Kenneth J. Turner. Managing Home Care with Loose Policies,
Ambient Intelligence and Smart Environments,
6(5):489-506, IOS Press, August 2014.

Journal of Ambient Intelligence and Smart Environments 6 (2014) 489–506
IOS Press

Managing Home Care with Loose Goals and Policies

Kenneth J. Turner *
Computing Science and Mathematics,
University of Stirling,
Stirling FK9 4LA,
UK

Abstract. Goals and policies are presented as special kinds of rules for managing systems in a flexible way without requiring specialised technical knowledge. However, it is not always possible to rely on exact information for such an approach. Policies should therefore not have to be formulated in terms of precise inputs and outputs. Instead, it is desirable to allow loose goals and policies that accommodate probabilistic system inputs/outputs and fuzziness in rules. This is a general solution that is relevant to many different kinds of applications. The paper uses automated home care management as a concrete illustration of how the approach works. The overall system architecture is presented, along with an overview of the language for expressing goals and policies. The extensions made to allow looser formulations are described. An extended worked example explains various aspects of the approach. The paper concludes with a user evaluation and a discussion of the work.

Keywords: Home Care, Fuzzy Logic, Goal, Policy, Policy Conflict, Probability, Telecare

1. Introduction

This section explains the advantages of rule-based systems and their role in home care. The relationship between the paper and the work of others is discussed.

1.1. Motivation

The general aim of this work is to support flexible and automated system management. The advantage of the approach is that system logic is exposed and made available for inspection and change. When it comes to adaptation and customisation of the system, this largely removes the need for specialised technical knowledge and programming ability.

The approach described in this paper uses rules (goals and policies) to determine how the system should react to inputs. This is a generic approach that has been used by the author for call control, wind farm management and home care.

The paper deals with two problems in rule-based systems: the sensor information that guides rules may not be certain, and it can be difficult to formulate precise rules. A new approach has been developed using loose rules that deal with uncertain data. As a concrete and important application, home care is used as the illustration.

1.2. Home Care Context

The world population is ageing, with the percentage of older people (over 65) gradually rising. In the UK, for example, this percentage was 24.4% in 2000 and is expected to be 39.2% by 2050 [16]. A similar situation applies in other developed countries, with much higher percentages forecast for some areas (e.g. 71.3% by 2050 in Japan). Clearly this will increase the demand for care of older people. Although people are living for longer, many have to deal with long-term, age-related conditions.

The growing percentage of older people, coupled with pressure on social and health care budgets, means

*E-mail: kjt@cs.stir.ac.uk

that care providers will be increasingly challenged to cope. As a result, it will not be feasible to provide sufficient care homes (which are much more expensive than looking after someone in their own home).

Technology to support home care delivery has been identified as part of the solution. Telecare (also called assisted living) refers to remote support of social care at home. This includes monitoring for undesirable situations (e.g. falls, flooding, night wandering) as well as services for the less able (e.g. curtain openers, door entry phones, home automation). Telehealth refers to remote support of health care at home. This includes remote consultation and diagnosis as well as monitoring health parameters (e.g. blood pressure, heart rate, seizure risk).

Home care technologies offer significant benefits. Particularly in rural settings, the ability to support care at a distance can save substantial travel. Many health authorities are promoting self-care at home rather than relying exclusively on centrally provided care. Trends, anomalies and alert conditions can be identified and reported to a central location (e.g. a health centre or a call centre). Family members can be reassured that the user is being monitored for undesirable situations. Professional carers can also be relieved of low-level monitoring tasks. Older people can therefore be assisted to stay longer in their own homes, where they are in familiar surroundings and near to the people and the area they know.

1.3. Home Care Systems

A home care system is a computer-based system that supports delivery of care in the home. Such a system is able to collect, analyse, react to, and forward care data from a variety of sources. Besides physical sensors for input, the system can also use software services (e.g. for communication, speech input or weather forecasts). The system is able to respond through a variety of actuators and services to control appliances, to manage the home environment, to signal alert conditions, etc.

Automated support for home care is a broad area that relies on techniques from telehealth, telecare, home automation and smart homes. For this reason, the treatment in this paper is intentionally broad. As an example of the breadth needed, home care may require attention to environmental factors. This is often an issue because older people are prone to not heating their home properly (thus risking a chill or pneumonia), or someone with an allergy may need to avoid pollen entering the home due to open windows. Managing energy consumption can also be an issue because older people may be subject to 'fuel poverty' and can thus benefit from automation in reducing energy needs.

As a further broad issue, time figures in several aspects of home care: medication needs to be taken at specific times, and alerts may depend on the time of day (e.g. out of working hours, a problem should be reported to a neighbour rather than to a clinic).

Information from external services also affects home care. For example, managing the home environment might use temperature or pollen forecasts, and speech recognition may be used for interpreting user requests (e.g. for help or advice).

It is desirable for a home care system to be user-adjustable, allowing customisation for individual user needs and adaptation to changing circumstances. However, current solutions tend to be rather inflexible. Even if the system is designed to be programmable, specialised technical knowledge is usually required. For this reason, the author has developed an approach to automated home care management using goals (high-level user objectives) and policies (lower-level system rules). For example, [19] describes how the approach has been used for flexible management of smart homes.

1.4. Related Work

1.4.1. Policy-Based Management

Policies have been used in applications such as access control, network/system management and quality of service. Policies are rules that are automatically applied when events occur. Most policy languages are in ECA form (Event, Condition, Action). Rule-based systems have a simpler EA form (Event, Action) because they do not distinguish triggers and conditions. Examples, drawn from a large field, include the following:

- Drools (*www.jboss.org/drools*) is a rule-based approach for enforcing business rules. This is an implementation of the Java community standard for a Java Rules Engine (JSR 94). Its focus is more on business logic than on system control.
- Homer [13] uses rules to manage the home. It is mostly focused on home automation, though it is also relevant to home care. The main contributions of Homer are an extensible component architecture and a policy system that integrates well with this.
- Police [8] follows a traditional policy-based approach. It deals particularly with management

and conflict in a distributed setting. The approach avoids modality conflicts, but provides mechanisms for handling domain-specific conflicts.
– Ponder [7] is a well-known policy-based approach. It offers a mature methodology for handling policies in applications such as system management and sensor networks. Ponder supports policy domains, policy conflicts and policy refinement.

For the work in this paper, the human aspects of home care tend to rule out the more technically-oriented policy approaches used in system management. As argued in [25], a different kind of policy approach is needed for 'softer' management tasks of the kind found in human-oriented systems such as telecare.

[17] uses ECA rules for ambient intelligence in a home context. This work is complementary to that reported here as it aims to infer what the rules should be. The approach observes how people interact with their environment and can thus infer what their preferences are, such as for lighting levels.

[11] describes a rule-based system for smart homes. However, this is a rather heavyweight solution that expects home devices to be interconnected via an Ethernet. The system supports basic ECA rules, but these do not seem to be intended for definition by end users.

Context-aware systems (e.g. [3]) aim to make a system reactive to context, and in that sense have some affinity to policy systems. Gaia [14] creates 'active spaces' from physical spaces supplemented by a context-aware infrastructure. However, context awareness is a separable aspect. The policy system described in this paper accepts information from an external context system in order to influence its behaviour. Any third-party context system could be used to provide this information.

The approach in this paper is called ACCENT (Advanced Component Control Enhancing Network Technologies [23], *www.cs.stir.ac.uk/accent*). This is an approach and a set of tools for managing systems through goals and policies. ACCENT and its accompanying policy language APPEL (Adaptable and Programmable Policy Environment and Language [24], *www.cs.stir.ac.uk/appel*) were originally developed for Internet telephony. Subsequently they have been extended into applications such as home care, sensor networks and wind farms.

