

Managing Home Care Networks

Kenneth J. Turner Liam S. Docherty Feng Wang Gavin A. Campbell

Computing Science and Mathematics, University of Stirling
Stirling, Scotland, FK9 4LA, UK
{kjt,lsd,fw,gca}@cs.stir.ac.uk

Abstract

Home care networks are a new development for automated support of care at home. To address the challenges of home care, the paper describes a component-based architecture developed by the MATCH project (Mobilising Advanced Technologies for Care at Home). Two key components are discussed for managing home care networks. A Service Registry supports generic and extensible registration of services, components, resources and devices in the home. The registry uses ontologies for semantically-based description and discovery. A Policy System automates support of how a home network should deliver care. The views of stakeholders in home care are represented as goals and policies. These are defined in a user-friendly manner and are applied at run-time. Conflicts among goals and policies are automatically detected and resolved.

1. Introduction

1.1. The Need for Home Care

The world population is gradually ageing, with the percentage of older people (over 65) expected to rise by 2050 to 19.3% world-wide and much higher in some countries (e.g. 36.4% in Japan). There is therefore a strong need to help older people prolong independent living in their own homes. Others could also gain from being able to stay at home (e.g. those with physical or mental disabilities, or those with long-term medical conditions).

A strong element of human involvement must remain in care delivery. However, appropriate technologies can help to support someone receiving care at home. These technologies can provide the user with advice, identify trends or anomalies that may require intervention, monitor potentially undesirable situations, provide reassurance to family members and informal carers, and relieve professional carers of routine low-level tasks. Technological support can also bring economic benefits. For example, the cost of look-

ing after someone in a care home can be up to £24k per year (30k Euros, 44k US dollars). The number of care homes will also become increasingly inadequate.

Technologies in support of home care have various labels. ‘Assisted Living’ means devices and services that help to prolong independent living at home. ‘Assistive Technology’ refers to devices that help with daily living. ‘Telecare’ refers to localised devices and services that aid daily living, but with a remote link to support services such as a call centre. ‘Telehealth’ refers to remote monitoring, consultation and diagnosis of health issues. ‘Smart Homes’ focus more on home automation and monitoring. ‘Home Care System’ is used in this paper to mean the devices and services deployed in the home to support care delivery.

Home care involves an interrelated set of services for social or health care (or both). Home care systems have previously used simple devices such as pendant or flood alarms. However, they are rapidly becoming multi-user, distributed, integrated and adaptable. Each of these aspects makes it difficult to develop an autonomous home care system.

1.2. The MATCH Project in Context

The MATCH project (Mobilising Advanced Technologies for Care at Home, www.match-project.org.uk) is supported by the Scottish Funding Council (Nov. 2005–Oct. 2009). The Universities of Stirling, Dundee, Edinburgh and Glasgow are working on MATCH to develop a research infrastructure for supporting care delivery to the home.

There is intensive global work on projects and programmes to support e-health (e.g. e-HealthCare, HAVEN, MIRTH, SAPHIRE, UBICARE), independent living (e.g. AMI, Assisted Living Innovation Platform, EQUAL, PERSONA, SOPRANO, SPARC), smart houses (e.g. AMIGO, Bath, Gator, House_n, Millennium Homes), and telecare (e.g. Continua Health Alliance, ETSI, SAPHE).

However, MATCH has a unique focus that is distinguished from other work in important ways. The emphasis is on delivery of care services to the home. Social care is dominant, though healthcare issues are also accommodated.

This requires a wide range of situations in the home to be monitored and managed. MATCH aims to interface with other care services, and therefore integrates a wide variety of care monitoring devices and techniques. The MATCH approach should be seen in the context of home networks rather than healthcare information systems. The work on smart houses (e.g. [5]) tends to concentrate on home automation (e.g. appliances, entertainment, security), with delivery of care being of lesser interest.

As the base, MATCH has adopted OSGi ('Open Services Gateway initiative', www.osgi.org). This is ideal as the approach is vendor neutral, device independent, and focused on service provision through SOA (Service Oriented Architecture). Several projects have used OSGi in healthcare, e.g. e-HealthCare (ehealth.sourceforge.net) and SAPHIRE (www.srdc.metu.edu.tr/webpage/projects/saphire).

