Feng Wang, Liam S. Docherty, Kenneth J. Turner, Mario Kolberg, Evan H. Magill. Services and Policies for Care at Home. In Jakob E. Bardram, Juan Carlos Chachques and Upkar Varshney, editors, Proc. *1st. International Conference on Pervasive Computing Technologies for Healthcare*, pages 7.1–7.10, Institution of Electrical and Electronic Engineers Press, New York, USA, November 2006

Services and Policies for Care At Home

Feng Wang, Liam S. Docherty, Kenneth J. Turner, Mario Kolberg, Evan H. Magill
Computing Science and Mathematics, University of Stirling, Stirling FK9 4LA, UK
Email fw, lsd, kjt, mko, ehm @cs.stir.ac.uk

Abstract—**It is argued that various factors including the increasingly ageing population will require more care services to be delivered to users in their own homes. Desirable characteristics of such services are outlined. The Open Services Gateway initiative has been adopted as a widely accepted framework that is particularly suitable for developing home care services. Service discovery in this context is enhanced through ontologies that achieve greater flexibility and precision in service description. A service ontology stack allows common concepts to be extended for new services. The architecture of a policy system for home care is explained. This is used for flexible creation and control of new services. The core policy language and its extension for home care are introduced, and illustrated through typical examples. Future extensions of the approach are discussed.**

Index Terms—**Home Care, Open Services Gateway initiative, Policy-Based Management, Service Discovery.**

I. Introduction and Motivation

A. Background

It is evident from Government and research statistics that the age distribution in many Western countries is shifting dramatically towards an older population. This factor alone will have an enormous impact on the demands for care services. Resource pressures and economic considerations are already driving many countries to look for new ways of delivering care services to greater numbers of people.

An emerging trend is increased use of network-based services for care delivery to the home. The last five years has seen major breakthroughs in bandwidth, availability and price of communications to and within the home. All of these make home care delivery a more feasible proposition. With the stabilisation of broadband and wireless, plus advances in assistive technologies, there is a real opportunity to create home care networks tailored to the needs of individuals.

Research indicates that the next generation of care products and services must provide personalised solutions across distributed networks, where care professionals or informal carers can monitor an individual's welfare and well-being in their own home. This will allow users to benefit from linked care services and household services, helping users to prolong an independent existence in their own homes.

The research described in this article aims to deliver care services locally to end users. It exploits a number of technologies of value in pervasive healthcare such as wireless networks, distributed communication, ontologies, policy-based management, and service-oriented architecture.

B. Research Challenges

Designing systems for care at home presents both technological and sociological challenges for researchers. The requirements for a home care system include the following aspects.

Easy Configuration The developers must allow for lack of technical knowledge among home users and the organisations that support them (e.g. health centres, social work departments). Automated configuration and remote management are needed to allow for easy installation and maintenance. Customisation and personalisation are essential to meet the needs of end users. Home visits are costly in staff time (especially in rural areas), and must be minimised

Easy Access The system needs to provide accessible and understandable user interfaces, no more complex than everyday domestic appliances. Conventional input devices (keyboard, mouse) are less suitable for non-technical users. A home care system should offer natural interfaces that exploit other modalities (e.g. speech, sound and touch). This is even more important where the end user has physical or mental impairments, whether through ageing or through illness.

Stakeholder Interests Many parties may be involved in home care: end users, health centres, community nurses, social workers, informal carers and family members. The interests of these stakeholders may conflict at a high level due to differing goals, or at a low level due to different technical implications. A home care system must provide mechanisms to support detection and resolution of such conflicts.

Technology Integration Many network technologies, device technologies and assistive technologies are available. An effective home care solution must be able to accommodate and integrate these. For example, there are many approaches to home networks such as ANSI X10 (powerline communication), the European Installation Bus, ETSI 300-220 (wireless communication), and UPnP (Universal Plug-and-Play).

C. Approach

Pervasive (or ubiquitous) computing has attracted considerable research and industrial interest (e.g. [1, 2, 3]). Many approaches to pervasive computing require specialised expertise for customisation or upgrading. Pervasive computing techniques have been applied in clinical settings (e.g. [4, 5]). There has been little research on pervasive computing for care at home, though some projects (e.g. [6]) are investigating sending the patient's medical data back to a care centre.

The authors believe that care at home requires much more than just relaying sensor data. A complete solution should be well integrated into the home and into the end user's lifestyle. Several commercial offerings support care at home. However such products follow proprietary rather than open architectures, and usually require specialised personnel to install and configure them.

The authors are part of a multi-partner team working on the MATCH project (Mobilising Advanced Technologies for Care at Home, www.match-project.org.uk). The mission of this project is to develop advanced software technologies in support of health and social care at home. MATCH is focusing on four technology areas of particular relevance to this goal: home network services, lifestyle monitoring, speech communication, and multimodal interfaces.

The authors have observed that approaches to smart houses and home care often focus on building smart devices. Such devices are specialised for particular functions, and do not lend themselves to well to other uses or to combination with other devices. The philosophy of the MATCH project is, as far as practicable, to use off-the-shelf, relatively dumb devices. This allows simple devices to be used to create smart services. Because this is achieved by software rather than hardware, new services and configurations are easily achieved.

This paper concentrates on one aspect of the MATCH project: service provision and management in home networks. OSGi (Open Service Gateway Initiative [7, 8]) has been selected as an industry-recognised approach to service provision. OSGi is neutral with respect to network technologies, and already supports a number of industry standards. Although originally conceived for service delivery to the home, OSGi has also been enthusiastically adopted for other applications such as automobile services.

Two key issues in home care networks are service discovery and policy-based management. Service discovery requires fine-grained description combined with flexibility. Policies are needed to let a variety of stakeholders state how they wish care services to be managed.

