Kenneth J. Turner and Gavin A. Campbell. Goals for Telecare Networks, in Abdel Obaid (ed.), Proc. 9th International Conference on New Technologies for Distributed Systems, pp. 270-275, Montreal, July 2009.

Goals for Telecare Networks

Kenneth J. Turner
Computing Science and Mathematics
University of Stirling
Scotland FK9 4LA
kjt@cs.stir.ac.uk

Gavin A. Campbell*
Computing Science and Mathematics
University of Stirling
Scotland FK9 4LA
gca@cs.stir.ac.uk

ABSTRACT

Telecare supports delivery of care to the home. Telecare networks link devices in the home, and provide access to local and remote services. Since telecare systems require a high degree of autonomy, it is desirable to have automated rules that manage their operation. Telecare systems also require flexibility and adaptability, but modifying their technology normally needs expert intervention. The authors and their colleagues have developed policy-based management as a higher-level, user-friendly way of supporting telecare. This work has now been extended to support abstract goals for telecare. These goals manage the distributed networks, devices and services used in telecare. Goals are automatically refined into policies that are executed dynamically. Sample goals and policies are given to illustrate the approach.

1. INTRODUCTION

1.1 Telecare

The world population is gradually ageing, with the percentage of older people (over 65) expected to rise by 2050 to 19.3% worldwide – and much higher in some countries [3]. Although the population is ageing, older people are generally healthier and more active than in previous generations. They usually prefer to stay in their own homes for as long as possible. This is fortunate, since the increasing proportion of older people will make it difficult to provide adequate numbers of care homes. In addition, the cost of looking after someone in their own home is roughly half that of a care home. There is therefore a strong need to help older people prolong independent living at home. Others could also benefit from being able to stay at home, such as those with physical or mental disabilities, or those with long-term medical conditions.

Telecare systems are computer-based systems that support delivery of care to the home. They can provide the user with advice, identify trends that may need intervention, monitor for undesirable situations, reassure family members and informal carers, and relieve professional carers of low-level monitoring tasks. Telecare systems should be appropriate (reflecting different stakeholder viewpoints), customisable (tailored to specific user needs), flexible (supporting a range of solutions), and adaptive (as care needs and conditions evolve). Unfortunately, most telecare solutions are relatively fixed. Where alteration is possible, this typically requires detailed technical knowledge and re-programming.

The authors are contributing to the MATCH project (Mobilising Advanced Technologies for Care at Home, *www.match-project.org.uk*). In support of telecare, this project is exploring a range of advanced technologies such as home care networks, lifestyle monitoring, speech and multimodal interfaces. One aspect of the research is reported in this paper: policy-based management of home care. Many similar projects focus on telehealth: remote monitoring and support of health care in the home. The MATCH project is unusual in its focus on social care (and its relationship to health care). For this reason, MATCH aims to support a mixture of devices and services that provide a comfortable and safe living environment for users – and not just to support their health.

1.2 Related Work on Telecare

There have been numerous projects on aspects of technologies for home care. Topics have included e-health (e.g. e-HealthCare, HAVEN, MIRTH, SAPHIRE, UBICARE), independent living (e.g. AMI, ALIP, 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 distinguishes it from other work in important ways. The emphasis is on delivery of care services to the home, particularly for social care (though health care is not neglected). MATCH aims to interface with other care services, and therefore to integrate a wide variety of care monitoring devices and techniques. The MATCH approach should therefore be seen in the context of home networks. The work on smart houses (e.g. [4]) tends to concentrate on home automation (e.g. appliances, entertainment, security). Like MATCH, a number of projects have used OSGi to support care. However these other projects mostly focus on health care, e.g. e-HealthCare (*ehealth.sourceforge.net*) and SAPHIRE (*www.srdc.metu.edu.tr/webpage/projects/saphire*).

The social care emphasis of MATCH is unusual. Other differentiating factors include the use of ontologies to enhance the 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.

1.3 Related Work on Goals

Planning in artificial intelligence goes back about 40 years (e.g. the STRIPS system). Much more recent is work on goals in agent-

*Supported by the UK Engineering and Physical Sciences Research Council, grant C014804.