MATCH has a unique emphasis on social care using OSGi. Other differentiating factors include the use of ontologies to enhance discovery of home care services, the use of goals and policies to manage these services, and the fusion of multiple technical disciplines: activity monitoring, home networks, multimodal interfaces, speech technology, and stakeholder requirements analysis.

2. Home Care System Architecture

The MATCH home care system follows a service-oriented architecture that links a large number of components. This paper focuses on the components responsible for managing home care networks: the Service Registry (section 3) that now supports a richer set of ontologies, and the Policy System (section 4) that now supports goals. To place these in context, some of the more important elements in the MATCH architecture are as follows.

Base Architecture: The home care system is supported by an OSGi residential gateway (a PC or set-top box in the home). OSGi is a service platform that supports components known as bundles. The service-oriented approach allows a wide variety of loosely-coupled components to be developed and integrated to provide home care services.

Message Broker: This supports communication among MATCH components irrespective of their physical location and which OSGi instance controls them. This is required because of the distributed nature of telecare, e.g. the need to link multiple homes and care centres. Messaging is based on REDS (Reconfigurable Dispatching System, zeus.elet.polimi.it/reds).

Device Support: Driver bundles have been created by MATCH and others to interface a variety of home network solutions: Jini (a service-oriented device architecture), UPnP (Universal Plug and Play for connectivity among consumer devices) and X10 (control of mains appliances). In addition, support has been created for more

specialised devices and functions such as communication via an 'Internet buddy' (www.nabaztag.com), gestural input using a Wii Remote or a SHAKE (www.dcs.gla.ac.uk/research/shake), infra-red control of domestic appliances, mobile messaging, and telecare devices (www.tunstall.co.uk and www.visonic.com).

Sensor Data Management: MATCH supports standard sensors such as beam-breakers, location sensors, movement sensors, opening sensors and pressure mats. Standard actuators are also supported such as gas or water shutoff valves and controllers for domestic appliances. Both wired and wireless devices can be used, though there is a preference for wireless as this is easier to install and reconfigure in a domestic situation. Data mining techniques are applied to sensor data as a means of visualising and analysing activity in the home. This informs professional judgement about long-term trends in the user's activity, e.g. becoming less active or sleeping poorly.

Task Manager: Actions within the MATCH system are mediated through directed graphs of subtasks similar to the task trees used by Concur [6]. Tasks may be simple atomic actions such as turning on a light, or may be long-lived activities that use inputs from a variety of sensors as well as interactions with people.

Interaction Manager: This component cooperates with the Task Manager to determine appropriate modalities for interacting with the end user. For example, a user with hearing difficulties might need visual alerts, or spoken reminders should be given in the user's native language. The currently supported modalities are audio, gestural, spoken, tactile and textual.

Speech Interaction: This supports speech synthesis using natural-sounding voices, speech recognition in restricted domains, and dialogue management for human-computer discussion (e.g. for making appointments). Since many target clients for MATCH are older people, speech synthesis has been adapted for auditory ageing (which may involve hearing loss). Equally, speech recognition has been tailored to reflect possible deterioration in how older people speak, and dialogue management reflects cognitive difficulties that older people might have.

3. Home Network Management

3.1. Service Description

The Service Registry manages the descriptions of services, components, devices and resources within a home care network. (For brevity, these are all called 'components' in what follows.) The registry also responds to enquiries about components that meet certain criteria. Registry clients make use of a uniform discovery interface that hides the underlying descriptions. This is essential because

the actual descriptions are likely to depend on the particular implementation (e.g. a video recorder is very different from the perspectives of Jini, UPnP and X10).

The Service Registry makes use of OWL (Web Ontology Language [9]) as a major standard for describing concepts and their relationships using ontologies. Ontology languages have emerged from work on the Semantic Web, whose goal is to have meaningful descriptions of web content, making indexing and searching easier. The OWL-S extension has been investigated for MATCH to define interface semantics for components. This has been used by others to describe service inputs and outputs in home networks (e.g. [4]). The MATCH approach focuses on integrating existing approaches to component description rather than on developing independent vocabularies.