Section II introduces the technical background to the work reported here. Section III provides an overview of the home care system that has been designed. The approach to service discovery is discussed in section IV. Policy-based management of home care services is described in section V. Finally, section VI summarises initial experience of the approach and discusses future work.

II. BACKGROUND AND RELATED WORK

A. Positioning of The Research

The MATCH project is distinguished in a number of respects. The emphasis is on delivery of care services to the home. Social care plays a dominant role, though healthcare issues are also accommodated. This requires a wide range of situations in the home to be monitored and managed. For a similar reason, assistive technologies are also important. MATCH is interfacing to healthcare monitoring devices rather than developing them.

MATCH is focused on home care services. As a result, the MATCH approach needs to be seen in the context of home network architectures rather than healthcare information systems. The work on smart houses tends to concentrate on home automation (e.g. appliances, entertainment, security). Delivery of care is of lesser interest. Smart houses often emphasise device control, with service provision being secondary.

OSGi is ideal for MATCH as the approach is vendor-neutral, device independent, and focused on service provision. The designers of OSGi envisaged healthcare and self-care as important applications. Several projects have applied OSGi to healthcare, e.g. e-HealthCare (ehealth.sourceforge.net), Home HealthCare (www.ida.liu.se/%7estuha/anna-web/projects/HHC-overview.htm) and SAPHIRE [20]. However, healthcare is not the main focus of MATCH. As far as the authors are aware, its emphasis on social care using OSGi is unique.

[18] defines a widely used standard for exchange of healthcare information. This is supported by open-source projects like MIRTH (www.mirthproject.org). A middleware standard for healthcare information systems is defined in [19]. This addresses middleware for storage and retrieval of shared healthcare data. The Continua Health Alliance (www.continuaalliance.org) is particularly concerned to ensure interoperability of telecare solutions. A number of specifications have been developed to support healthcare applications of CORBA (Common Object Request Broker Architecture). However, all these approaches are exclusively for healthcare applications (typically electronic patient records), and so are of only peripheral relevance to MATCH.

Other differentiating factors in MATCH include the use of ontologies to enhance discovery of home care services, the use of policies to manage these services, and the fusion of multiple disciplines (e.g. activity monitoring, home networks, multimodal interfaces, speech technology, stakeholder analysis).

B. Open Services Gateway Initiative

OSGi defines a standardised, component-oriented execution environment for Java applications running on networked devices. An OSGi application (called a bundle) is a collection of software components rather than a monolithic chunk of code. The core of OSGi is a framework that manages the life-cycle of bundles, as well as providing important common services. Bundles can be installed, updated, started, stopped, and removed without stopping the platform. This is an important advantage for home care applications. Bundles can share code by exporting and importing packages. They can also use functionality provided by other bundles at run-time through a service registry.

On top of the framework, OSGi provides many standard services such as package administration, device access, protocol support (e.g. Jini, UPnP, X10), and miscellaneous capabilities such as an HTTP service and XML parsing. Remote management supports remote deployment, monitoring and maintenance of unattended devices. These facilities greatly simplify the process of developing solutions for home care.

C. Service Discovery

Service discovery within OSGi is limited to finding an implementation of a given service interface. LDAP

(Lightweight Directory Access Protocol) may optionally be used to discriminate among multiple implementations. In a pervasive environment, it is likely that context information will be more important for service discovery rather than properties of the service implementation. Contextual information includes the characteristics of a service with respect to its environment. The standard OSGi approach to service discovery uses simple key/value property pairs that are registered with the framework along with the service implementation. This method restricts the description to the service implementation, i.e. its technical characteristics. Ambiguities may arise due to inconsistency in concepts. For example, location may mean the URL from which a bundle was loaded or the physical location of a device controlled by the bundle. [9] highlights ambiguity as one of the potential problems when performing service discovery in OSGi.

Ambiguous descriptions could perhaps be avoided by using unique URIs to identify service properties. However, it is hard to make this work in practice. Even in natural language, similar terms can be used with the same meaning, and the same term can be used with different connotations. It is therefore problematic to make automated service discovery work reliably.

Semantic descriptions are popular for web services. Semantically-based description languages for web services include SWSF (Semantic Web Services Framework) and OWL-S (Web Ontology Language with semantic markup for services). These aim to support standard ways of describing services and interacting with them, though at present they are still under development. Protocols such as UPnP contain service and device descriptions in an XML format. However unlike OWL-S, these are fixed and cannot evolve as new information is added.

Ontologies have become popular as the basis for consistent communication among agents and web services. RDF (Resource Description Framework) and its associated RDFS schema can be used to ascribe meaning to data. DAML+OIL (DARPA Agent Markup Language plus Ontology Interchange Language) and OWL allow more expressiveness and functionality. The OWL-DL description logic supports logical inference and reasoning, allowing inconsistencies to be discovered and new information to be inferred.

[10] describes a middleware architecture for enhancing service descriptions in an OSGi framework, using XML descriptions of service behaviour. This approach concentrates more on how a service behaves, rather than on what the service offers. The solution in the current paper gives a fuller description of services, including a complete description of their effects.

The need for context awareness within a networked environment is emerging as a relatively new area of research. Some efforts (e.g. [1, 11, 12]) use real-time context information to aid in system decision-making. Other work (e.g. [13, 14]) uses ontologies to describe context terms within the environment. The authors' approach differs in using service-oriented context information, instead of just application or system aspects.

D. Policy-Based Management

Policy-based management has been applied to a number of areas, including the management of networks/distributed systems [15, 16] and system configuration [9]. However the target users of most policy systems are IT professionals who can be expected to have specialised technical knowledge.