1.4.2. Goal-Based Management

Goal refinement has been investigated for many years. In artificial intelligence, for example, planning approaches such as STRIPS (Stanford Research Institute Problem Solver) go back about 40 years. More recent work includes the following:

– Agent systems often follow a goal-based approach. As an example, 3APL (Agent Programming Language, *www.cs.uu.nl/3apl*) defines goals and beliefs. Plans are created from predefined rules, using an action base to achieve goals.
– Goal refinement has been treated from a logic perspective by several researchers. For example, [2] uses event calculus for formal refinement of goals into system operations that achieve them.
– Requirements engineering has also made use of goal concepts. Approaches such as KAOS (originally Knowledge Acquisition in Automated Specification [26]) make use of refinement patterns to decompose goals into subgoals. [15] uses temporal logic in the refinement of goals into subgoals, and then subgoals into policies.

The work in this paper has a deliberately pragmatic philosophy for combining goal-based and policy-based management. Logical approaches require specialised expertise that make them inappropriate for ordinary users. They also usually involve lengthy computations that make them unsuitable for real-time use. They are most appropriate for static analysis, but even there they require detailed knowledge that may be unavailable or proprietary (e.g. how particular devices work).

1.4.3. Uncertain Values

Probability theory is widely known and needs no introduction. Fuzzy logic (e.g. [1,30]) is well established in mathematics and in control theory, so only a very brief summary is given below.

A fuzzy logic system takes crisp inputs and fuzzifies them using linguistic variables (e.g. an indoor temperature of 30°C might be treated as *hot*). Inference rules use these fuzzy inputs to determine fuzzy outputs (e.g. it might be inferred that the air conditioning should be set to *high*). The fuzzy outputs are accumulated and then defuzzified to obtain crisp outputs (e.g. the air conditioning should be set to 90%). It is claimed that fuzzy logic offers a flexible and easily understood ways of controlling systems.

Although the techniques of fuzzy logic and probability are well known, there have been very few at-

tempts to combine these with policy-based management. [9] presents an approach for managing Quality of Service in differentiated service networks. This uses plain if-then rules in conjunction with fuzzy logic to manage packet scheduling in the face of variable demand. [28] uses simple policies in a probabilistic way to manage load-sharing and protection in networked systems. The focus of that work is on using ant-like mobile agents to find an optimal solution for conflicting policies.

The present paper is believed to be novel as it combines management using full (ECA) policies with fuzzy and probabilistic values. The application to home care is also new.

1.5. Contribution of The Paper

As reported in [19], the author had previously created and evaluated an approach for flexible management of smart homes. The present paper reports the following new contributions:

- Previously the policy system and its policy language dealt only with certainty in inputs, expressions, outputs and rules. The new approach allows for uncertainty in the form of fuzzy and probabilistic values in triggers, conditions and actions. The combination of fuzziness and probability with policy-based management is believed to be unique, with only a couple of related efforts in networking [9,28]. The application to automated home care is also unique.
- The new approach allows rules for home care to be defined in a more natural way because people normally prefer to think in loose terms such as medication being 'late', the user 'probably' having fallen, or the pollen level as being 'low'. As discussed in section 6.2, an evaluation with users has shown that the new approach is easier to formulate and to understand than the original one. It is also less 'brittle' in not requiring precise values that a user might find difficult to choose.
- The new support for fuzzy and probabilistic values has had a profound effect on the design of the policy system and its underlying policy language. Where previously the system could follow simple boolean logic, it now has to reason with information that is definite (boolean) or uncertain (fuzzy, probabilistic).
- Resolution of conflicts among policies now takes uncertainty into account. For example, a high-confidence action may be chosen in preference to a low-confidence action that conflicts with it.
- The domain ontologies have been extended to support fuzziness and probability, combining this with other knowledge about the relevant application area (home care in this paper).

1.6. Structure of The Paper

Section 2 motivates the new work on uncertain values and fuzzy rules, and introduces the support for them. Section 3 provides an overview of the APPEL language for goals and policies. Section 4 discusses how probabilistic inputs and fuzzy rules are handled by the system. Section 5 gives an extended example of the approach by illustrating its use in home care management. Section 6 rounds off the paper with a summary and user evaluation of the work.

2. The ACCENT Policy System

This section motivates the new work on allowing uncertain values and fuzzy rules. The high-level architecture of the home care system is introduced.

2.1. Uncertainty in Automated Home Care

Home care systems typically depend on precise inputs. This is appropriate for conventional rule-based systems, e.g. for access control or system management. Here the inputs provide exact information such as 'the user wishes to access this resource' or 'the channel has this bandwidth'. This is also the approach taken in the author's previous work, but it implies a misleading accuracy when used in the context of home care.

As discussed in section 1.3, automating home care requires attention to a wide range of factors. This information is often uncertain rather than being precise. Examples of this imprecision include the following:

Health Information: Measurements of vital signs such as blood pressure or blood sugar are never exact. Signals from equipment such as medication dispensers are also open to interpretation. For example it may be reported that a pill has been dispensed, but has the user actually taken it? Similarly, other sensors such as fall detectors and seizure detectors do not guarantee 100% accuracy in what they report.

User Activity: Home care systems often try to monitor what users are doing in the home. This is useful to assess, say, whether they are cooking meals regularly or are having an undisturbed sleep. However, this has to rely on activity sensors that may be imprecise. Activity information is also difficult to interpret if the home has several residents.

Environmental Conditions: The home system typically ensures that the user is comfortable, adjusting factors like room temperature and humidity, light level, audio, etc. However, physical measurements are always subject to some uncertainty. Thus a temperature sensor might report that a room is 20°C, but the actual measurement is really 20 ± 1°C (i.e. with some mean and standard deviation).

Time Conditions: Computer clocks are prone to drift and so may be some seconds out. As a result, there is imprecision in the absolute time. This can become significant near a time boundary. For example if a condition is from 9AM to 5PM, the time may not be correctly determined close to the boundaries of this range. Similarly, the day or date may not be precise close to midnight. This could result in a rule being incorrectly triggered or being missed.

External Services: Making sure the home remains comfortable could depend on weather forecasts which are not, of course, certain. Other software services include speech recognition so that the user can make requests for action or help. In practice, speech input is associated with a confidence level. For example, the user may have asked for assistance ('Help!', confidence 0.7) or may have just cursed ('Hell!', confidence 0.3).

A further issue is that it can be difficult in some applications to write precise rules. In purely technical applications this is not likely to be a problem. But in human-oriented applications such as home care it is unrealistic to expect users to be completely precise when defining rules. Rather, a degree of looseness is more appropriate. For example, a precise rule for medication reminders might say 'when the level of medication compliance is below 3, set the reminder interval to every 10 minutes'. A resident is unlikely to be able to fix on such exact values. A more human-meaningful rule would be 'when the the level of medication compliance is *low*, set the reminder interval to *frequent*'.

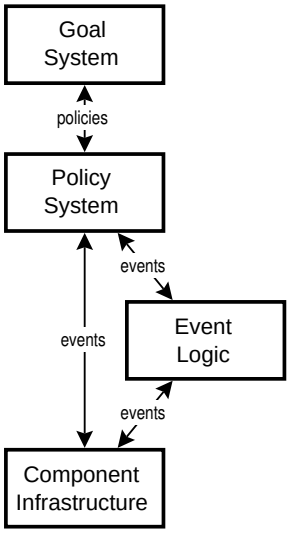

Fig. 1. System Architecture

To address these problems, the paper describes new developments in a rule-based system to permit two kinds of looseness: fuzziness in rules, and probabilistic system inputs/outputs. This is a new development in policy-based management that makes it more appropriate for human-oriented applications.