NOTERE 2009 July 1-3, 2009, Montreal, Canada
.

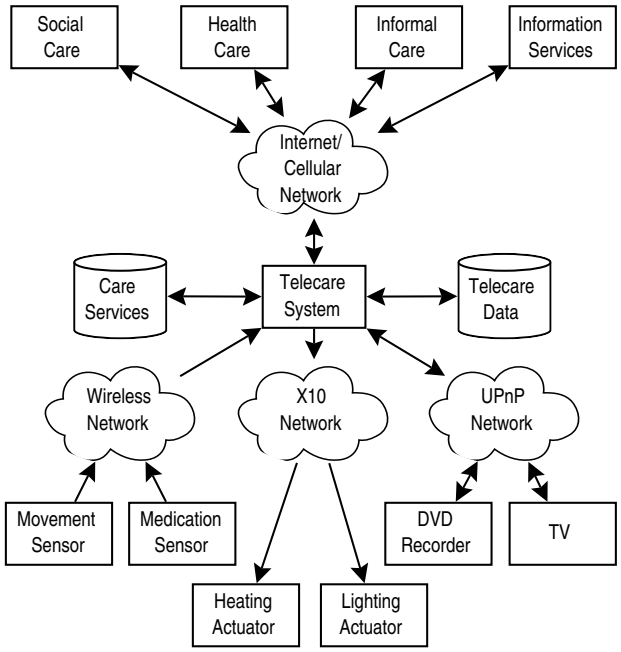

Figure 1: MATCH Environment

based systems. Goals have also been addressed in requirements engineering. Systems such as KAOS [9] aim to build a formal proof that the requirements derived for a system meet its goals.

Goals in a policy context are interpreted differently [2, 5]. Goals can be treated as the top of a policy hierarchy. The idea is that goals can be refined into policies, and even policies into lower-level ones. A formal approach to goal refinement has been developed using Event Calculus and KAOS [1]. As another formal approach, [6] uses temporal logic in a two-stage refinement process from goals to subgoals, and subgoals to policies.

The approach described in this paper differs in significant ways. Although some offline analysis is performed, the bulk of the analysis happens at run time. This allows the choice of policies to depend on the prevailing circumstances. The approach is based on numerical rather than logical reasoning. This is more flexible in that goals need be fulfilled only as far as possible, and not in any absolute sense. A pragmatic rather than a theoretical approach is followed. Formal methods are avoided as they are technically challenging, and performance issues effectively preclude their use at run time.

2. BACKGROUND

2.1 The MATCH Telecare System

The MATCH telecare system runs unobtrusively in the home on a PC-like system. The system adopts a service-oriented architecture, and is supported by OSGi ('Open Services Gateway initiative', *www.osgi.org*). A variety of care services have been created by MATCH for this platform. Monitoring data and system data are stored in the home, but may be remotely interrogated or transferred (subject to security restrictions). Figure 1 shows the telecare system embedded in its wider context. The telecare system is used to link carers and telecare devices via a number of networks. Links to external support are as follows:

professional social care: social workers, occupational therapists, home assistants

professional health care: nurses, surgeries, clinics, hospitals

informal care: family members, neighbours, wardens

information services: health, travel, weather, etc.

These wide area links can use the Internet (via a home broadband connection) or a cellular network (via a mobile phone).

Links to internal devices include the following. The devices shown in figure 1 are just examples from a much wider range.

sensors: these are typically wireless (for ease of installation), but can also be wired (e.g. the KNX standard, *www.knx.org*)

actuators: these are typically wired (e.g. through an X10 mains interface), but can also be wireless

appliance networks: there can be several of these, conforming to standards such as HAVi (Home Audio/Video interoperability, *www.havi.org*), Jini (a service architecture, *www.jini.org*), LonWorks (a building control network, *www.echelon.com*) and UPnP (Universal Plug and Play, *www.upnp.org*)

2.2 Policies

The authors and their colleagues have developed a flexible solution to managing telecare, based on user-defined policies. Policies operate at a relatively high level, and offer many advantages such as orientation towards user needs, customisability and adaptability. A system called ACCENT (Advanced Component Control Enhancing Network Technologies, *www.cs.stir.ac.uk/accent* [8]) has been developed for supporting policy-based management. ACCENT was designed to be friendly for end users.