One exception is the policy system developed by the ACCENT project [17], which was developed for end users to manage their calls. Call control policies are in ECA form (Event-Condition-Action). When the given combination of triggers occurs under defined conditions, the specified actions are performed. Conditions make use of so-called environment variables. These may be established by a trigger, by a context system (e.g. user role, capability), or by the policy server (e.g. current time, policy preference). A policy wizard allows policies to be defined by non-technical users using stylised natural language.

The ACCENT policy system is designed to be neutral with respect to triggers, conditions and actions. This reflects the fact that any policy language it supports has a core structure that is then specialised for different applications. The system is also designed to be generic in that it deals with 'servers' via a defined API. In call control, the policy system is interfaced to communications servers that handle with the underlying networks.

Since the authors had direct access to the ACCENT system, it was of interest to see if this could be adapted for home care policies. This has proven to be practicable, though not entirely straightforward. Triggers may arise from a wide variety of sensors in the home, while actions may apply to many different kinds of actuators. In call control, state is implicit (being managed by a communications server). For home care, the policy system has to be aware of the system configuration and state. Call control also does not involve correlation across different calls, whereas the activities of home devices typically have to be coordinated.

III. HOME CARE SYSTEM OVERVIEW

A high-level view of the system architecture is shown in figure 1. Inputs come from sensors in a broad sense: physical devices as well as logical information sources and user inputs. Outputs go to actuators that may similarly be physical, logical or user-oriented. Home services and device control are located in what OSGi terms a residential gateway. This is linked to the outside world, typically via a broadband connection to the Internet. However, for some purposes it is convenient to link to a cellular network – either directly or via an Internet gateway. This reflects the strong interest in using mobile devices for symptom monitoring. These typically make use of digital phone systems such as GPRS (General Packet Radio Service) or UMTS (Universal Mobile Telecommunications System).

Subject to user agreement, information from the home can be sent to a variety of care providers: health centres, social work departments, and informal carers. Information can also flow from care providers to the home. However, many other Internet-based sources of information can be used in the home (e.g. weather reports about severe conditions).

An obvious concern with any networked system is security. Unauthorised access must be prevented to the home system and the data it collects. OSGi provides a general security framework that is being extended to meet the needs of home

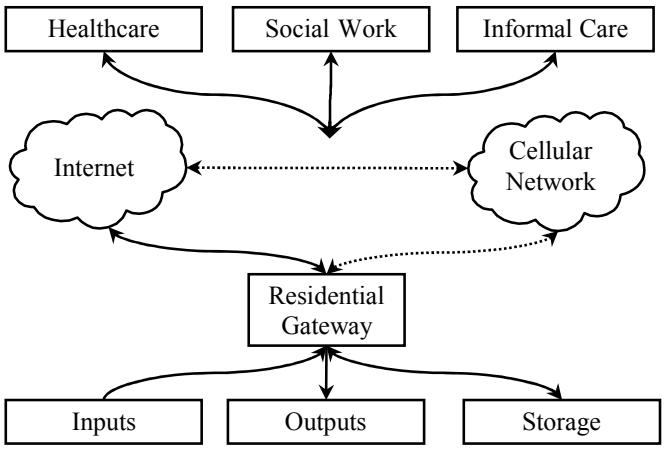

Figure 1. High-Level Architecture for Home Care

care. This is coupled with security policies that define who may configure devices and services, who may use what data for what purpose, etc.

The home care devices supported include general ones (e.g. flood detectors, movement sensors) as well as specialised assistive technology devices (e.g. epilepsy monitors, fall detectors). The MATCH project is also investigating a range of modalities such as speech communication, non-speech audio, haptic (touch) devices, and analysis of tremor and gait. Speech has obvious applications for input (voice commands) and for output (synthesised reminders and advice). However, speech communication for care at home presents particular challenges. Due to age or illness, users may speak unclearly or may have hearing limitations. Speech recognition in a home environment is also difficult. Non-speech interfaces must also be designed to meet the needs of target users.

With user consent, data about health and lifestyle can be captured and stored within the home. Policies control what information may be recorded, processed, and released to approved third parties. Lifestyle monitoring can make use of such information to detect significant variations from the norm, and to determine long-term trends such as deterioration in user capabilities. Research in activity monitoring and lifestyle modelling has shown exciting potential. It enhances the understanding of activities that affect quality of life, and how changes in these activities influence the independence of older or infirm people. The results from lifestyle monitoring allow user needs and well-being to be monitored, so that external help can be sought in good time. Activity tracking and modelling are being extended to give users and their carers more confidence while they are out and about. This helps users to remain mobile and active.

The home care system is not closed. Instead, it is connected to external care parties through the Internet or other data networks. Call centres currently use a restricted range of services such as telephony. It is planned to extend the range of services to include aspects such as remote health monitoring and telecare, remote system management, and remote configuration. An advantage of the approach is that the configuration of home devices need not be fixed in purpose. The project is developing a goal-directed, top-down approach that configures devices and their combination to achieve specific purposes. For example, a movement sensor may be used to support many goals such as security, monitoring room usage, noting household entry/exit, controlling lighting, and activating nearby appliances.

IV. SERVICE DISCOVERY

A. Ontology-Based Service Description and Discovery

Home care relies on a variety of services to control devices, provide care, and support management using policies. There is a need for service discovery, particularly in support of automated configuration and goal-directed configuration. As noted earlier, conventional OSGi service discovery suffers from problems of inconsistency and ambiguity in service description. The authors' preferred approach is to use ontologies that provide a uniform and semantically-based way of describing services and the concepts they need. Ontologies offer the consistency and reusability required. Ontology-based service descriptions are rich enough to capture context information about a service, allowing for more expressive service description and discovery.