2.2. High-Level System Architecture

The whole system that supports goals, policies and devices is called ACCENT (Advanced Component Control Enhancing Network Technologies [23]). This has the following layers:

Goal System: This is the primary level at which end users can configure the system behaviour. Goals are high-level user objectives such as complying with medication, avoiding allergens, or eating properly. When external events happen, the goal system chooses an optimal set of policies to achieve them.

Policy System: This is a secondary level at which users might configure the system behaviour, though it is expected that only formal carers will take responsibility for defining policies. The policy system receives triggers (e.g. medication has been dispensed, the user has fallen) and responds with actions dictated by the policies (e.g. turn off a medication reminder, report a fall to the health centre).

Event Logic: Event logic can optionally be used to filter and manipulate triggers and actions. As an ex-

ample of a synthetic trigger, a fall alert may not be reported unless the fall detector signals a possible fall and the user remains motionless. As an example of a synthetic action, a medication reminder to a user might be tried in several ways (e.g. as a spoken message then as a text message).

Component Infrastructure: This includes all the system components along with the general infrastructure. In the case of a home care system, it includes all the sensors, actuators and services that support care. The work in this paper builds on the OSGi platform (originally Open Services Gateway initiative, *www.osgi.org*). The infrastructure exchanges messages with external devices and services.

The remainder of this paper focuses on goal and policy aspects. See [18] for more information about the event logic, and [21] for details of the component infrastructure.

2.3. Goal and Policy System Architecture

A more detailed expansion of the goal and policy systems is given in figure 2. System elements have the following functions:

Managed System: the home system under control.

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 user's diary or the household configuration).

Policy Server: the heart of the policy system. The policy manager is the interface to the policy store, isolating the rest of the system from the particular choice of database. It receives new or updated goals and policies from the policy wizard, and also contextual information from the context manager. When goals or prototype policies are modified, the static analyser is notified. This may result in changes to the generated policies. The policy manager is also asked to query the policy store when event triggers are received. These arrive from the managed system and are passed to the policy selector. This chooses relevant policies (i.e. those associated with this trigger and whose conditions are met). If any triggered policies are derived from goals, the dynamic analyser is notified. This produces an optimal set of policies that are submitted to the conflict manager. Conflicts among policy actions are detected and resolved. This results in an optimal and compatible set of actions for execution by the managed system.

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

Conflict Analyser: a tool to analyse policies offline for conflict-prone interactions [6,20].

Ontology Server: a generic interface to ontology information about each application domain [5]. A domain-specific ontology is used by the policy wizard to define valid goals and policies. Ontologies are also used by the conflict analyser and by the goal system.

Goal Server: the heart of the goal system. The static analyser is invoked when goals or prototype policies are added, modified or deleted. The dynamic analyser is invoked when optimising goal-derived policies.

A worked example is given in section 5 to show how the main system elements cooperate with each other.

3. The APPEL Policy Language

This section provides an overview of the policy language as background for what follows. Only the major features of the language are covered here, so [24] can be consulted for more detailed information.

3.1. Goal and Policy Language

The ACCENT system supports a policy language called APPEL (Adaptable and Programmable Policy Environment and Language [24]). APPEL is an XML-based language, but is designed for human-oriented management tasks. A regular policy normally has one policy rule, but can combine multiple rules. For example, a policy can try alternative policy rules in sequence until one is found to be activated.

A rule is activated if its triggers occur and its conditions hold. Multiple triggers can be combined with **and** and **or**, while complex conditions can use **and**, **or** and **not**. An activated policy then performs its action (or multiple actions combined with **and** and other operators). Other language elements such as goals, prototype policies and resolution policies resemble regular policies in structure. The form of policy triggers, conditions and actions depends on the application domain and is defined by a domain-specific schema.

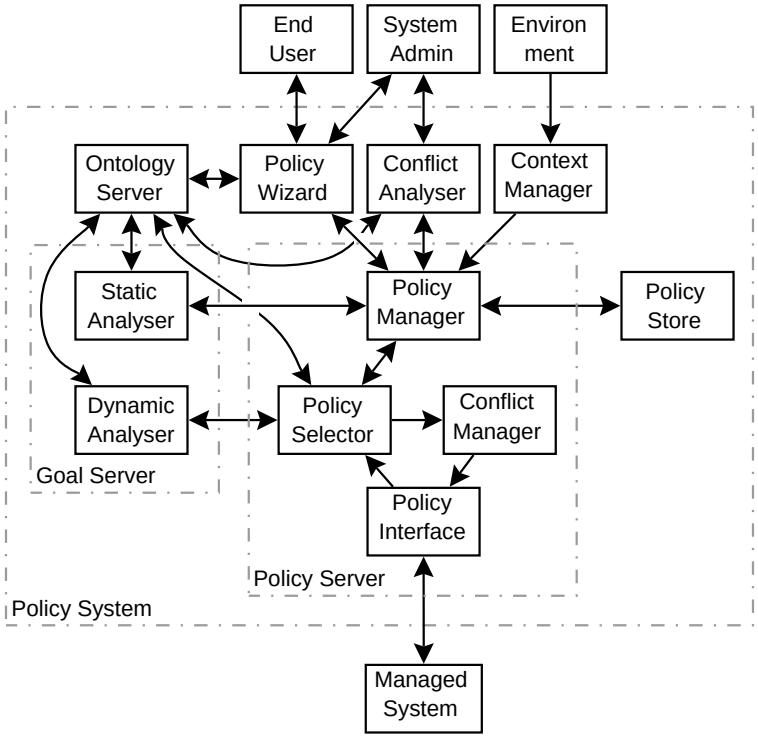

Fig. 2. Policy System Architecture

Obviously a user is not expected to write or to understand XML. A variety of wizards therefore support user-friendly definition and editing of APPEL. A web-based wizard allows local or remote editing using near-natural language (and is multi-lingual). A digital pen wizard allows simple policy definition by ticking items on a form. A speech-based wizard supports policy definition through speech dialogues using VoiceXML [27]. For readability, examples of APPEL in this paper use near-natural language in the style of the web-based wizard.

3.2. System Inputs and Outputs

For home care, device inputs and outputs take the form of triggers and actions with the following arguments:

device_in(message_type, entity_name, entity_instance, message_qualifier, parameter_values)
device_out(message_type, entity_name, entity_instance, message_qualifier, parameter_values)

Only the first argument here is mandatory, the others being optional (empty or omitted). The *message_type* indicates the kind of input or output (e.g. a heart rate reading or a spoken reminder). The *entity_name* indicates the class of entity involved (e.g. a cardiac sensor or a reminder subsystem). If there is only one such entity (e.g. for reminders), this is sufficient to identify it. However, *entity_instance* is often also needed to indicate the particular instance (e.g. the cardiac sensor for *John*). The *message_qualifier* can be used for an associated time period or confidence. For example, the value 10:00 indicates that an input reading applies to the previous ten minutes, or that an output action should be performed in ten minutes. A confidence value (0 to 1) can also be used as an input or output qualifier. Finally, *parameter_values* carries parameters for the message (e.g. a heart rate on input or a reminder message on output).

Besides device inputs/outputs, home care uses common APPEL features such as variable manipulation, expressions, timer handling and message exchange.

3.3. Policy Variables

The policy system maintains a number of variables: values defined by users (or policies), and system variables of various kinds. A controlled system variable represents something that the policy system can manage (e.g. taking medication on time or eating meals regularly). An uncontrolled system variable cannot be

managed by the policy system (e.g. the user's blood oxygen level or the pollen count). A derived system variable is defined by a formula in terms of other system variables (e.g. a measure of sleeplessness).

When a device input arrives with a parameter value, this is automatically stored in the corresponding system variable (e.g. *heart_rate*). In general, variables can be used in expressions or as arguments of rule elements.