Policies are written in APPEL (Adaptable and Programmable Policy Environment and Language, *www.cs.stir.ac.uk/appel*). APPEL falls into the category of languages called ECA (Event Condition Action). When a system event (or combination of events) occurs, a check is made on the policy condition(s). Other conditions include the policy's period of validity and whether it matches the user's current profile. If the policy applies, its action(s) are performed.

The policy server is designed as a general-purpose component that can be used in many domains. It therefore relies on the managed system (telecare system here) to notify it of significant events. These events should be relatively high level and infrequent (e.g. the user has fallen, a room has been entered, a visitor has arrived). Conversely, policy actions are performed by the managed system (e.g. alert a family member by text message, turn on the room light, sound a chime).

The APPEL policy language has a core that is specialised for each domain – telecare is currently one of several very different applications. Since policies are internally XML, extensibility is achieved by defining APPEL through a hierarchy of schemas. These are supplemented by ontologies that define the concepts and relationships in each application domain. For the work reported here, the schemas and ontologies have been extended to support goals and prototypes for telecare.

As reported in [10], policy-based management has been developed as part of the MATCH system. Policies have been used in a variety of ways to support telecare:

sensor management: Sensor nodes typically have limited battery power that needs to be carefully managed. For example, policies have been defined to conserve power by altering sensor measurement and reporting intervals, with different strategies depending on the available power level.

telecare services: The services that support care can be defined and managed through policies. For example, policies can control the modality through which users receive reminders (e.g. spoken messages, audio chimes, visual displays, tactile alerts). Policies can also define the actions to be taken in the event of an anomaly (e.g. inform a carer by text message if the user has not risen normally, suggest seeking medical help if the user's heart rate is too high).

home automation: Life in the home can be made safer and more comfortable using policies. For example, the user can be warned of windows left open on leaving the house, or lights can be automatically turned on if the user gets up at night. Efficient use of energy for heating can be managed by policies.

entertainment: Policies can choose the user's favourite music, or can record TV programmes that the user would normally watch.

preferences: In general, policies can capture user preferences. For example, these might include what kinds of food should be stocked, the preferred living room temperature, and what mode of communication is preferred (e.g. house phone, text message, email).

It follows that the policy system must be accessible to the ordinary user. The most visible and important component of the policy system is the policy wizard. This exists in several forms: a web-based wizard that uses near-natural language, a form-based wizard that defines policies by ticking choices on forms, and a voice-based wizard that allows policies to be retrieved and edited using a phone (or microphone). Other conveniences such as pre-defined policy templates make it easier for non-technical users to formulate their own policies.

The policy-based approach also supports multiple stakeholders. In telecare, these include the users themselves, their informal carers and their formal carers. It is not expected that those in care will define policies directly (though this would be possible). Rather, their needs are likely to be expressed to formal carers as part of care assessment. A formal carer can then define policies to meet the user's needs. The care needs of a user often vary over time, especially if they have a degenerative condition such as dementia. An advantage of policies is that they can be changed quickly, without requiring technical expertise such as reprogramming. Furthermore they can be changed remotely, without requiring an on-site visit (which can be time-consuming in rural areas).

Although policies are fairly high-level, they are still relatively imperative. A more abstract way for users to formulate their needs has now been achieved through the design and implementation of a goal system. Goals are persistent, high-level, user-oriented objectives for how a system should behave. They are declarative statements of what is required, not operational statements of how goals should be achieved. In fact, they are sufficiently abstract that they cannot be realised directly. A process of refinement is therefore needed to map goals onto lower-level policies that achieve them.