Use of ontologies ensures consistency among services developed by different parties. An ontology describes the concepts and their relationships within a given domain – here, home care. Developers can extend this ontology when creating new services. For example, a generic ontology can define the characteristics of location. This can then be specialised for new purposes, e.g. to talk about absolute, relative or network location.

OWL (and its variants) are appropriate for defining ontologies in home care. Apart from expressiveness, OWL is endorsed by the W3C and is well supported by tools.

It is unlikely that a single ontology would be sufficient in capturing all concepts and properties in a domain that continues to evolve. This is an issue in home care, where the configuration of devices and services needs to adapt to the changing needs of users. Ontologies allow concepts and properties to be readily modified to match an evolving domain.

In the area of service description, suppose a new lighting service were developed after an initial service description ontology had been created. A developer could use concepts from the initial ontology as well as concepts created specifically for the new service. Languages like OWL allow a base ontology to be imported and extended. Home care is a rapidly evolving application domain, with new devices and services constantly under development As an understanding of home care evolves, common concepts can be moved from more specialised ontologies into the shared base.

Currently, service discovery in OSGi is limited to queries about service properties. An ontology has been developed for home care services, capturing information about technical aspects of a service as well as the context of its use. This

supports queries where the effect of a service rather than its implementation are important. As an example, suppose that an application wished to control temperature in a room with windows. The issue here is that room temperature is affected by more than just heating or air conditioning. Windows let in sunshine, which heats a room. Windows allow heat to escape by radiation and conduction. Windows can also be opened to equalise the internal and external temperatures. A heating application therefore needs to be able to discover which rooms in the house have windows, and which of these rooms have temperature control services.

In a pervasive environment, the authors believe that current OSGi service discovery mechanisms are too limited. Context-dependent queries founded on ontology-based service descriptions provide a more robust and expressive approach.

B. Service Properties

Using the results of service discovery requires an understanding of how to use the returned services and what their effect is. A basic approach provides only information about the interface to a services – its syntax. For example, it may be discovered how to call a temperature control service, but other effects of this may not be known. Thus, heating one room may be countered by trying to cool an adjacent room. A cost-effective way of cooling a room in summer may be to open the windows, but this could compromise security.

Such side effects are described using an ontology for service description. This results in a fuller description of services, including their primary functions as well as secondary or side effects. This allows the requester to be more precise about the home care service required. More interestingly, it supports automated detection of undesirable conflicts among services. In a home setting, it is likely that services will be developed by independent vendors. This is already the case, with current offerings including services for security, climate control, energy monitoring, and entertainment. As already mentioned with temperature control vs. security, independent services may inadvertently interfere with each other. This is a well-known issue in telephony, where it is called the feature interaction problem. In policy-based management, such interference is referred to as policy conflict.

OSGi provides a security manager that restricts access to implementations of a particular interface. While appropriate for certain service types such as home security, this can be too coarse-grained for some services. Authorisation and access are therefore defined as part of the generic service ontology. Service discovery and usage can then be controlled. An authorisation hierarchy (lattice) allows finer-grained management. For example, services with limited privileges are not allowed to discover or to use more strictly controlled services. Security is enforced by the service implementation rather than its interface.

C. Ontologies for Home Care Services

The authors have developed a number of OWL ontologies to support more precise and expressive service discovery. These ontologies represent the essential characteristics of home care services deployed using OSGi. The ontologies are intentionally abstract. For example, the base ontology describes general concepts like vendor, location, service type and environment. These are then defined in more detail by ontologies specific to these concepts. Thus, the generic notion of location is extended to allow relationships among rooms to be expressed (such as 'next to'). Similarly, the generic notion of service type is extended to talk about service authorisation and secondary effects (linked to effects in the environment ontology). The approach thus gives both an abstract and a detailed view of the same ideas.

The collection of ontologies is referred to as a service ontology stack, as illustrated in figure 2. Using these ontologies, developers can create their own service descriptions – by customising one of the service type ontologies provided, or by using concepts from lower-level ontologies. This allows home care services to be given more expressive descriptions.

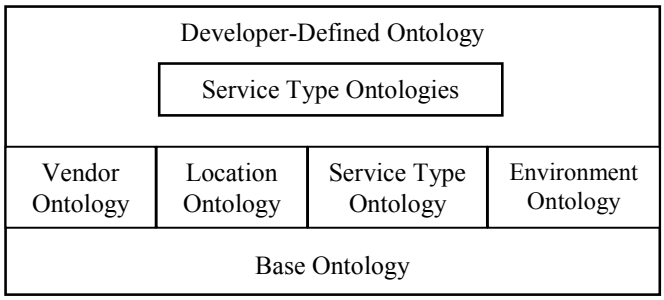

Figure 2. Service Ontology Stack

However, the service descriptions cannot be used directly by OSGi. A semantic service discovery bundle has been created using the Jena2 Semantic Web Toolkit (jena.sourceforge.net). Jena stores and reasons about service ontology descriptions, and supports queries about services. This bundle works in conjunction with the current OSGi service registry, as shown in figure 3.