3.4. Regular Policies

A regular policy is a normal rule like the ones found in many policy systems. As an example from home care, the following policy meets a common requirement. If an older person needs to go to the toilet at night, there is a risk that they will fall in the darkness. When an occupancy sensor reports that the user gets out of bed at night (11PM to 7AM), the toilet light is switched on. When the bed becomes occupied again, the toilet light is switched off. The two policy rules are tried in sequence, checking first for an unoccupied bed and then for an occupied bed.

policy night light
preference must
when the bed reports it is unoccupied
if the time is 11PM to 7AM
do turn on the toilet light
failing that
when the bed reports it is occupied
if the time is 11PM to 7AM
do turn off the toilet light

Policies can optionally have a preference from *must* (strongest) to *should* (middling) to *prefer* (weakest); negative forms of these can be used to express prohibition. In the event of a policy conflict, the preference is one way of resolving this.

Although rules can be defined from scratch, the system is provided with a library of pre-defined templates for ease of use. The user may just need to select a template, but a few key values may have to be defined (e.g. an emergency telephone number or the user's normal bedtime).

3.5. Goals

Goals are similar in structure to policies but do not make use of a trigger as they are always available. An optional condition specifies the circumstances in which the goal applies. Only two types of actions are used: to maximise or to minimise some measure of a goal, defined in terms of system variables.

Policies are lower-level, system-oriented and executable. In contrast, goals are higher-level, user-oriented and declarative. Goals are therefore not directly executable. In order to assess the extent to which a goal is being achieved, it is associated with a goal measure. This depends on system variables that assess goal achievement.

Sometimes there is an ideal value of a system variable (e.g. indoor temperature), so deviations from this are optimised. There may also be a threshold below which the value of a system variable does not matter (e.g. an audio volume), so only values which exceed the threshold need to be optimised.

As an example from home care, suppose the user has the goal of staying comfortable. This is high-level and not executable. To give the goal meaning, the user has to specify the factors that contribute to the goal (i.e. to its measure). The following indicates that being comfortable depends on the indoor temperature (ideally 21°C), the audio volume (significant only above a threshold of 80dB), and the risk of getting a chill.

goal be comfortable
do maximise indoor temperature with ideal 21°C **and**
minimise audio volume with threshold 80dB **and**
minimise chill risk

Each of these factors is internally associated with a weight that is automatically determined by the system so that they make similar contributions. The goal system includes automated sensitivity analysis to check how its behaviour depends on the choice of weights. It has been found in practice [22] that the choice of weights is not critical to goal achievement – the outcome is usually the same even if goal weights vary over a ratio of 10:1.

A user can define multiple goals, each of which has a relative importance assigned by the user. For example, the user may decide that staying comfortable is twice as important as saving energy. (The weights are easily adjusted by sliders in the web-based wizard.) The weighted combination of goal measures constitutes an overall evaluation function to be optimised dynamically by the system.

3.6. Prototype Policies

Goals are achieved by special policies called prototypes. These are like normal policies but indicate how they affect goals. This is done by specifying their effect on the system variables. Simple effects (there are more complex possibilities) set a system variable to some value ('='), increase it ('+=') or decrease it ('-=').

As an example from home care, the following prototype aims to ensure that the house does not become too hot. If the indoor temperature is reported as over 25°C, the air conditioning is set to 90%. This has the effect of reducing the temperature by 4°C and increasing energy consumption by 4kWh. In case of conflict with another policy, the user would only *prefer* this policy to be followed.

prototype ensure the house is not too hot
preference prefer
when the indoor temperature is reported as over 25C
do set the air conditioning to 90%
effect indoor temperature -= 4°C **and**
energy consumption += 4kWh

When goals and prototype policies are defined, they are statically related through the system variables they involve. This causes prototype policies to be instantiated as special goal-related policies. These are like regular policies but they indicate which goals they contribute to. When prototypes are instantiated, they inherit the conditions of goals they contribute to. This ensures that prototypes apply only when their goals do.

When triggers occur, they cause regular and goal-related policies to be activated. Goal-related policies are selected to maximise the overall evaluation function, and are then combined with regular policies. This allows the system to react appropriately and dynamically to changing circumstances.

3.7. Resolution Policies

Goals can conflict with each other (e.g. staying comfortable might contradict saving energy). Policies can also conflict with each other (e.g. the actions of different policies may simultaneously wish to issue a medication reminder but also to cancel reminders). To handle this situation, special resolution policies are used. These are rather specialised and can be defined only by the system administrator (most likely being selected from the system library). Resolutions define what conflicts are detected, and also state how to resolve them. This means that conflict handling is not hard-wired in the policy system, so it can be adjusted according to the user's preferences and circumstances.

The basis for policy conflict handling is described in [4], though the approach has subsequently been considerably extended. A resolution policy may deal with just one specific conflict, but more typically addresses all conflicts of a certain class (e.g. related to some system variable or device). Although resolution policies may be defined manually, they can be determined largely automatically with limited need for human judgement [6,20]. By definition, resolution policies state what conflicts are. A situation that does not match a resolution policy is therefore not a conflict. This means that completeness is important, hence the desirability of the automated approach.

Resolution policies are defined in advance for each application domain, though they can be adjusted for individual needs. This is necessary because the nature of conflicts and their resolution can be subjective. For example duplicated reminders to take medication might or might not be a nuisance, or it may be debatable whether to allow the heating to be turned on while opening the windows for ventilation.

The triggers of resolution policies are policy actions. Conditions can also be imposed on resolution policies. The actions of resolution policies are either generic or specific. A generic action chooses one of the conflicting policies, e.g. selecting the more confident one or the one with the stronger preference. A specific action is a regular policy action.

The following resolution deals with actions for a device that accepts *off*, *on* and *set* requests. Suppose that the message types differ but the entity names and instances are the same. This means that different actions are being requested of the same thing (e.g. the alarm is to be turned off and set to some volume level at the same time). In such a case the action with the stronger preference is chosen.

resolution level conflict
when there are two device outputs
if the message types are off, on or set **and**
the message types differ **and**
the entity names and entity instances are equal
do choose the policy with the stronger preference

4. Loose Goals and Policies

This section describes the extensions made for support of fuzzy and probabilistic values.

4.1. Confidence Values

Accommodating uncertain values in APPEL has had a significant affect on the language and its implementation. In place of truth values, confidences are used in all conditions. In fact there are three kinds of confidence: boolean (the usual true and false), fuzzy (membership μ of some fuzzy set) and probability (*pr*). All three kinds of confidence are measured on a scale from 0 (false, no membership, impossible) to 1 (true, full membership, certain).

The operators **and**, **or** and **not** behave as usual for boolean confidences. For fuzzy confidences, they have the following definitions:

μ(A **and** B) = min(μ(A), μ(B))
μ(A **or** B) = max(μ(A), μ(B))
μ(**not**(A)) = 1 - μ(A)

In fact the literature has several interpretations of these operators for fuzzy sets, but the above are the commonest ones. For probabilistic confidences, the operators have the following definitions:

pr(A **and** B) = pr(A) $\times$ pr(B)
pr(A **or** B) = pr(A) + pr(B) - pr(A) $\times$ pr(B)
pr(**not**(A)) = 1 - pr(A)

In fact the first two rules are valid only if *A* and *B* are independent. Fortunately this is a reasonable assumption for policies because they almost invariably deal with independent triggers and conditions. Only fairly pathological policies would violate this assumption, e.g.:

when the front door opens **and** the front door sensor fires
if the indoor temperature is above 15°C **and**
the indoor temperature is above 20°C

The APPEL comparison operators are **eq**, **ne**, **lt**, **le**, **gt**, **ge**, **in** and **out**. The latter two are used for inclusion or exclusion of a value from a list (e.g. 'day **in** 6, 7', 'time **in** 11PM..7AM'). These operators have a straightforward interpretation for boolean and probabilistic confidences. However, only the **is** operator is used for comparison in fuzzy logic (corresponding to **eq** in APPEL). For consistency across all kinds of values, the APPEL operators have been given meanings for fuzzy values as well. As an example 10 is less than *cool* in figure 4 because it is less than all values in this set.