2.3 Goals

Goals originated in artificial intelligence, where they are typically used by a planning system to build sequences of actions that achieve them. A similar approach has been adopted in agent-based systems. Goals in the context of policy-based management have received little study. Refinement of goals into policies typically makes use of logic: policies are identified to meet goals through a process of logical entailment. In the work reported here, goal refinement is instead viewed as a numerical optimisation problem. The main advantage is that goals can be supported in a fully dynamic way. As the system and its environment change, the policies that achieve goals are automatically evolved to give the best match to current circumstances.

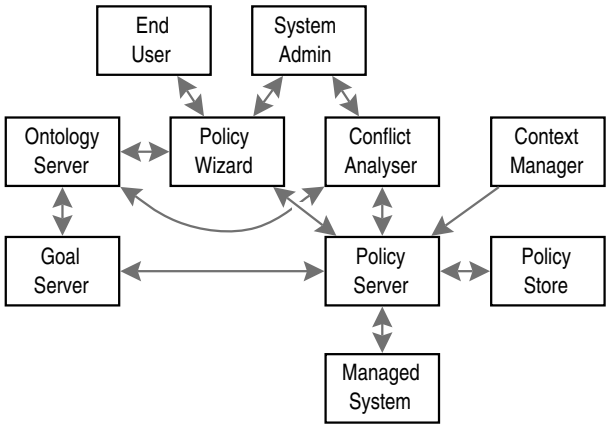

Figure 2: System Architecture

Goals are associated with numerical measures of how well they are achieved. Goal measures are maximised or minimised. The measures are arbitrary functions over system variables, though for practical and technical reasons they are normally weighted sums. Since there are usually multiple (often conflicting) goals, their measures are combined into an overall evaluation (fitness) function that assesses how well a candidate set of policies meets the goals.

Goals are realised through prototype policies ('prototypes') that contribute to them. These prototypes form a library of 'building blocks' for accomplishing goals. When goals are defined, they are statically analysed against the available prototypes. This identifies the policies that *may* contribute towards each goal. The actual selection of policies is deferred until run time since the current circumstances can then be used to make an optimal choice.

3. THE GOAL SYSTEM

3.1 System Architecture

The overall system architecture to support goals and policies is shown in figure 2. This is an extension of previous work on policies for call control [8]. Although the system primarily deals with policies, it also handles goals, prototypes, resolutions (that deal with conflicts), and variables (that may be used in goals and policies). The main components communicate via socket connections, allowing one or many physical systems to be used.

Managed System: the system under control (here, a telecare system).

Policy Store: an XML database that stores information about goals and policies.

Policy Server: the heart of the policy system. The policy server receives goals and policies from the policy wizard, and also information from the context manager. When goals or prototypes are modified, the goal server is called to statically analyse them. When an event occurs in the managed system, it selects relevant policies (i.e. those associated with this trigger and whose conditions are met). If any triggered policies derive from goals, the goal server is asked to find the optimal

set. Conflicts among policies are then automatically detected and resolved. Finally, an optimal and compatible set of actions is sent to the managed system.

Policy Wizard: a user-friendly interface for defining and editing goals and policies.

Context Manager: an interface for providing additional information about the managed system (e.g. the home configuration).

Conflict Analyser: a tool to analyse policies offline for possible conflicts.

Ontology Server: a generic interface to ontology-based information about an application domain (e.g. telecare). Domain-specific ontologies are used by the policy wizard, the conflict analyser and the goal server.

Goal Server: the heart of the goal system. The static goal analyser is invoked when goals or prototypes are altered (see section 3.3). The dynamic goal analyser is invoked when goal-derived policies are triggered (see section 3.4).

3.2 Goals and Prototypes

Syntactically, a goal is a simplified form of policy. There is no trigger, because goals always apply. A goal may have a (compound) condition, but lack of a trigger means the condition may use only general contextual information (e.g. the current time or room temperature). A goal has a single action of the form maximise(*measure*) or minimise(*measure*). The measure is a numerical assessment of how well a goal is achieved. In general, a measure is defined by a formula using system variables from the domain ontology. Some of these are held per user (or entity), while others are shared across the system. Quantitative variables have an obvious measure (e.g. temperature in $^\circ$C), while qualitative variables are just numbers on a scale (e.g. social contact 0 is low, 10 is high).