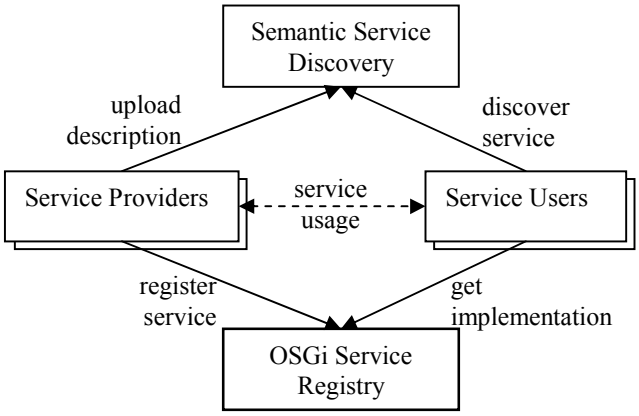

Figure 3. Semantic Service Discovery

A requester can submit queries to the semantic service discovery bundle. This returns results as fully-qualified class names for matching service instances within the framework. A requester may have to include authorisation information to discover services that are strictly controlled. On receiving fully

qualified class names, the requester makes a selection and performs normal OSGi service discovery, supplying the service name and class name in an LDAP query. This obtains the desired service implementation. If a service is removed from the OSGi framework, the semantic service discovery bundle is notified so that it can remove the relevant ontology description from its registry.

The semantic service discovery bundle also provides methods for a service to modify its ontology description, allowing context information to be changed dynamically. Using the fully qualified class name as a unique identifier, semantic service discovery updates obsolete information.

V. POLICY-BASED SYSTEM MANAGEMENT

A. Policy System Design

To support better flexibility and control, home care services are managed by a policy system. This allows users to formulate policies for how they wish the care system to behave. In fact, a user in this context means a range of stakeholders including end users, care providers and informal carers. The policy system is part of the residential gateway shown in figure 1. It is integrated into the OSGi framework, interacting with it to manage devices and services.

Figure 4 shows the architecture of the policy system. The inputs and outputs are as discussed for figure 1; they include physical devices, but also interactions with users and services.

Policies are expressed in a language called APPEL (ACCENT Project Policy Environment/Language). This provides a set of core constructs. APPEL is then specialised for each application domain by defining the triggers, conditions and actions in that domain.

The policy store holds policies internally as XML documents that conform to the APPEL schema. However, the policy system must support non-technical users. The policy wizard is therefore vital as an easy-to-use way of creating, modifying and deleting policies. The wizard allows policies to be formulated and edited using stylised natural language. The wizard is web-based, partly because this is now familiar to many users, and partly because policies can then be edited remotely. However, it is recognised that a textual web interface may not be suitable for all users. Other approaches being considered for the policy wizard include a graphical interface and a speech-based one.

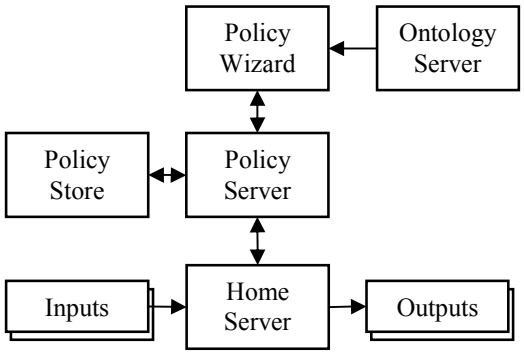

Figure 4. Policy System Architecture

Since the wizard must allow policy definition in a variety of applications, domain-specific knowledge must be factored out. This is achieved by a system called POPPET (Policy Ontology-Parsing Program – Extensible Translation). This uses ontologies, but for a very different reason from their use in service discovery. In this case, ontologies are used to capture the concepts and relationships for policies about home care. The core APPEL language has its own ontology that is extended by an ontology for the wizard. Finally, a domain-specific ontology sits on top of this. The wizard uses POPPET to discover the specific kinds of policies that may be formulated in a domain.

The policy wizard offers a number of simplifications for ease of use. In fact a user need not define policies in order to benefit from them; the policy system administrator may define policies on behalf of a group of users. The administrator also manages user profiles that include user expertise. This affects how much of the policy language the user sees, a novice user being presented with just the basics. A range of template policies is provided, allowing the user to select pre-defined policies rather than having to create them from scratch. These policies may have a few parameters that the user is expected to complete (e.g. a name or telephone number). Alternatively, policies may be parameterised by variables that are instantiated at run time. These variables are defined separately from policies, either manually through the wizard or automatically through the system.

Apart from policies, the policy store holds user profiles, the system configuration, and the system state. Configuration information is necessary so that policies can refer symbolically to things like ‘the front door’, ‘the community nurse’ or ‘the security service’. System state is needed so that policies can be interpreted dynamically in context. Status variables keep track of device and service state, such as whether a bed is occupied or whether a service is online.

The home server provides the communication mechanism between home devices and the policy system. Components communicate with the policy system by sending or receiving events. These contain a trigger name and a set of attributes. For example, a UPnP passive infrared sensor sends out movement events with attributes identifying the source of the event. An X10 dimmer module accepts actions with attributes such as the house code, the unit code and the dimming level.

A policy system input (typically a sensor) causes the home server to report a triggering event. The policy server retrieves the policies associated with this trigger. Policies are selected only if they meet certain conditions. These include what the policy applies to, the period of validity of a policy, and what profile the policy belongs to. Policies may be grouped into categories such as ‘at night’ or ‘on holiday’, allowing sets of policies to be activated easily. Variables in the selected policies are retrieved from the policy store; this includes configuration variables and status variables. Policy conditions are evaluated to determine which policies apply. All enabled policies then dictate the actions to be performed. In the absence of conflict, these actions are sent to the home server. In turn, this causes various outputs (typically to actuators).

The policy server also deals with conflicts among policies due to contradictory actions. Opposing actions are examples of simple conflicts (e.g. 'open the door' vs. 'do not open the door'). However, conflict detection can be much more sophisticated. For example, 'open a window' conflicts with 'heat the room' during winter. Conflict handling is externalised by design. That is, the policy server does not have built-in rules to detect and resolve conflicts. Instead, conflicts are *defined* by higher-level resolution policies. These take conflicting policy actions as triggers that lead to resolution. A generic resolution chooses among the conflicting actions according to some high-level criterion (e.g. a carer's policy is preferred to an end user's policy). A specific resolution dictates explicitly what actions are taken to handle the conflict. This might be an automatic response or might involve a person.