4.2. The Role of Domain Ontologies

As noted in figure 2, an ontology server provides domain-specific knowledge to several components of the policy system. Domain ontologies are organised into three hierarchical levels:

- a base ontology contains information about policies in general, e.g. what triggers, conditions and actions are and how they relate
- a wizard ontology then adds user interface information, e.g. how much of the policy language to expose to a novice user, or what the choices are for some policy element
- a variety of domain-specific ontologies then add knowledge about particular application domains, e.g. the devices used in home care and their characteristics.

In the context of this paper, the home care domain ontology defines information about key system variables (e.g. air quality or medication level) and their fuzzy values (e.g. poor or low). For automated conflict analysis, the ontologies also define the underlying causes of conflict. These causes are used to determine in advance what conflicts are likely.

The domain ontologies are stored per home. This means that they can be adjusted to suit individual circumstances (e.g. adjusting the level of risk for a particular user). The ontologies are defined using OWL (Web Ontology Language [29]) and maintained using the Protégé visual editor (*protege.stanford.edu*). Ontology tuning is a task for the system installer using information from a care assessment with the user.

4.3. Fuzzy Values

When numerical information is used in triggers, conditions and actions, the values can be the names of fuzzy sets. These are defined in the ontology for each domain since they vary according to the application.

The fuzzy distributions supported are shown in figure 3; fuzzy set membership μ is on a scale from 0 to 1. Most of these distributions are common in fuzzy logic systems. The *fall* distribution notionally extends all the way to the left, but in practice is given a lower bound. The *rise* distribution is similarly bounded in practice. The *single* distribution has just one value.

As an example, the home care ontology defines the fuzzy sets shown in figure 4 for outdoor temperature. (The fuzzy sets for indoor temperature are not identical as they are perceived differently.)

Fuzzy input sets are normally given names of the form <*entity_instance*>_<*entity_name*> so they are linked to particular entities. For example, *cool* can have different meanings for *indoor_temperature* and *outdoor_temperature*. However, fuzzy input sets may just be named after <*entity_name*> if they are generic. Thus *dry* has the same definition for both indoor and outdoor *humidity*.

When a trigger value is matched against a fuzzy input set, the crisp trigger value is fuzzified. The resultant triggering confidence is therefore fuzzy. Where a condition has a pair of values, one may name a fuzzy set so that the comparison results in a fuzzy value.

As examples from home care, the following device inputs are fuzzy triggers:

device_in(reading,temperature,indoor,,low)
device_in(reading,blood_pressure,John,,high)

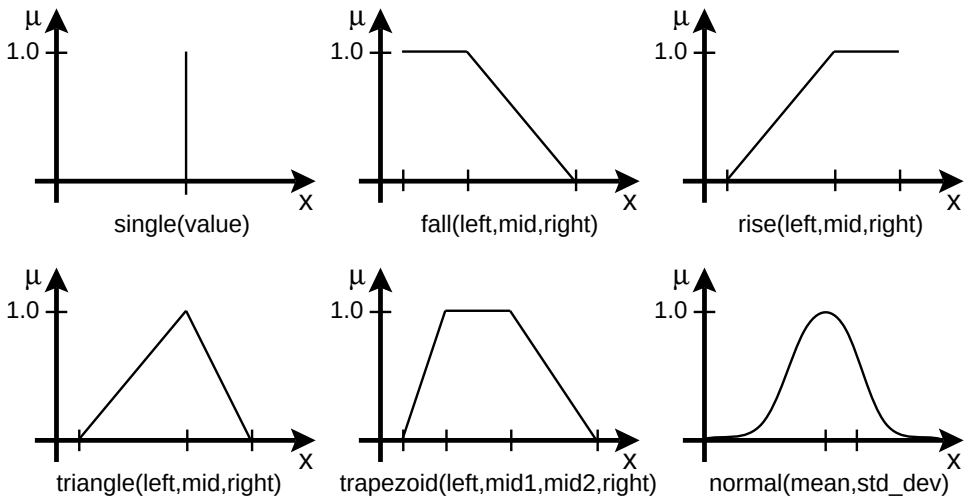

Fig. 3. Fuzzy Distributions

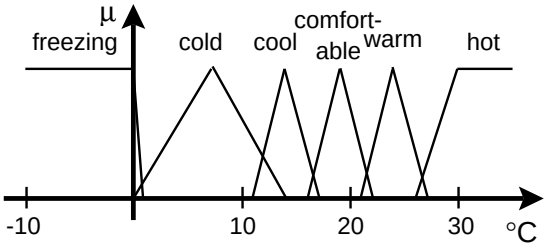

Fig. 4. Fuzzy Outdoor Temperatures

Fuzzy output sets are associated with <*entity_name*> or <*entity_instance*>_<*entity_name*>. However, the former is more common as actions are usually generic across all entity instances (e.g. various *heating* actions).

As examples from home care, the following device outputs are fuzzy actions:

device_out(set,heating,,,very_high)
device_out(set,volume,hall,,low)

Figure 5 shows a few examples of how fuzzy sets are defined for home care, using the terminology of figure 3. As is usual in this kind of approach, the fuzzy sets overlap to some extent. If necessary the definitions can be tuned according to the user's preferences (e.g. what 'bright' or 'cold' means). This would be established through a care assessment. However, the nature of fuzzy logic means that precise definition of fuzzy sets is not critical.

4.4. Probabilistic Values

When numerical information is used in trigger, conditions and actions, the values can be associated with probabilities. For a trigger, the message qualifier can provide a *single* probability. For a trigger or a condition, a value can be associated with a *normal* distribution for some mean and standard deviation.

The policy system can receive probabilistic values in triggers, or can define them in actions. Although probabilities are handled by the policy system, they are interpreted at device level. For example a temperature sensor will have a defined accuracy, so its temperature readings will be reported with some distribution. A speech recogniser will provide a probabilistic confidence when it interprets possibilities for what the user said. Probability can also influence device output. For example a reminder subsystem might emphasise a high-confidence reminder, or might repeat one that is not acknowledged by the user.

Factor	Fuzzy Name	Distribution
air quality (0..10)	poor	fall(0,2,4)
	acceptable	triangle(3,5,7)
	good	rise(6,9,10)
lighting (% of full)	very_dim	fall(0,20,30)
	dim	triangle(20,30,40)
	moderate	triangle(30,50,70)
	bright	triangle(60,70,80)
	very_bright	rise(70,80,100)
outdoor temperature (°C)	freezing	fall(-10,0,1)
	cold	triangle(0,7,14)
	cool	triangle(11,14,17)
	comfortable	triangle(16,19,22)
	warm	triangle(21,24,27)
	hot	rise(26,30,35)

Fig. 5. Home Care Examples of Fuzzy Sets

As examples from home care, the following device inputs are probabilistic triggers (with confidence 0.8 and a normally distributed percentage respectively):

device_in(active,movement,hall,,0.8)
device_in(forecast,frost,,,normal(70,20))

Besides trigger parameters, variables can also have probabilistic values. A system variable can be given a probabilistic value implicitly through a trigger. A probabilistic value may also be assigned explicitly to a variable. When a probabilistic trigger is matched, the resultant triggering confidence is probabilistic. Where a condition has a pair of values, one may be probabilistic and so result in a probabilistic confidence.

Actions can also be probabilistic. As examples from the home care domain, the following device outputs are probabilistic actions (with confidence 0.7 and 0.4 respectively):

device_out(set,heating,,0.7,80)
device_out(report,alert,,0.4,User may have fallen)

This may be used to indicate the confidence with which an action should be performed, and hence how the action is realised.