Uncontrolled Variables: these variables are beyond the control of the policy system, typically 'environmental' factors (e.g. outdoor temperature, weather forecast).

Controlled Variables: these variables are managed by the policy system (e.g. energy use, medication compliance).

Derived Variables: these are pseudo-variables defined in terms of (un)controlled variables (e.g. the measure of user activity).

Goals are realised through sets of policies. The need to support goals leads to defining separate prototype policies. These are very similar to regular policies, but are considered separately by the goal system.

Prototypes have an effect attribute that defines how they modify one or more system variables (i.e. how they contribute to goal measures). The effect of a prototype is an abstraction of the actions it can perform. At definition time, effects are used to identify the relationship between goals and prototypes. At run time, they determine the policies that optimally satisfy the goals. Prototypes are also allowed to have parameters that are optimised at run time by the goal system.

An individual effect names a system variable, an operator, and an expression (e.g. 'medication_compliance += 3'). The basic operators are '=' (set a variable), '+=' (increase it) or '-=' (reduce it). In special cases, prototype effects are not allowed at the same time. There are therefore special 'exclusive' forms of the operators: '+~' and '-~'.

3.3 Static Analysis

Static analysis in the goal system is activated when a goal or prototype is created, modified or deleted. Whether a prototype contributes to a goal is determined by comparing its effects to how the goal measure is defined (i.e. which system variables it uses). A prototype contributes to a goal if it affects one or more system variables involved in the measure. The prototype effect may modify an arbitrarily complex measure. The result may therefore not be known until run time, when an effect may worsen or improve the outcome of using this prototype.

A library of useful prototypes is created by a domain expert. This uses information in a domain-specific ontology about system variables. Typically a prototype contributes to one or more goals. However, a prototype may not contribute to any goals if it is irrelevant for the current ones. Policies are created from prototypes with conditions that combine the conditions of goals and prototypes (though there are subtleties in doing this). To the rest of the policy system, these generated policies look like regular ones – but distinguished by a supports_goal attribute. Prototype parameters remain uninstantiated until run time.

3.4 Dynamic Analysis

Dynamic analysis in the goal system is activated when an event trigger identifies the policies to execute. If no policies are goal-derived, the policy server continues policy execution as normal. However if there are such policies, the dynamic analyser is asked to choose a subset that optimises the overall goal evaluation function. This function combines goal measures, and thus depends on system variables that may change dynamically. This selects the best policies for the current circumstances. It also means that policies can change as the system evolves over time. This dynamic approach is much more flexible than the offline, logic-based techniques used in most other approaches.

An optimisation algorithm determines the best policy combination as judged by the evaluation function. Prototype parameters are also chosen optimally through the same procedure. The goal system is designed to use any optimisation algorithm that conforms to a defined interface, though only a simple but effective algorithm is used currently. Certain kinds of conflicts are automatically handled during optimisation (e.g. if prototype effects are incompatible). The policy system can handle hundreds or thousands of policies. Even if these are all derived from goals, it has not been found that performance is an issue. What matters is how many such policies are triggered *at the same time* – typically only around half a dozen.

The optimal policy set is analysed for conflicts among their actions. This uses special resolution polices that detect problems and produce a compatible set of actions [7]. Finally, these optimal and compatible actions are executed by the managed system.

4. APPLICATION TO TELECARE

4.1 Goals and Prototypes

A telecare system typically has many system variables. For example, uncontrolled variables may include forecast (weather forecast) and outdoor (outdoor temperature, $^\circ$C). Controlled variables may include additive (additive intake, g/day), awake (awake time, hours), chill (chill risk, 0..10), contact (social contact, 0..10), energy (energy consumption, kWh), medication (medication compliance, 0..10), night (night awake time, hours), pollen (pollen risk, 0..10), security (security level, 0..10), viewing (TV viewing time, hours), and volume (noise level, dB).

