the one that was defined earlier since it has existed for longer. Generic resolutions can also base priority on other factors such as choosing a higher-level domain policy in preference.

The generic policy ontology defines the structure of resolution policies. The details of triggers, conditions and actions must be defined by the domain-specific ontology.

3.2. *Supporting Ontologies for Policies*

The ontology framework defines only the structure of policy-related knowledge. Ontologies therefore contain no individuals or ontology class instances. Specific data values (e.g. trigger/action parameters, condition values) are defined by actual policies.

The POPPET system (Policy Ontology Parser Program – Extensible Translation) has been developed to support ontologies for policies. This has been implemented

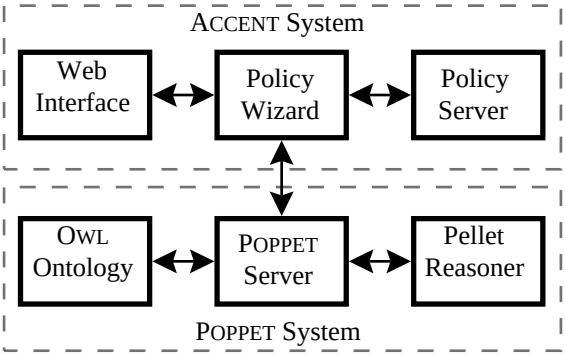

Figure 5. *Using Ontologies with Policies*

using Java as the programming language, Protégé as the OWL editor (*http://protege.stanford.edu*), Jena as the ontology parser (*http://jena.sourceforge.net*), and Pellet as the ontology reasoner (*http://pellet.owldl.com*).

Figure 5 shows how the ACCENT and POPPET systems interface with each other. POPPET runs on its own server, and is called by RMI (Remote Method Invocation). When started up, it parses an ontology document at a given URL and reasons about its contents using Pellet. A model of the ontology is constructed and stored for later queries. A remote application may then interrogate the stored ontology model using a variety of generic methods. When used by ACCENT software, queries to POPPET allow the wizard to determine the details of the domain-specific policy language (e.g. policy triggers and their categorisation for users).

Although POPPET has been designed to support ACCENT, the approach is general and is useful to other applications. For example POPPET handles the often complex issues in supporting an ontology, without requiring an application developer to grapple with these. POPPET also provides a common set of terms and their relationships that can be used by any application in the domain.

3.3. *Supporting System State and Configuration*

Because the ACCENT work focused on call control, state was not an issue for the policy system. This is because the controlled system (e.g. a SIP gateway or a telephone exchange) already maintains a full knowledge of state and configuration. Triggers are therefore generated according to system state (e.g. incoming call, call hang-up).

For sensor and home care networks, the situation is very different. Here, the sensors are simple devices that have limited knowledge of state. This requires the policy system to maintain the state history for use in policies. A wind farm policy, for example, might handle a blade vibration warning differently if recent wind conditions have been gusty. Similarly, a home care policy dealing with a heart rate alert might react differently if it is known the user is watching a scary movie.

Timing also play a limited role in call control since telephony systems already handle this issue. Sensor and home care networks require more explicit support of timing. For example, two turbine over-speed warnings within a minute might require

shutdown. A health visitor might be prompted to call if the user misses taking medication twice in one day. The policy system therefore records not only the state history but also the times at which state changes occurred. Policy conditions can then be expressed on system state, e.g. whether some component is in a particular state or whether some event has occurred within a given time period.

System configuration in call control is handled externally to the policy system. Call policies simply identify users by their addresses (e.g. a SIP address or a telephone number). Configuration is a much larger issue for sensor and home care networks. Both must be capable of self-configuration, for simplicity in initial installation and for adaptation to changes or failures. Goal-directed configuration is also required, so that sensors (and actuators) can be configured to meet policy requirements.

3.4. *Specialising APPEL*

APPEL can readily be specialised for new domains. However, the differences between call control and sensor or home networks are quite striking. Section 4 discusses the policy language variants for these domains in more detail.

For call control, there is a rich vocabulary of triggers, conditions and actions. This requires complex rules for which conditions and actions are permitted for each trigger. In contrast, sensor and home networks have a much simpler vocabulary. To some extent, this results from state in call control being held externally: a much richer interface is needed to operate on this. The triggers, conditions and actions for sensor and home networks can be made much simpler and more regular.