4.5. Combining Fuzzy and Probabilistic Information

The confidences associated with multiple triggers or conditions can be combined with the usual operators. Using **and** and **or**, boolean confidences can be combined with all-fuzzy or all-probabilistic confidences; the result is fuzzy or probabilistic. However, fuzzy and probabilistic confidences cannot be directly combined as they are conceptually different. If this situation arises, the mean of the probability is used as a definite value. This avoids trying to compare a fuzzy confidence with a probabilistic confidence.

The overall confidence in a policy trigger is combined with the overall confidence in its condition. This uses the **and** operator to obtain an overall confidence for the policy being activated. This confidence is then associated with the policy's actions. To avoid policies being activated by unlikely circumstances, the activation confidence must exceed a small threshold before it is considered to be relevant.

The equivalent of conflict handling for fuzzy actions is the usual procedure of fuzzy logic: accumulation and defuzzification. If a number of fuzzy actions refer to the same entity and fuzzy output set, a single membership for that fuzzy output is determined by the Root Sum Square method. This calculates the square root of the squares for all contributing membership values.

Fuzzy actions for the same device are then combined using the Centre of Gravity method, resulting in a single definite action. This means that the centroid value for each fuzzy output set is weighted by its membership.

In home care, for example, fuzzy actions might simultaneously request that the heating be set to *low* and also to *moderate*. Based on the definitions of the fuzzy output sets, a centre of gravity is calculated for the combined fuzzy actions. This will result in a crisp value for the heating level being set.

Probabilistic actions are not combined like fuzzy ones: they are analysed for conflicts as usual. If two incompatible probabilistic actions are found, a resolution policy can choose the more probable action, say. Suppose one policy wishes to sound an alarm with

probability 0.8 and another wishes to cancel the alarm with probability 0.4. Choosing the more probable action would be an obvious strategy and is, in fact, the default resolution.

5. Home Care Management

This section presents an extended worked example that shows how goals and policies are used in practice. Following the argument in section 1.3 about the breadth of home care, this example involves a range of factors. The definitions are a subset of those used in an actual home environment. Although it is possible to create these from scratch, they are normally taken from the system library (possibly needing some values to be filled in such as an emergency number).

5.1. Definitions

5.1.1. Goals

For home care, the controlled system variables include additive intake (g/day), audio volume (dB), awake time (hours), chill risk (0..10), energy consumption (kWh), indoor temperature (°C), pollen level (0..10), security level (0..10), social contact (hours), and TV viewing (hours). The uncontrolled system variables include outdoor temperature (°C).

Sample goals are shown in table 1, drawn from a larger set that deals with a number of other factors. The weights chosen by the user for each goal are shown.

5.1.2. Prototype Policies

Sample prototype policies are shown in table 2. When the policy system is managing a device, the policies have a direct effect. However when people are involved, the policy actions have only an indirect effect. For example, prototype 2 merely asks rather than requires the user to go for a walk. Actions are also high-level and do not imply a particular interaction modality (e.g. speaking a reminder as opposed to displaying it). Thus in prototype 2, user preferences will determine whether asking the user is achieved through a synthesised voice message, a display message, a text message, etc.

5.1.3. Regular Policies

Table 3 shows example regular policies. Normally, prototype policies are preferred as they can be selected based on their contribution to goals. However, regular policies are useful as a 'backbone': they are always available irrespective of the current goals. Thus regular policy 1 ensures that the house can never get too cold, while regular policies 2 and 3 ensure that action is always taken on elevated blood pressure.

No.	Definition	Weight
1	**goal** be secure **do** maximise security level	1.0
2	**goal** be social **do** maximise social contact	0.5
3	**goal** be active **if** it is a weekday **do** maximise awake time **and** minimise TV viewing **and** maximise social contact	1.0
4	**goal** avoid allergens **do** minimise pollen risk **and** minimise additive intake	0.5
5	**goal** use less energy **do** minimise energy consumption	1.5
6	**goal** be comfortable **do** maximise indoor temperature with ideal 21°C **and** minimise audio volume with threshold 80dB **and** minimise chill risk	2.5

Table 1
Sample Goals

5.1.4. Resolution Policies

Table 4 shows example resolution policies. Resolution 1 detects and resolves situations where policies wish to execute conflicting power level actions. Resolution 2 deals with different alerts being output to the same recipient. If this happens, the default resolution is applied: choose the policy with the stronger preference, otherwise the one with the firmer confidence, or failing that the newer one.

5.2. Goal and Policy Realisation

As goals and prototypes are defined, the prototypes that contribute to goals are statically determined. Table 5 shows the results of statically analysing prototypes against goals. This creates a goal-derived policy for each prototype, stating which goals each such policy contributes to.

A substantial number of things happen when a trigger occurs. Policies are activated, and goal-related policies are separated from regular ones. The goal-related policies are then optimised and combined with

No.	Definition
1	**prototype** cool house naturally **preference** should **when** the indoor temperature is reported as hot **if** the outdoor temperature is less than warm **do** turn off the air conditioning **and** open the windows for 1 hour **effect** indoor temperature -= 5°C**and** security level -= 1 **and** pollen risk += 0.5
2	**prototype** encourage walk when cold **if** the time is 5PM **and** the user has stayed in 2PM-5PM **and** the outdoor temperature is cold **do** ask the user to go for a walk **effect** social contact += 1hr **and** chill risk += 2
3	**prototype** ensure daily phone call **if** the time is 5PM **and** the user has had no phone calls 9AM-5PM **do** ask a friend to phone in evening **effect** social contact += 1hr
4	**prototype** ensure daily contact **if** the time is 5PM **and** the user has stayed in 9AM-5PM **do** ask a neighbour to drop by in the evening **effect** social contact += 2hr
5	**prototype** ensure house is not too cold **when** the indoor temperature is reported as cool **if** a window is open **do** set the heating to low for 1 hour **and** close the windows **effect** indoor temperature += 5°C **and** energy consumption += 3kWh **and** security level += 3
6	**prototype** ensure the house is not too hot **preference** prefer **when** the indoor temperature is reported as hot **do** set the air conditioning to high for 1 hour **effect** indoor temperature -= 4°C **and** energy consumption += 4kWh

Table 2
Sample Prototype Policies

the regular ones. Policy actions are extracted, and conflicts are detected and resolved. Finally, the optimal and conflict-free actions are performed.

No.	Definition
1	**policy** avoid cold house **when** the indoor temperature is reported **if** the indoor temperature is at best cold **do** set the heating to moderate
2	**policy** check moderate blood pressure **when** systolic blood pressure is reported **if** systolic blood pressure is moderate **do** ask the user to rest
3	**policy** check high blood pressure **when** systolic blood pressure is reported **if** systolic blood pressure is high **do** ask the user to seek medical help

Table 3
Sample Regular Policies

No.	Definition
1	**resolution** level conflict **when** there are two device outputs **if** the message types are off, on or set **and** the message types differ **and** the entity names and entity instances are equal **do** choose the action with the stronger preference
2	**resolution** alert conflict **when** there are two alert outputs **if** the recipients are the same **and** the alerts differ **do** choose the default resolution

Table 4
Sample Resolution Policies

5.3. Execution Scenarios

The following describes realistic scenarios that illustrate how goals and policies are handled at run-time.

5.3.1. Scenario 1

This scenario illustrates the selection of policies to optimise goals. Suppose there is a clock trigger because the time is now 5PM. Also suppose that the indoor temperature is 17°C, the outdoor temperature is 10°C, the user has stayed in all day and has received no phone calls, and a window is open.

The policy server determines that the activated policies are instantiated prototype 2 (fuzzy confidence 0.57), prototype 3 (boolean confidence 1.0) and prototype 4 (boolean confidence 1.0). These are passed to the dynamic analyser, which finds that prototypes 3

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

Table 5
Prototype Effects on Goals

and 4 are the optimal choice in the current circumstances.