Goal	Definition
1 (doctor)	**do** maximise medication compliance **measure** 2.0×medication
2 (warden)	**do** maximise security level **measure** 7.0×security
3 (relative)	**do** maximise social contact **measure** 3.3×contact
4 (therapist)	**if** it is a weekday **do** maximise user activity **measure** 0.6×awake - 5.0×viewing + 10.0×contact
5 (doctor)	**do** minimise allergen exposure **measure** 3.3×pollen + 5.0×additive
6 (warden)	**do** minimise energy consumption **measure** 3.3×energy
7 (warden)	**if** it is 11PM–7AM **do** minimise housing disturbance **measure** 1.0×threshold(volume,60) + 1.0×night
8 (user)	**do** minimise user discomfort **measure** 1.0×ideal(indoor,21) + 2.5×chill + 0.2×ideal(volume,80)

Table 1: Sample Goals

Sample goals for telecare are shown in table 1, drawn from a larger set that deals with many other factors. The person likely to define each goal is indicated. Instead of using the internal XML representation, goals are described here in stylised English similar to the policy wizard interface.

The measures are mostly weighted sums, though two standard functions can be used: ideal (deviation from an ideal value) and threshold (amount above a minimum value). Scaling factors for measures (e.g. 2.0 in goal 1) are automatically chosen to ensure measures have similar numerical values in typical circumstances.

The measures are combined into an overall evaluation function that is usually a weighted sum of the goal measures. Maximised measures have positive values, while minimised ones are negative. The goal system includes automated sensitivity analysis to check how its behaviour depends on the choice of weights. It has been found in practice that the choice of weights is not critical, e.g. the outcome is usually the same even if goal weights vary over a ratio of 10:1. A weight of 1 is therefore usually a satisfactory choice.

The prototype library includes a wide range of options to support telecare. Table 2 shows example prototypes, again in stylised English. These are relevant to maintaining an active and comfortable environment for the user. Prototypes 2, 3 and 4 differ in their conditions (temperate, cold, warm weather) and hence in their effects.

When the policy system is managing a device (e.g. air conditioning), its actions are direct. However when people are involved, its actions are indirect. For example, prototype 4 encourages (but does not force) the user to go for a walk. Actions are also high-level and do not imply a particular modality. For example, prototype 6 asks a neighbour to drop by. Depending on user preferences, this might be achieved through a text message, a synthesised voice message, or an email message.

4.2 Static Analysis

As each goal is defined, the prototypes that contribute to it are determined. Table 3 shows the results of statically analysing prototypes against goals. (Goals 1 and 7 are not supported by these particular prototypes.) This creates a goal-derived policy for each prototype, stating which goals each policy contributes to.

Prot.	Definition
1	**when** indoor temperature $>$ 30 **and** outdoor temperature $<$ 25 **do** turn heating off **and** turn air conditioning off **and** open windows for 1 hour **effect** indoor -= 4 **and** security -= 3 **and** pollen += 1.5
2	**when** the user has not left the house during 2PM-5PM **and** outdoor temperature is 5..25 **do** encourage the user to go for a walk **effect** contact += 1
3	**when** the user has not left the house during 2PM-5PM **and** outdoor temperature $<$ 5 **do** encourage the user to go for a walk **effect** contact += 1 **and** chill += 5
4	**when** the user has not left the house during 2PM-5PM **and** outdoor temperature $>$ 25 **do** encourage the user to go for a walk **effect** contact += 1 **and** pollen += 4
5	**when** the user has no phone calls during 8AM-5PM **do** ask a friend to phone in the evening **effect** contact += 1
6	**when** the user has not left the house during 9AM-5PM **do** ask a neighbour to drop by in the evening **effect** contact += 2
7	**when** indoor temperature $<$ 18 **and** windows are open **do** turn heating on for 1 hour **and** turn air conditioning off **and** close windows **effect** indoor+= 5 **and** energy += 3 **and** security += 3
8	**when** indoor temperature $>$ 27 **do** turn heating off **and** turn air conditioning on for 1 hour **effect** indoor -= 6 **and** energy += 2

Table 2: Sample Prototypes

	Prototype							
Goal	1	2	3	4	5	6	7	8
2	✓						✓	✓
3		✓	✓	✓	✓	✓		
4		✓	✓	✓	✓	✓		
5	✓			✓				
6							✓	✓
8	✓		✓				✓	✓