The registry exploits the descriptive power and logical reasoning of OWL to support a rich environment for describing components used in home care. OWL offers classification, consistency, inference and reasoning in description and discovery. Relationships are inferred among domain-specific properties. The terms and concepts used within a home care network are given a richer, semantically-based interpretation. This allows the MATCH approach to adapt to real-world variety, rather than requiring the world to adapt to MATCH. For example, standard device descriptions are used instead of requiring them to be rewritten for MATCH.

The Service Registry has a number of translation bundles, one for each component domain. These extract generic descriptions from domain-specific ones (e.g. as dictated by a particular protocol). Each translation bundle has knowledge of the relationships between the domain-specific vocabulary and the corresponding generic terms. Where possible, descriptions are translated automatically. Manual intervention may, however, be required (e.g. to define contextual information about a service).

Locating the desired components within a network requires them to be able to describe themselves. Service architectures and protocols for home networks typically have their own approach for component description. Some (e.g. X10) have almost no descriptions, while others use a defined schema or a loose collection of attributes.

Issues can also arise in the choice of terms, since differing vocabularies or schemas are used for different kinds of components. For example, one protocol may describe a component using an *offersService* property, while another may denote the same relation using a *hasFunction* property. While these concepts are semantically similar, they are syntactically different so achieving uniformity is difficult. Classifying network services has a similar problem, e.g. what is termed a *VisualOutput* service might in fact satisfy the requirement for a *VideoDisplay* service.

3.2. Ontology Hierarchy

For comprehensive and flexible description of home networks, a hierarchy of ontologies is used to describe different aspects of the home environment. Higher-level ontologies are grounded in definitions from lower-level ones. The ontology hierarchy is extensible, allowing new concepts and new kinds of components to be described as home care expands. Further protocol ontologies can readily be created and integrated. As new kinds of services become available, additional core ontologies can also be added. The ontology hierarchy in Figure 1 has five main levels.

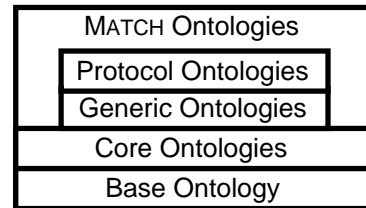


Figure 1. Ontologies for Home Care Networks

Base Ontology: This covers basic concepts in the home environment, e.g. *Device*, *Home*, *Service* and *User*. These concepts are the root ideas used in the higher levels.

Core Ontologies: Each of these elaborates one concept from the base ontology. Each core ontology is responsible for defining the relationships within its domain, and with other concepts in the base ontology. For example, a core ontology specifies that a *Service* may be owned by a *Device* or *SoftwareApplication*; both of these appear in the base ontology. More specialised concepts such as *Channel* and *MessageType* are used to describe service operation, and are related to the basic *Service* concept. As a whole, the core ontologies specify general kinds of home network components such as services and devices.

Generic Ontologies: These build on lower ontologies to define the concepts and relationships required for rich service descriptions. For example, a generic ontology states that a *Lamp* offers a *LightService*; both these concepts are defined at a lower level. A *LivingRoomLamp* is defined as a subclass of *Lamp*, with property *hasLocation* set to *LivingRoom*. Note that it is not necessary to say that a *LivingRoomLamp* offers a *LightService* – this is inferred automatically from the subclass relationship.

Protocol Ontologies: These define relationships and properties for particular domains (usually protocols), grounding them in lower-level ontologies. For example, the UPnP *deviceType* attribute has the same meaning as the *Device* concept specified within the base ontology. The value of this UPnP attribute corresponds to the *DeviceClass* concept within the core ontologies. Thus a relationship is formed between a domain-specific attribute and a generic

ontology concept. As an example, a UPnP lamp is of type *Lamp* from the core ontologies. The lamp inherits properties such as offering a *LightService*. Relationships are also created between the *ServiceType* concept and domain-specific classes. Attributes are either specific to a domain (and are thus captured within a protocol ontology), or are generic enough to derive from lower-level ontologies. This allows domain-specific descriptions of network components to generate a domain-independent classification.