The actions of these prototypes are extracted (contacting a friend and a neighbour) and are checked for conflicts, though in this case there are none. Finally, the actions are performed: a friend is asked to phone and a neighbour is asked to drop by.

5.3.2. Scenario 2

This scenario illustrates the use of fuzzy logic to determine policy actions. Suppose there is a temperature trigger because the interior is now 13°C. Also suppose that a window is open.

The policy server determines that the activated policies are instantiated prototype 5 (fuzzy confidence 0.33) and regular policy 1 (fuzzy confidence 0.4). These are passed to the dynamic analyser, which separates goal-related policies from regular policies. As only prototype 5 is goal-related, this is considered for optimisation. Even if only one policy is activated, it can still be rejected by the optimiser if it makes things worse. However, prototype 5 is in fact selected by the optimisation. The optimised and regular policies and then combined.

When the policy actions are extracted, it is found that both policies have fuzzy actions for the same entity (set the heating to *low* in one case and to *moderate* in the other). The two fuzzy outputs are therefore accumulated, resulting in a single crisp action with heating level 34%. As there is only one action there is no need to check for conflicts. The final step is to execute this action, setting the heating to this value.

5.3.3. Scenario 3

This scenario illustrates how resolutions deal with a conflict. Suppose there is a temperature trigger because the interior is now 30°C. Also suppose the outdoor temperature is 10°C.

The policy server determines that the activated policies are instantiated prototype 1 (fuzzy confidence 1.0) and prototype 6 (fuzzy confidence 1.0). These are passed to the optimiser, which selects both policies since they combine to improve the current situation.

The resulting actions are all definite (turn off the air conditioning, open the windows, set the air conditioning to high). When these are scanned for conflicts, resolution 1 detects a problem with the first and third actions (*off* vs. *set*). The resolution is to apply actions from the policy with the stronger preference: here, prototype 1 (with the stronger *should*). The final step is to execute its actions, turning off the air conditioning and open the windows.

5.3.4. Scenario 4

This scenario illustrates the use of probabilities and default conflict resolution. Suppose there is a blood pressure trigger because the user's systolic pressure is now 138 ± 3 mmHg.

The policy server determines that the activated policies are regular policy 2 (probabilistic confidence 0.68) and regular policy 3 (probabilistic confidence 0.33). These probabilities carry over to the corresponding actions.

The resulting actions are of the same kind (asking the user to do something), so they are handled by resolution 2. This asks for the default resolution. Since neither policy has a preference, this chooses the firmer confidence (i.e. the more probable action of policy 2). The final step is to execute its action, asking the user to rest. As this is of moderate confidence (0.68), the system will simply speak the message to the user.

6. Conclusion

6.1. Summary

It has been argued that goals and policies offer flexibility and adaptability in system management. They also open the system behaviour to inspection and to change. This is particularly useful in home care systems, which are typically hard to modify without specialised expertise. Home care also requires a high degree of customisation and adaptation to user needs.

However, for applications such as home care a conventional rule-based approach is less appropriate. System inputs may be imprecise, and it may be difficult to formulate precise rules. Examples of both aspects have been given for home care.

The high-level architecture of the ACCENT system has been explained. The APPEL language it sup-

ports has been introduced. The approach has been enhanced by new extensions to support probabilistic inputs/outputs and the use of fuzzy rules. This now allows home care to be managed by loose goals and policies.

An extended example has been given, illustrating the new kinds of goals and policies that are useful in home care. This has shown the use of fuzzy and probabilistic values, and how these are handled by the main system elements.

6.2. User Evaluation

Usability of various ACCENT aspects has been evaluated in separate work:

- A detailed usability evaluation was previously carried out of the original ACCENT work on home care [19]. It has also been shown that a wide variety of users appreciate a policy-based approach and the power that it confers [12]. This demonstrates that ordinary users are able to relate to policies and to formulate them successfully.
- A separate evaluation was undertaken of the usability of goals. A group of care managers was recruited as the most likely kind of user to define home care goals. A short scenario was provided to describe the situation of a hypothetical older couple. The participants were then asked to formulate home care goals for this couple, and to define policies that could be used to realise these. All participants were able to come up with plausible goals for home care. Perhaps more surprisingly (because it is a more technically challenging task), all participants successfully thought of policies that could be used to achieve the goals.
- A further study has examined whether ACCENT is able to satisfactorily explain its operation. Again, a number of care managers were recruited to participate as these are likely to be the major system users. The participants were given individual training in the basics of defining policies and in how to check the operation of policies. The participants were then asked to use the system to answer a variety of questions about policies. This was achieved with 94% accuracy.

For the new work reported in this paper, an evaluation has been carried out into whether users prefer to use the looser formulation that is now possible. A group of 14 participants was recruited. This comprised a mixture of informal carers and older people, so the age distribution was bimodal (average age 31 years and 57 years in each category). There were equal numbers of men and women, all with at least a basic ability to use computers.

A quantitative analysis was performed of preferences as to loose or precise formulation of policies. This was done through a written survey that started with a basic explanation of policies. Participants then evaluated 17 pairs of alternative policies. 12 of the examples used fuzzy and precise alternatives, while the other 5 used probabilistic and precise alternatives. The order and nature of the alternatives was intentionally mixed up in the survey so as not to bias the results.

The participants were asked to indicate which alternative they found easier to understand and to be more appropriate for home care. Overall, the loose formulation was preferred in 77% of cases. The following policy extracts illustrate where the loose formulation was strongly preferred:

- ‘the air quality is poor’ vs. ‘the air quality is 2’: it is understandable that 92% preferred the fuzzy form as users may well be unfamiliar with the numerical pollution scale
- ‘the pollen forecast is high’ vs. ‘the pollen forecast is 7’: for a similar reason, 85% preferred the fuzzy form
- ‘set the lighting to moderate’ vs. ‘set the lighting to 40%’: it is understandable that 92% preferred the fuzzy formulation as users are unlikely to think of light levels numerically.

Interestingly, there were two cases where the preference for a loose formulation was not strong:

- ‘the user’s TV viewing today has been high’ vs. ‘the user’s TV viewing today has been over 5 hours’: opinions were equally split on this, presumably because users can relate to a certain number of hours watching TV
- ‘the chance of rain is moderate’ vs. ‘the chance of rain is 50%’: surprisingly only 42% preferred the imprecise form, perhaps because people are accustomed to weather forecasts with a probability of precipitation or they can relate to a 50:50 chance.

There is therefore evidence that users would generally prefer to use a loose formulation. However, both loose and precise formulations can be used: there is no obligation to use one or the other. Users can thus choose to write policies in the way that is most under-

standable for them. The new work has therefore added extra flexibility and freedom of expression.

6.3. Discussion

The author is unaware of any previous attempt to combine conventional policy-based management with both fuzzy logic and probability. Although there is a little related work on network quality of service, the application to managing home care is novel.

The resulting approach offers greater flexibility, but more importantly is practical and realistic. The support of fuzziness means that system rules can be formulated in a more human-meaningful way, without the unnecessary precision required by previous work. The support of probabilities means that the system no longer has to rely on precise sensor inputs.

The operation of goals and policies imposes only a low computational load. The entire procedure for selecting policies, optimising goals, resolving conflicts, and dictating actions takes about one second in a typical home. In fact, policy-related events in a home care system are relatively infrequent. The processing overhead is therefore acceptable – especially given the flexibility and control that goals and policies offer. However, there are ways to reduce this overhead such as caching goal analysis results.

The approach scales satisfactorily. Because of the way that policies are indexed and retrieved, there is little penalty in dealing with hundreds of policies. In fact the total number of policies is not relevant – what matters is how many are triggered at one time (which is typically up to half a dozen). The goal optimisation phase is currently less efficient, but runs acceptably fast in a typical home setting (with, say, less than 15 goals). However, better optimisation strategies would help if large numbers of goals had to be optimised.