Table 3: Goals affected by Prototypes

4.3 Dynamic Analysis

Suppose that the indoor temperature is 17°C when 5PM comes round. The policy server will determine that policies 1, 2, 5, 6 and 7 are eligible for execution. These will be passed to the dynamic analyser, which will retrieve the current values of the system variables. The dynamic analyser will report that policies 2, 5 and 6 are optimal in these conditions. Policy 7 would have been included if the indoor temperature had been 16°C or lower, or policy 8 if it had been 26°C or higher. The actions of the optimised policies are then checked for conflicts (such as trying to turn the heating both off and on), though in this example there are none.

A substantial number of things happen when a trigger occurs. The entire procedure for selecting policies, optimising goals, resolving conflicts, and dictating actions takes about two seconds: one second in the policy server, and one second in the goal server. If goals are not used, only the policy server is involved. Fortunately, policy-related events in a telecare system are relatively infrequent. The processing overhead is therefore believed to be acceptable – especially given the considerable flexibility and control that goals and policies offer. However, the authors expect to be able to reduce this overhead in a number of ways such as caching goal analysis results.

5. CONCLUSION

It has been seen how goal refinement into policies can be formulated as a numerical optimisation problem. The achievement of goals is assessed through measures defined in terms of system variables. Prototypes contribute to goals through their effects on these variables. When goals and prototypes are altered, static analysis determines the relationship among these. This creates regular policies that are linked to the goals they support. When system events occur, an optimal selection is made of goal-derived policies. The dynamic nature of the analysis means that goals are best achieved according to the current circumstances.

The goal system has been used to enhance the policy-based system for managing telecare. This is linked to a variety of sensors, actuators, appliances and services. The work has so far been evaluated only in a lab setting, but will shortly be deployed in the homes of real users.

The goal system has been illustrated for telecare. However, the techniques and tools are multi-purpose. For example, they have also been used to support goals and policies for managing sensor networks, wind farms and Internet telephony. It is therefore believed that the approach is general and will find value in many domains.

6. REFERENCES

[1] A. K. Bandara, E. C. Lupu, J. D. Moffett, and A. Russo. A goal-based approach to policy refinement. In *Proc. Workshop on Policies for Distributed Systems and Networks*, pages 229–239. IEEE Computer Society, Los Alamitos, California, USA, 2004.

[2] S. Davy, B. Jennings, and J. Strassner. The policy continuum – Policy authoring and conflict analysis. *Computer Communications*, 31(13):2981–2995, 2008.

[3] L. A. Gavrilov and P. Heuveline. Aging of population. In P. Demeny and G. McNicoll, editors, *The Encyclopedia of Population*, pages 27–50. MacMillan, London, UK, Jan. 2003.

[4] 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.

[5] J. D. Moffett and M. S. Sloman. Policy hierarchies for distributed systems management. *Selected Areas in Communications*, 11(9):1404–1414, 1993.

[6] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, and A. L. Lafuente. Using linear temporal model checking for goal-oriented policy refinement frameworks. In *Proc. Workshop on Policies for Distributed Systems and Networks*, pages 181–190. IEEE Computer Society, Los Alamitos, California, USA, 2005.

[7] K. J. Turner and L. Blair. Policies and conflicts in call control. *Computer Networks*, 51(2):496–514, Feb. 2007.

[8] 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.

[9] A. van Lamsweerde and E. Letier. From object orientation to goal orientation: A paradigm shift for requirements engineering. In *Proc. Radical Innovations of Software and Systems Engineering in The Future*, number 2941 in Lecture Notes in Computer Science, pages 153–166, Berlin, Germany, Mar. 2003. Springer.

[10] F. Wang, L. S. Docherty, K. J. Turner, M. Kolberg, and E. H. Magill. Services and policies for care at home. In J. E. Bardram, J. C. Chachques, and U. Varshney, editors, *Proc. 1st Int. Conf. on Pervasive Computing Technologies for Healthcare*, pages 7.1–7.10. Institution of Electrical and Electronic Engineers Press, New York, USA, Nov. 2006.