However, this is deceptive because much of the variety is moved out of trigger or action names and into their parameters. A typical call control trigger is *disconnect_outgoing*, meaning that the initiating user has hung up. This carries a single parameter: the user's address (or number). A typical sensor network trigger is *unit_in*, meaning there has been a significant state change in a system unit. This trigger has to carry many parameters: the unit class (e.g. gearbox temperature sensor), the unit identifier (e.g. 23), the unit values being reported (e.g. temperatures), and the period during which the values were recorded (e.g. the last five minutes).

3.5. *Defining Resolution Policies*

The policy system supports a wide range of resolution policies. For ACCENT, these were defined manually by a domain expert. Furthermore, they had to be written in XML and uploaded to the policy system.

In new work, support for resolution policies has been added to the policy wizard. This allows resolutions to be created and edited in a user-friendly manner. In addition, the definition of resolution policies has been semi-automated. (It cannot be fully automated as it requires human judgment.) The actions of regular policies are the triggers of resolution policies. In principle, every possible pair of actions must be considered for conflict. However the parameters of an action also have to be taken into account, e.g. *add_medium(audio)* and *add_medium(video)* are really distinct actions although they share a common action name.

Trigger	Condition Parameters	Actions
unit_in	time_interval, unit_class, unit_identifier, unit_value	log_event, restart_timer, send_message, start_timer, stop_timer, unit_out
timer_expiry	timer_identifier	

Figure 6. *Sensor Network Triggers, Condition Parameters and Actions*

Conflict detection is aided by associating actions (and relevant parameters) with their effects. For example, adding video to a call increases cost, requires more bandwidth, and invades privacy (e.g. the user's location is revealed). The effects of an action (with certain classes of parameter) are defined in the domain-specific ontology.

The RECAP tool (Rigorously Evaluated Conflicts Among Policies) has been developed to support conflict detection and resolution. The algorithm used by the policy server to detect conflicts is commutative (if P1 conflicts with P2, then P2 conflicts with P1), transitive (if P1 conflicts with P2, and P2 with P3, then P1 conflicts with P3), and associative (pairs of actions can be considered in any order for conflict). This means that RECAP has to consider only pairs of regular policy actions (plus relevant parameters). The tool user reviews these pairs, marking which are genuine conflicts and which are not. RECAP generates outline resolution policies from this information, and uploads them to the policy server. The policy wizard is then used to fill in the details of resolutions. When RECAP starts up, it reads existing resolution policies so that the user does not need to recreate them. This allows the tool user to modify what actions are considered to be in conflict, and how to deal with these situations.

4. Policies for Sensor and Home Care Networks

4.1. *Sensor Networks*

The policy system supports ECA policies (Event-Condition-Action). The triggers, condition parameters and actions used for sensor networks are shown in figure 6. The policy system receives triggers from and sends actions to 'units' in a broad sense. This includes physical plant (e.g. a temperature sensor or a generator set) as well as ancillary devices and software (e.g. an operator console or a log database). Most triggers and actions involve external units. However, some triggers (e.g. *timer_expiry*) and some actions (e.g. *log_event*) are handled internally by the policy server.

The policy system is designed to support the expression and realisation of high-level goals. A goal is a policy without a triggering event. Goal refinement will be realised through AI planning techniques, but has not yet been implemented. The following are examples of the kinds of goals that might be formulated for a wind farm: maximise power output; optimise turbine efficiency according to recent wind conditions; minimise blade stress as long as at least 1MW is generated per turbine.

Trigger	Condition Parameters	Actions
data_in, device_in, human_in	unit_class, unit_identifier, unit_value	data_out, device_out, human_out, log_event, restart_timer, send_message, start_timer, stop_timer
timer_expiry	timer_identifier	

Figure 7. *Home Care Triggers, Condition Parameters and Actions*

It is possible to write policies with arbitrarily complex combinations of triggers, conditions and actions. The following illustrates the kinds of policies that are supported for wind farms:

– if the temperature falls outside the range -15°C to +35°C, display a visual alert

– if a wind force above 8 is forecast, set turbines for 50% of rated power output

– if turbine speed exceeds 2 radians/sec, check blade stress every 5 minutes

– logs must be backed up every day at midnight

– an operator may shut down a turbine only after a system alert.