MATCH Ontologies: These describe network components in a manner relevant to home care. This supports more than service discovery – it also supports discovering the communication properties of a service. Standard descriptions convey what a service *does*. The MATCH ontologies convey *how* a service performs its task by means of channels and message types. This level of the hierarchy does not contain pre-defined ontologies. Rather, it encompasses the descriptions derived from the network components. Descriptions of home care components deal with two aspects: concepts defined lower in the hierarchy, and literal values (such as service name) that are unique to each description. The registry manages the MATCH ontologies, performing continuous reasoning and inference over descriptions.

3.3. Service Discovery

The ontology hierarchy allows components to be discovered in a generic way. For example, a registry client may wish to discover if there are any *VideoRecorder* devices within the *LivingRoom*. All video recorders, regardless of their technology and connection, are described in a uniform way within a core ontology. A MATCH component needs to know only the domain-independent concepts used by the base, core and generic ontologies. Domain-specific attributes can be queried if the registry client understands a particular domain, but this is not required to make effective use of components in the home network.

4. Goals and Policies for Home Care

4.1. Policy-Based Management

Policy-based management originated as a means of autonomously managing networks. Policies have been used in many kinds of system management tasks such as access control, network security and quality of service. Ponder (*ponder2.net*) is a widely used and typical approach to policy-based management. Languages like Ponder are well suited for policies in technical applications, but are less well suited to unstructured and user-oriented domains such as found in home care [7]. In these kinds of applications, it can be hard to identify the subject/target of policies, the users and the managed objects. Rule-based sys-

tems have been used in context-aware applications for home and health monitoring (e.g. the CAMUS project, uclab.khu.ac.kr/index_research.php).

Goals state high-level objectives for home care, while policies govern the choices that the home care system may make. As an example, a home care goal might be to ensure that the user remains active. This could be supported by policies that encourage users to sleep appropriate hours, to engage in external activities, and not to watch too much TV. There has been little research on goal refinement into policies, [1] being one of few examples. As a sister project to MATCH, the PROSEN project (Proactive Condition Monitoring of Sensor Networks) has developed techniques for refining high-level goals into policies.

4.2. The MATCH Policy System

A policy system has been created for home care to automate support of stakeholder goals and policies. These are expressed in APPEL (Adaptable and Programmable Policy Environment and Language, www.cs.stir.ac.uk/appel). The policy system architecture is shown in figure 2.

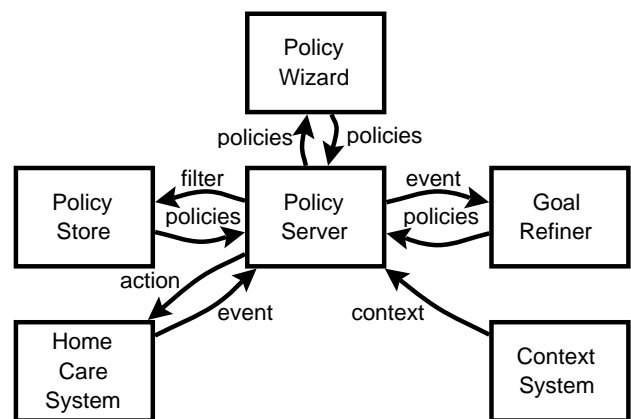


Figure 2. Home Care Network Policy System

Since APPEL is XML-based, a Policy Wizard offers user-friendly definition of stakeholder views (e.g. using near-natural language). Goals and policies are held in a Policy Store – an XML-based database (tuple space).

The Policy Server reacts to events from the home care system such as sensor inputs, user requests and service accesses. It retrieves the goals and policies applicable to an event. These dictate actions that the home should perform such as actuator outputs, user responses and access authorisations. A separate Context System feeds contextual information into the Policy Server, e.g. the user's diary or contact list. The Goal Refiner takes high-level goals for home care and automatically derives operational policies from them.

In general, there are many stakeholders in home care: end users, their informal carers, professional carers, engi-

neers and policy makers. As a result, stakeholder views on how home care should be delivered can readily conflict with each other. As an example, suppose a user with a heart condition lives in sheltered housing. The user may wish to watch TV at any time. To minimise disturbance, the warden forbids TV viewing between 11PM and 7AM. Given the user's heart problems, the doctor wishes the user to avoid scary films. Clearly these views can conflict.