B. The Core Policy Language

The core policy language builds on previous work for call control [17]. The core is defined by a domain-independent XML schema. Specific triggers, conditions and actions are not specified by the core language. Instead, these are added as extensions in domain-specific schemas. The core language offers a range of constructs including the following.

Parameterised Policies The language supports parameterised policies that are instantiated with particular values for policy variables. This is useful, for example, in template policies for non-technical users. Policy variables are also used to give symbolic names to devices that otherwise have configuration-specific addresses. The system configuration held in the policy store performs this mapping.

Domains Individual policies apply to whoever defines them. However policies may be defined for domains, i.e. groups of users such as those in a specific nursing home or in a category like community nurse.

Modality A policy may define a preference (e.g. *must*, *should* or *prefer*, along with negative versions of these). This information implies a weight for the policy that is taken into account if conflicts have to be resolved.

Rule Combinations Policies comprise rules that may be combined in various ways, e.g. subject to some condition, tried in sequence, or executed in parallel.

Rules An optional trigger specifies the external event(s) that may activate a policy. Triggers may be combined with *and/or*. An optional condition defines the circumstances in which a policy may apply. Conditions rely on information established by triggers, such the device, the system state, or the time of triggering. Conditions may be combined with boolean operators. An action gives the effect of a rule. Actions may be combined with various operators such as *and/or*, though more sophisticated combinations are possible such as parallel execution.

C. The Policy Language for Home Care

The core policy language has been specialised for the triggers, conditions and actions required for home care.

1) Triggers

Sensor inputs are handled by a generic device trigger. It would have been possible to define a range of device-specific triggers. However, this would have considerably complicated the policy language. It would also have biased it towards a particular set of device types. A study was made of the kinds of devices that are useful in home care This resulted in a taxonomy of such devices, showing that there is considerable commonality among them from the event point of view.

A single *device_in* trigger is therefore used, with an argument that indicates the device (e.g. *front_door*) and one that indicates the status (e.g. *open*). It is preferable to use symbolic names for devices, mapped to addresses through the system configuration; however, well-known addresses could be used (e.g. for the residential gateway itself). Device inputs establish other values that can be used in policy conditions.

In call control, policies mainly depend on the attributes of triggers. This is sometimes sufficient for home care applications. For example, a policy may just need to know that the front door has been opened. However, more typically a home care policy needs to know the overall system state. For example, movement down the path after the front door is locked suggests that someone has just left the house.

Unlike call control, home care policies are often strongly influenced by timing. For example, it may be necessary to issue a warning that the cooker is still on if the user leaves the kitchen for some period of time. This requires a timer trigger that needs to know whether the cooker is on or off. The cooker status established by other triggers is therefore held in the policy store.

Time-based triggers have been added to the policy language and server. An *at* trigger happens at a certain point in time. An *every* trigger occurs at a fixed time each day. An *after_event* trigger is fired if a specified event happens after a given period of time, while a *no_event* trigger occurs if this does not happen. A *repeated* trigger means a given event re-occurs a given number of times in some period.

Other triggers are used for particular services. For example, speech input uses a *recognise* trigger.

2) Conditions

A policy may have a single condition or a combination of these. A single condition has a parameter (a value established by a trigger), an operator (that performs the check) and a value (established by the trigger or known from the system state). As noted earlier, the status of system entities is maintained (e.g. a door is open or a user is watching television). When the system status is updated, the entity, its new state and the time of the change are all recorded. The term 'entity' is used broadly here to include devices, services and people.

The core policy language defines a limited range of condition operators that are interpreted according to the kind of value being checked. Apart from comparison operators like = and >=, there are *in* and *out* operators to check for inclusion or exclusion. The latter are particularly useful for ranges or sets (e.g. a time or location check).

3) Actions

Actuators may be of many different kinds, controlled by different protocols. For example a lamp may be switched on using X10, or a video recorder may be managed through UPnP. It is a design goal that the policy system be independent of particular devices and protocols; it is up to OSGi to handle

these differences. For generality, a single *device_out* action is therefore used. This carries arguments that give the device identifier, command name and parameter names/values. As for inputs, outputs normally use symbolic names for devices.

Other actions are used for particular services. For example, speech output uses a *speak* action.

D. Policy Examples for Home Care

To make the policy language concrete, the following examples show how it can be used to control a variety of devices in support of home care. The policy language is more widely used, e.g. for managing services and for defining higher-level goals. The example policies are shown in XML form, but omitting some 'red tape' such as the obvious closing tags.

Although the policy system is focused on supporting care at home, the experience of social work departments is that it is vital to provide end users with a comfortable living environment. It is therefore often important to protect the home as well as its occupants. To this extent, some of the policy examples here would be useful in any home.

1) Medical Care Policies

Older people, especially those with dementia, are prone to waking in the middle of the night and leaving the house. Figure 5 shows how the user can be advised to return to bed using synthesised speech (say, of a family member).

Although policies have an XML form internally, users do not see this. The example below is rendered in stylised natural language as follows: when the front door opens; if the time is between 11PM and 7AM, and the main bed is unoccupied; speak a message in the hallway 'it is night time, go back to bed'.