The policy server receives triggers from the processing nodes placed near turbines, and sends actions to these. To maximise flexibility of the system, the processing nodes support a simple form of ECA rules as well. These rules are called decisions to distinguish them from the policies and goals supported by the policy server. In fact there is a refinement hierarchy: goals to policies, and policies to decisions. A decision has a trigger (e.g. outside temperature), an optional condition (e.g. $<$ -15°C or $>$ 35°C), and an optional action (e.g. log the event). A *unit_out* action can be sent to a processing node to set up such a decision.

The policy server can refine policies into decisions by looking at the policy triggers and conditions. Consider the first policy above. This requires to be triggered by temperature sensors for a range of values. An implicit *unit_out* action is generated when such a policy is defined. This automated refinement of policies into decisions reduces the amount of explicit setup required.

4.2. *Home Care Networks*

The triggers, condition parameters and actions used for home care networks are shown in figure 7. As will be seen, the language is rather similar to that for sensor networks. Because a home network must manage several categories of 'units', a distinction is made between the triggers for data (e.g. databases, files), devices (e.g. sensors, actuators), and humans (e.g. speech input, haptic output).

As for sensor networks, the policy system supports goals and policies. However there is no equivalent of decisions as the low-level components are simple. The following are examples of the kinds of goals that might be formulated for home care: ensure that the user remains active; check that the user eats properly; the user must avoid too much stress.

The following illustrates the kinds of policies that are supported for home care:

– on weekdays at 7PM, set the video recorder to record channel 3 for one hour

– during householder absence, turn upstairs lights on and off randomly at night

– if the room temperature exceeds 28°C for 10 minutes, switch on the ceiling fan

– if user heart rate exceeds 180 beats/minute, advise immediate medical attention

– maintenance engineers may change any system parameter.

The policy server receives triggers from sensors and other software (e.g. a behaviour analyser or a database). It generates actions for actuators and other software (e.g. a speech synthesiser or a tone generator).

5. Conclusions

It has been seen how the ACCENT policy-based system for call control has been generalised and extended. Domain concepts have been defined in separate ontologies that extend the generic and wizard aspects of policy support. The main ACCENT component that needed generalisation was the policy wizard. This now uses the POPPET system to handle ontology queries about domain-specific aspects.

The policy language now deals more effectively with system state, allowing it to be recorded automatically. Timing issues are also handled. Support for resolution policies has been regularised, including semi-automated detection of policy conflicts through the RECAP tool. The new approach to policy support has been illustrated for sensor networks (in wind farms) and for home care networks (in telecare).

The main work that remains is implementation of goal refinement into policies. At present this is performed manually. Nonetheless, the enhanced approach to policy-based management has demonstrated its value in two novel applications.

Acknowledgements Gemma Campbell was supported by the UK Engineering and Physical Sciences Research Council under grant C014804. Feng Wang was supported by the Scottish Funding Council under grant HR04016. The authors thank their colleagues on the ACCENT, MATCH, and PROSEN projects for their collaboration.

6. References

Damianou N., Dulay N., Lupu E. C., Sloman M., Ponder: A Language Specifying Security and Management Policies for Distributed Systems, Technical Report n° 2000/1, Imperial College, London, UK, 2000.

Perry M., Dowdall A., Hone K. S., Multimodal and Ubiquitous Computing Systems: Supporting Independent-Living Older Users, *IEEE Trans. on Information Technology in Biomedicine*, vol. 8, n° 3, p. 258-270, September, 2004.

Turner K. J., Blair L., Policies and Conflicts in Call Control, *Computer Networks*, vol. 51, n° 2, p. 496-514, February, 2007.

Turner K. J., Reiff-Marganiec S., Blair L., Pang J., Gray T., Perry P., Ireland J., Policy Support for Call Control, *Computer Standards and Interfaces*, vol. 28, n° 6, p. 635-649, June, 2006.

World Wide Web Consortium, *Web Ontology Language (OWL) – Reference*, Version 1.0, World Wide Web Consortium, Geneva, Switzerland, February, 2004.