The Policy System handles conflicts at several levels among goals and policies. When these are defined, they are filtered for potential conflict [3]. This identifies problems that require a goal or policy to be reformulated, or that need a dynamic strategy for conflict resolution. When goal refinement is performed, conflicts among goals are detected and resolved. When policies are executed, conflicts among their actions may also be detected and resolved.

The sample goals and policies that follow are written in stylised natural language, partly because this is close to the format supported by the Policy Wizard, and partly because the XML syntax is not so readable. Examples in XML can be found from the APPEL web page cited above.

As a concrete example of message flow in the MATCH system, consider *Policy 1* below. The physical trigger (a door switch) is passed via the message broker to the policy store, where the door state (open) is recorded. Since the state has changed, the policy system is triggered. The policy condition refers to the previously stored bed occupancy state. The policy action is sent via the message broker to the interaction manager. This decides that speech output is appropriate for the user. The speech synthesiser is then tasked to speak the reminder on a nearby loudspeaker.

4.3. Sample Policies for Home Care

In general a policy has an optional trigger (**When**), an optional condition (**If**) and an action (**Do**). All three elements may composite (e.g. multiple triggers combined with 'and'/'or'). Policies can control the home care system [8], but can also help users.

Policy 1: Night wandering is a common issue among the elderly. The following policy says that if the user gets out of bed and tries to leave the house during the night, a synthesised reminder should be spoken:

When the front door is opened, **If** the hour is 11PM–7AM and the user's bed is unoccupied, **Do** remind the user to go to bed as it is night time.

Policy 2: Older people can benefit from simple home automation tasks. The following policy will record a weekday news programme in case the user forgets to watch it:

When it is 10PM on a weekday, **Do** record BBC 1 for 30 minutes.

4.4. Sample Goals for Home Care

Goals state high-level aims for a system. The achievement of a goal is assessed by a numerical goal measure. Although goals resemble policies, goals do not have triggers as they always apply. Goal conditions can therefore refer only to environment values such as the current time or the room humidity. Goal actions state the goal measures that should be minimised or maximised. These are defined in terms of controlled variables (that policies can influence, e.g. the indoor temperature), uncontrolled variables (that cannot be influenced, e.g. the outdoor temperature), and derived variables (that depend on other variables, e.g. the amount of condensation on windows).

In general there may be many goals, and these may conflict in various ways. The relative importance of goals is therefore defined through an evaluation function that combines the goal measures appropriately. This might be a simple weighted sum, or may be an arbitrarily complex function of the measures.

Policies are executable, but goals are inherently declarative and so are not directly executable. The Goal Refiner therefore needs a basis for determining the policies that should be used to achieve goals. This is done by defining prototype policies that are dynamically instantiated to best meet the goals. Prototypes can have parameters that are also dynamically optimised. A prototype has a numerical effect (**For**) on a controlled variable, and therefore on goal measures. The effect may set a controlled variable to a particular value (=), or may increase/decrease it by a certain amount (+, −). A prototype may have several such effects.

Goal refinement is a general activity that can be applied in different domains. [2] explains how goal refinement can be used to optimise sensor networks of the kind that might be deployed in the home. In such a case, the policy system can exercise technical control over the home care system. However, the approach can also be used to optimise goals for stakeholders in home care. An important difference here is that the policy system should avoid *controlling* the user. Rather, the system should *support* the user.

Goal 1: As a *simple* example, the following goal states that the user should maintain an active lifestyle during the week:

When it is a weekday, **Do** maximise *activity*.

Measure 1: Suppose that time spent awake contributes positively to activity, but that time spent viewing TV contributes negatively. The measure of *activity* might be a weighted sum of these:

$$1.5 \times \text{awake_time} - 0.5 \times \text{viewing_time}$$

Prototype policies are defined in a library as candidates for helping with home care goals. The example prototypes below might be considered for Goal 1.

Prototype 1: The following encourages the user to rise an hour earlier than they would naturally if not woken:

When it is 8AM and the user is in bed, **Do** ring the alarm clock, **For** *awake_time* += 1.

Prototype 2: The following limits TV viewing time:

When the TV has been on for two hours per day, **Do** switch it off, **For** *viewing_time* = 2.