This example illustrates several points about the policy language. Triggers and actions have fixed values such as *device_in* or *speak*. For proper validation, arguments must be carried as XML attributes (*arg1*, *arg2*, etc.). Policy variables are distinguished from literal values by a colon prefix, e.g. *:front_door* or *:master_bed*. Combinators like *and* are binary, so a compound condition must be constructed from pairs of sub-conditions.

```
<policy_rule>
 <trigger arg1=":front_door" arg2="open">
  device_in(arg1,arg2)
 <conditions>
  <and/>
  <condition>
   <parameter>time
   <operator>in
   <value>23:00:00..07:00:00
  <condition>
   <parameter>:master_bed
   <operator>eq
   <value>unoccupied
 <action arg1=":hallway"
          arg2="it is night time, go back to bed">
  speak(arg1,arg2)
```

Figure 5. Night Wandering Policy

2) Home Appliance Policies

Home appliances can be controlled by a variety of means. For example, X10 allows standard electrical appliances such as cookers or washing machines to be switched on/off through mains wiring. UPnP allows home entertainment devices to be controlled via an in-house network. Policies can dictate how these appliances are used. Such policies can be conveniences for the user, saving them from having to remember to do things. However, policies can also be used to avoid potentially dangerous situations. For example, if a user needs supervision while cooking then the cooker should not turn on unless a helper is present.

The example in figure 6 describes a situation in which someone has a home help call every evening to cook dinner. To save time, a policy turns on the oven at 7PM and sets it for 200°C. This ensures it is warmed up for the home help calling.

```
<policy_rule>
 <trigger arg1="19:00:00">every(arg1)
 <action arg1=":oven" arg2="on"
          arg3="temperature=200">
  device_out(arg1,arg2,arg3)
```

Figure 6. Oven Warming Policy

3) Security Policies

Figure 7 shows a policy that turns on an alarm for 20 minutes if there is movement in the hallway or lounge between midnight and 6AM.

```
<policy_rule>
 <triggers>
  <or/>
  <trigger arg1=":hallway" arg2="movement">
   device_in(arg1,arg2)
  <trigger arg1=":lounge" arg2="movement">
   device_in(arg1,arg2)
 <condition>
  <parameter>time
  <operator>in
  <value>24:00:00..06:00:00
 <action arg1=":alarm" arg2="on" arg3="period=20">
  device_out(arg1,arg2,arg3)
```

Figure 7. Night-Time Movement Policy

4) Entertainment and Communications

Home care policies are also valuable in supporting everyday activities. For example, someone with limited dexterity, impaired dexterity or weakened cognitive abilities may find it difficult to operate appliances like washing machines, video recorders or telephones.

As an entertainment example, the policy in figure 8 states that the user wishes to record channel 3 for one hour, starting at 9PM on 31st May 2006. (Date and time formats follow XML schema conventions.)

```
<policy_rule>
  <trigger arg1="2006-05-31T21:00:00">at(arg1)
  <action arg1=":video_recorder" arg2="record"
          arg3="channel=3,period=01:00:00">
    device_out(arg1,arg2,arg3)
```

Figure 8. Video Recording Policy

VI. EVALUATION

A. Current Status

Knopflerfish [12] has been used as an open-source implementation of OSGi Release 4. Jena and Protégé (protege.stanford.edu) have been used to create ontologies in support of service discovery and policies. The existing ACCENT policy system has been re-used, though it required packaging the policy server as an OSGi bundle. An event service is supported directly by OSGi version 4, significantly reducing the development effort. Using OSGi as the foundation for home care services has been a positive experience, with particular benefits gained from the clean separation of services, devices and protocols.

The semantic service discovery bundle has been developed and deployed within OSGi, allowing experimentation with complex service queries. An initial service ontology stack has been used to create descriptions of home care services. Using a set of basic service type ontologies, evaluation of semantic service discovery has demonstrated its expressiveness. Currently, queries may be single, multiple (fixed-content) or complex combinations.

Practical evaluation of the approach has been conducted in a laboratory that serves as a home environment. Various wireless sensors are used to detect movement, flooding, smoke, bed occupancy, and door opening. A standard wireless receiver has been interfaced to a PC using a USB adapter. An OSGi bundle was written to read the wireless sensor inputs. For output, OSGi bundles have been written to control X10 appliances (on, off, dim), to control UPnP alarm devices, to interface with SIP (Session Initiation Protocol, used for Internet telephony), and to send text messages for mobile telephones.

The policy examples given earlier have been tried in practice, though some devices had to be simulated by software in the meantime. It has been found that policies are conducive to flexibility and ease of change. The approach of other developers typically requires programming a PIC (Programmable Interface Controller), FPGA (Field Programmable Gate Array), or custom code for a service. In contrast, a wide variety of policy-based services can be created without having to write any software. For development purposes, policies can be entered in XML format. However, the wizard is the primary means for ordinary users to define policies.

B. Future Work

The efficiency and scalability of semantic service discovery will be investigated as descriptions become more comprehensive. An issue here will be maintaining only the most relevant or current information in the ontological knowledge base. The greater precision of service description will be exploited to detect potential interference and how to handle this. It is planned to extend the framework with a service that identifies and resolves interference before services are instantiated.

The conflict detection and resolution techniques developed by ACCENT will be of value in this role and for handling policy conflict. This will be important in enhancing the robustness of home care services. Conflict handling allows interference to be detected statically (at definition time) or dynamically (during execution). So far, most effort has gone into dynamic handling as this is the more challenging issue. However, it is believed that similar ideas can be extended to static handling as well.

Of necessity, resolution requires ignoring certain policy actions. If this is required, the policy system must be able to explain why this happened. The use of synthesised speech is an obvious possibility here. Different schemes will also be investigated for achieving the best possible resolution. For example, denying an action may be associated with a certain penalty. The policy server also allows for pre-negotiation and post-negotiation (negotiating a conflict before or after committing to particular policy actions).