Operating system and memory demands are low. Being written in Java, ACCENT runs on several operating systems. (Home care device drivers for these systems are also readily available.) The ACCENT system typically requires up to 90Mb of memory. It has been demonstrated to run happily on desktop systems as well as on credit card-sized computers (Raspberry Pi, *www.raspberrypi.org*).

One limitation of the work is that support of fuzziness is limited, though it takes a mainstream approach. Fuzzy logic systems typically offer a wider range of capabilities than ACCENT. For example, FCL (Fuzzy Control Language [10]) allows for a greater variety of operators and techniques. However, ACCENT is not intended as a general-purpose fuzzy logic system. Rather it seeks to gain by integrating fuzzy logic capabilities with policy-based management.

Another limitation is that probabilities are currently just single values or belong to normal distributions. This is realistic for typical sensors and services. However, there are opportunities to support probabilities more comprehensively (e.g. other kinds of distributions).

At the present time, the new ACCENT capabilities reported in this paper are currently undergoing robustness testing in a home care lab and in a user's home. Once confidence has been built in the extensions they will be deployed for evaluation more widely.

Acknowledgements

This work builds on foundations developed during the MATCH project (Mobilising Advanced Technologies for Care at Home), supported by the Scottish Funding Council under grant HR04016. Substantial contributions to the original ACCENT system were made by Stephan Reiff-Marganiec, Lynne Blair, Gavin A. Campbell, Feng Wang and Ross Mills (University of Stirling). The author is grateful to Erfu Yang (University of Stirling) for advice on using fuzzy logic.

References

[1] Y. Bai, H. Zhuang, and D. Wang, editors. *Advanced Fuzzy Logic Technologies in Industrial Applications*. Springer, Berlin, Germany, Sept. 2006.

[2] 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.

[3] J. E. Bardram. The Java context awareness framework – A service infrastructure and programming framework for context-aware applications. In H. W. Gellersen, R. Want, and A. Schmidt, editors, *Proc. 3rd Int. Conf. on Pervasive Computing*, number 3468 in Lecture Notes in Computer Science, pages 98–115. Springer, Berlin, Germany, May 2004.

[4] L. Blair and K. J. Turner. Handling policy conflicts in call control. In S. Reiff-Marganiec and M. D. Ryan, editors, *Proc. 8th Int. Conf. on Feature Interactions in Telecommunications and Software Systems*, pages 39–57. IOS Press, Amsterdam, Netherlands, June 2005.

[5] G. A. Campbell and K. J. Turner. Ontologies to support call control policies. In N. Meghanathan, D. Collange, and Y. Takasaki, editors, *Proc. 3rd Advanced Int. Conf. on Telecommunications*, pages 5.1–5.6. IEEE Computer Society, Los Alamitos, California, USA, May 2007.

[6] G. A. Campbell and K. J. Turner. Policy conflict filtering for call control. In L. du Bousquet and J.-L. Richier, editors, *Proc. 9th Int. Conf. on Feature Interactions in Software and Communications Systems*, pages 83–98. IOS Press, Amsterdam, Netherlands, May 2008.

[7] N. Damianou, E. C. Lupu, and M. Sloman. The Ponder policy specification language. In *Policy Workshop 2001*, number 1995 in Lecture Notes in Computer Science. Springer, Berlin, Germany, Jan. 2001.

[8] T. Dursun and B. Örencik. Police: A novel policy framework. In *Proc. ISCIS 2003*, number 2869 in Lecture Notes in Computer Science, pages 819–827. Springer, Berlin, Germany, 2003.

[9] M. P. Fernandez, A. de Castro, P. Pedroza, and J. F. de Rezende. QoS provisioning across a DiffServ domain using policy-based management. In F. R. Chang and A. Henley, editors, *Proc. Global Telecommunications Conference*, pages 2220–2224. Institution of Electrical and Electronic Engineers Press, New York, USA, Nov. 2001.

[10] IEC. *Programmable Controllers – Part 7: Fuzzy Control Programming*. IEC 61131-7. Int. Electrotechnical Commission, Geneva, Switzerland, 2001.

[11] C. Leong, A. R. Ramli, and T. Perumal. A rule-based framework for heterogeneous subsystems management in smart home environment. *IEEE Transactions on Consumer Electronics*, 55(3):1208–1213, Aug. 2009.

[12] C. Maternaghan. Can people program their home? Technical Report CSM-191, Computing Science and Mathematics, University of Stirling, UK, Apr. 2012.

[13] C. Maternaghan and K. J. Turner. Pervasive computing for home automation and telecare. In S. I. A. Shah, M. Ilyas, and H. T. Mouftah, editors, *Pervasive Communications Handbook*, pages 17.1–17.25. CRC Press, Boca Raton, Florida, USA, Nov. 2011.

[14] M. Román, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. Gaia: A middleware infrastructure for active spaces. *Pervasive Computing*, 1(4):74–83, Oct. 2001.

[15] 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.

[16] Select Committee on Economic Affairs. *Aspects of The Economics of An Ageing Population*. The Stationery Office Limited, London, Nov. 2003.

[17] L. S. Shafti, P. A. Haya, M. García-Herranz, and E. Perez. Inferring ECA-based rules for ambient intelligence using evolutionary feature interaction. *Ambient Intelligence and Smart Environments*, 5(6):563–587, Dec. 2013.

[18] K. J. Turner. Device services for the home. In K. Drira, A. H. Kacem, and M. Jmaiel, editors, *Proc. 10th Int. Conf. on New Technologies for Distributed Systems*, pages 41–48. IEEE Computer Society, Los Alamitos, California, USA, May 2010.

[19] K. J. Turner. Flexible management of smart homes. *Ambient Intelligence and Smart Environments*, 3(2):83–110, May 2011.

[20] K. J. Turner. Handling conflict-prone policies in multiple domains. Technical Report CSM-199, Computing Science and Mathematics, University of Stirling, UK, July 2014.

[21] K. J. Turner. The ACCENT policy system for home care. Technical Report CSM-188, Computing Science and Mathematics, University of Stirling, UK, Apr. 2014.

[22] K. J. Turner and G. A. Campbell. Goals and conflicts in telephony. In M. Nakamura and S. Reiff-Marganiec, editors, *Proc. 10th Int. Conf. on Feature Interactions in Software and Communications Systems*, pages 3–18. IOS Press, Amsterdam, Netherlands, June 2009.

[23] K. J. Turner, G. A. Campbell, and F. Wang. Goals and policies for home care. In K. J. Turner, editor, *Advances in Home Care Technologies: Results of The MATCH Project*, pages 3–49. IOS Press, Amsterdam, Netherlands, Oct. 2012.

[24] K. J. Turner, S. Reiff-Marganiec, L. Blair, G. A. Campbell, and F. Wang. APPEL: Adaptable and Programmable Policy Environment and Language. Technical Report CSM-161, Computing Science and Mathematics, University of Stirling, UK, Apr. 2014.

[25] 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.

[26] 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.

[27] VoiceXML Forum. *Voice eXtensible Markup Language*. VoiceXML Version 2.1. VoiceXML Forum, Piscataway, New Jersey, USA, May 2010.

[28] O. Wittner and B. E. Helvik. Distributed soft policy enforcement by swarm intelligence: Application to loadsharing and protection. *Annales des Télécommunications*, 59(1-2):10–24, Jan. 2004.

[29] World Wide Web Consortium. *Web Ontology Language (OWL) – Overview*. Version 1.0. World Wide Web Consortium, Geneva, Switzerland, Feb. 2004.

[30] L. A. Zadeh. Is there a need for fuzzy logic? *Information Sciences*, 178(13):2751–2770, July 2008.