Prototype 3: The following presumes that alerting a relative will result in the user taking medication as required:

When the user is an hour late in taking medicine, **Do** ask a relative to remind the user, **For** *medication_dosage* += 1.

Prototype 4: The following encourages the user to go to bed 2 hours earlier than they would naturally:

When it is 10PM and the user is up, **Do** remind the user to go to bed, **For** *awake_time* -= 2.

The Goal Refiner statically filters prototypes against goals by looking to see whether their effects alter the goal measure. In the example above, prototypes 1, 2 and 4 are all associated with goal 1 as they affect its measure. A prototype may contribute to several goals, to one goal or to none of the current goals. The relevant prototypes are instantiated as policies, but linked back to the goals that they affect.

When a trigger occurs, all relevant policies are retrieved as usual. For policies that derive from goals, the Goal Refiner makes an optimum choice among them: a subset of the policies is chosen to optimise the goal measure. In the above example, choosing prototypes 1 and 2 but not prototype 4 maximises the *Measure 1* given earlier.

In realistic applications, goal refinement is much more complex in a number of ways (but still fully automated). For example, there may be 20 goals, 15 variables, and 100 prototype policies; choosing an optimal set of policies is then non-trivial. Different goals may conflict, so a prototype that helps to achieve one goal may hinder another goal. A prototype may therefore not be selected, even though it contributes to some goals.

Goal measures may depend on uncontrolled variables, meaning that the best choice of policies will vary according to the current circumstances. Different policies may therefore be applied at different times. Prototypes, and therefore their effects, may be parameterised (e.g. the time period for prototype 2). The optimisation step also chooses the best values for these parameters dynamically.

There is an initial effort in setting up the library of prototypes. Goals, policies, measures and the overall evaluation function are defined through consultation with the user and with other stakeholders. In practice, this is combined with the assessment procedures that care providers already carry out. Once this point has been reached, the system for supporting goals, policies and conflict resolution is fully automated. This gives considerable flexibility in how home care

networks are managed. Home care support and priorities can also be readily modified by non-technical people.

5. Current Status and Future Work

The Service Registry has been fully integrated with the MATCH home care system to deal with a variety of components from different domains. Future work will focus on ontologies for new classes of services and devices.

The Policy System is operational and integrated with the MATCH system. Goal refinement and conflict handling are being tailored more closely to the needs of home care.

The MATCH system is still under development, but already has a substantial library of components. It has been successfully used in a variety of laboratory-based trials. Trials through real home deployments will begin shortly.

Acknowledgements

The MATCH project has been made possible through financial support from the Scottish Funding Council (grant HR04016). The authors thank their colleagues on MATCH and PROSEN for their fruitful collaboration.

References

- [1] A. Bandara. *A Formal Approach to Analysis and Refinement of Policies*. PhD thesis, Imperial College, London, July 2005.
- [2] G. A. Campbell and K. J. Turner. Goals and policies for sensor network management. In M. Benveniste *et al.*, editors, *Proc. 2nd Int. Conf. on Sensor Technologies and Applications*, pp. 354–359. IEEE, New York, USA, Aug. 2008.
- [3] G. A. Campbell and K. J. Turner. Policy conflict filtering for call control. *Proc. 9th Int. Conf. on Feature Interactions*, pp. 83–98. IOS Press, Amsterdam, May 2008.
- [4] E. Christopoulou and A. Kameas. GAS ontology: An ontology for collaboration among ubiquitous computing devices. *Int. J. on Human-Computer Studies*, 62(5):664–685, 2005.
- [5] 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, Mar. 2005.
- [6] G. Mori, F. Paterno, and C. Santoro. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Trans. on Software Engineering*, 30(8):507–520, Aug. 2004.
- [7] 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, June 2006.
- [8] F. Wang, L. S. Docherty, K. J. Turner, M. Kolberg, and E. H. Magill. Services and policies for care at home. *Proc. 1st Int. Conf. on Pervasive Computing Technologies for Healthcare*, pp. 7.1–7.10. IEEE, New York, USA, Nov. 2006.
- [9] World Wide Web Consortium. *Web Ontology Language (OWL)*. Version 1.0. World Wide Web Consortium, Geneva, Switzerland, Feb. 2004.