Automated configuration and goal-directed configuration will be investigated. The former is desirable for simple on-site installation. The latter will be useful for refining high-level goals into more operational objectives. It is anticipated that AI planning techniques will be used for this aspect. The composition of OSGi services into new services will also be studied.

Remote management and security are important future goals. The corresponding capabilities of OSGi will be enhanced to make them suitable for home care. Secure definition of policies is already supported. Management policies will be defined to allow control of services and access to data. It will be crucial to manage what data may be used for what purposes. This includes whether data may be exported out of the home, and how it may be analysed. Health and lifestyle data must obviously be kept confidential, and be processed in an authorised manner by identified individuals.

ACKNOWLEDGEMENTS

The authors thank their colleagues on the MATCH project at Stirling (Louise Bellin and Julia Clark) for their advice. The authors are also grateful to their collaborators at the Universities of Dundee, Edinburgh and Glasgow who are creating home care facilities that complement the work reported here. Numerous external partners in MATCH have been very helpful in providing insights into health and social care, as well as into device technologies. Gemma Campbell at Stirling was responsible for developing the POPPET system.

REFERENCES

[1] A Middleware Infrastructure to Enable Active Spaces. M. Román, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H.

Campbell and K. Nahrstedt, *IEEE Pervasive Computing*, 1(4):74–83, Oct.–Dec 2002.
[2] D. Garlan, D. Siewiorek, A. Smailagic, P. Steenkiste. Project Aura: Toward Distraction-Free Pervasive Computing, *IEEE Pervasive Computing*,1(2):22–31, Apr.–Jun. 2002.
[3] A. Helal, W. Mann, H. Elzabadani, J. King, Y. Kaddourah and E. Jansen. Gator Tech Smart House: A Programmable Pervasive Space, *IEEE Computer*, 38(3):50–60, March 2005.
[4] J. E. Bardram. Applications of ContextAware Computing in Hospital Work – Examples and Design Principles, *Proc. ACM SAC '04*, Nicosia, Cyprus, Mar. 2004.
[5] M. Drugge, J. Hallberg, P. Parnes, and K.Synnes. Wearable Systems in Nursing Home Care: Prototyping Experience, *IEEE Pervasive Computing*, 5(1):86–91, Jan.–Mar. 2006.
[6] J. E. Bardram. The Personal Medical Unit – A Ubiquitous Computing Infrastructure for Personal Pervasive Healthcare. In T. Adlam, H. Wactlar, and I. Korhonen, eds., *Proc. 3rd. Ubiquitous Computing for Pervasive Healthcare Applications*, Nottingham, UK, Sep. 2004.
[7] D. Marples and P. Kriens. The Open Services Gateway Initiative: An Introductory Review, *IEEE Communications Magazine*, 39(12):110–114, Dec. 2001.
[8] C. Lee, D. Nordstedt and S. Helal. Enabling Smart Spaces with OSGi, *IEEE Pervasive Computing*, 2(3):89–94, Jul.–Sep. 2003.
[9] M. Burgess. A Site Configuration Engine, *USENIX Computing Systems*, 8(3):309–337, 1995.
[10] Y. Qin, H. Hao, Li Jun, G. Jidong and L. Jian. An Approach to ensure Service Behaviour Consistency in OSGi. *Proc. 12th Asia-Pacific Software Engineering Conference*, pp. 185–192, 2005.
[11] A. Ranganathan, J. Al-Muhtadi and R. H. Campbell. Reasoning about Uncertain Contexts in Pervasive Computing Environments, *IEEE Pervasive Computing*, 3(2):62–70, Apr. 2004.
[12] X. Wang, J. S. Dong, C. Y. Chin, S. R. Hettiarachchi and D. Zhang. Semantic Space: An Infrastructure for Smart Spaces, *IEEE Pervasive Computing*, 3(3):32–39, Jul. 2004.
[13] E. Christopoulou, C. Goumopoulos, I. Zaharakis and A. Kameas. An Ontology-based Conceptual Model for Composing Context-Aware Applications, *Proc. 6th. International Conference on Ubiquitous Computing*, Nottingham, UK, 2004.
[14] X. H. Wang, T. Gu, D. Q. Zhang and H. K. Pung. Ontology Based Context Modelling and Reasoning using OWL, *Proc. IEEE International Conference on Pervasive Computing and Communication*, Orlando FL, USA, Mar. 2004
[15] J. Lobo, R. Bhatia and S. Jaqvi. A Policy Description Language. *Proc. American Association for Artificial Intelligence*, Orlando FL, USA, Jul. 1999.
[16] N. Damianou, N. Dulay, E. Lupu and M. Sloman. Ponder: A Language specifying Security and Management Policies for Distributed Systems, Technical Report, Imperial College, London, UK, 2000.
[17] K. J. Turner, S. Reiff-Marganiec, L. Blair, J. Pang, T. Gray, P. Perry and J. Ireland. Policy Support for Call Control, *Computer Standards and Interfaces*, 28(6):635–649, Jun. 2006.
[18] ANSI. *Application Protocol for Electronic Data Exchange in Healthcare Environments*, ANSI/HL7 V2.5, American National Standards Institute, Washington DC, USA, 2003.
[19] CEN. *Medical Informatics – Healthcare Information Systems Architecture - Part 1: Healthcare Middleware Layer*, ENV 12967-1, European Committee for Standardization, Delft, Netherlands, 1998.
[20] A. Hein, O. Nee, D. Willemsen, T. Scheffold, A. Dogac and G. B. Laleci. Intelligent Healthcare Monitoring based on Semantic Interoperability Platform - The Homecare Scenario, *Proc. 1st European Conference on eHealth*, Fribourg, Switzerland, Oct. 2006.