Claire Maternaghan and Kenneth J. Turner. A Configurable Telecare System, in Fillia Makedon, Margit Betke, Ilias Maglogiannis and Grammati Pantziou (eds.), Proc. 4th Int. Conf. on Pervasive Technologies related to Assistive Environments, pp. D.25-D.32, ISBN 978-1-4503-0772, ACM Press, New York, USA, May 2011.

A Configurable Telecare System

Claire Maternaghan and Kenneth J Turner Computing Science and Mathematics, University of Stirling, Stirling, FK9 4LA, UK
cma | kjt @cs.stir.ac.uk

ABSTRACT

The Homer system for telecare and home automation is described. Core capabilities are shared between these applications, supplemented by application-specific devices and services. Current home systems do not support simple, yet sophisticated, ways of controlling the home in a generic and high-level way. In contrast, Homer is designed to make it easy for non-technical users to achieve this. Developers create home components that expose their services and functionality in a way that encourages combination. Components are made accessible to end-user applications through an HTTP interface, allowing use of any interface technology. Internally, Homer supports automation through policies that combine the functionalities and services offered by components. These policies can be created using many kinds of user interfaces. The Homer architecture, components, policies and user interfaces are discussed. Finally, the paper concludes with an evaluation of the work in comparison to similar systems.

Categories and Subject Descriptors

D.2.10 [**Sofware Engineering**]: Design; D.2.11 [**Software Engineering**]: Software Architectures

General Terms

Home Automation, Software Engineering, Service Oriented Architecture, Telecare

1. INTRODUCTION

1.1 Background

The world population is gradually ageing [5]. As a result, there is increased pressure in most countries to provide adequate support for older people. Although technology is only part of the solution, telecare (remote support of home care) has been enthusiastically promoted as a way of helping older people to continue living independently in their own homes. Telecare involves some kind of computer-based system in the home that monitors for undesirable situations such as falls, bed wetting or overflowing baths. The home provision is supplemented by a link to a call centre for dealing with alerts and calls for help.

However, telecare technologies are still relatively undeveloped. Commercial systems often do not incorporate the latest research advances. More seriously, telecare systems are usually relatively fixed in function. Where changes are possible, they normally require specialised technical expertise and often reprogramming. As a result, telecare systems can be hard to customise for individual circumstances, and can be hard to adapt as these change over time [16].

Home automation has a longer history going back several decades. However, most approaches are relatively unsophisticated. Indeed, home *control* rather than *automation* would often be a better designation. Much of the commercial effort in this area is concerned with capabilities such as being able to stream audio and video around the home. Although some home systems do offer programmability, this usually requires specialised technical expertise and is aimed more at the hobbyist rather than ordinary householders.

This paper describes Homer – a home system that is designed to meet the needs of both telecare and home automation. Both applications share a common core of capabilities, though they also require specialised devices and services in each application. The study in [8] discovered that users would like the ability to control the home (though they would not wish this to seem like programming). As the target users have very limited technical knowledge, a home system needs to be made easy to use. However, the system also needs to offer more sophisticated capabilities to specialists (e.g. a care professional or a home system installer). Simple tasks must therefore be easy, while complex tasks must be possible.

1.2 Context

This paper touches on many related fields: home automation and smart homes, telecare, component architectures, policy-based management, and end-user programming. As a result, only a high-level overview of related work is practicable here. Section 6 compares the authors' work with the most relevant similar systems.

Home Automation: At device level, several standards have evolved to support home automation. These include infrared (home appliance control), KNX (building management and domestic applications), Lonworks (building management and home automation), University Plug and Play (networked devices) and X10 (mains appliance control). More interesting is packages that aim to offer higher-level control over home devices. These include Control4 (a widely adopted framewok), Cortexa (rule-based, but not flexible or simple enough), Girder (technical knowledge needed to define input-output event mappings), Home Automation Inc. (designed for installers rather than end users), and HomeSeer (particularly focused on control via remote devices). In general, these approaches lack either the sophistication needed for full home automation or the simplicity required by non-technical users.

Telecare: Commercial telecare solutions are available from companies such as Cisco, General Electric, Initial, Intel, OmniQare, Philips and Tunstall. In fact they are often focused on telehealth (remote health monitoring) rather than telecare (which emphasises social care). Current telecare systems are relatively unsophisticated, and generally require specialised installation expertise (especially if they have to be modified). OmniQare is unusual in being a framework for third parties to add telecare services. As telecare is a fairly recent development, standards are still in their infancy. The Continua Health Alliance (*www.continuaalliance.org*) and the European Telecommunications Institute (*www.etsi.org*) are working towards telehealth and telecare standards, but interoperability among different devices and systems is still a long way off.

Component Frameworks: Many component architectures have been developed. In the context of home systems, relevant approaches include Atlas (home sensor/actuator platform), Jini (distributed network architecture), Open Services Gateway initiative (service platform, *www.osgi.org*), Service Component Architecture (implementation-independent component interconnection, *www.osoa.org*), and Service Oriented Device Architecture (device interworking*www.eclipse.org/ohf/components/soda*). Of these, approaches based on Service Oriented Device Architecture have proven particularly popular. OSGi (Open Services Gateway initiative) has also been widely adopted for home systems, e.g. Atlas and the Homer system described in this paper.

Policy-Based Management: Policies are automated rules for controlling systems, having been used in applications such as access control, network or system management, and quality of service. Examples of the many approaches include ACCENT (domain-independentt policies [16]), Drools (business rules, *www.jboss.org/drools*), Police (emphasis on distributed policies) and Ponder (distinctive features such as domains, conflict handling and refinment [2]). Although simple rules are supported by some commercial home automation packages, the richer field of policies applied in the home has not been widely explored (ACHE [9] and [6] being a few examples).

End-User Programming: A number of techniques have been developed to allow end users to program computer-based systems. Programming by demonstration (e.g. a CAPpella [3], Alfred [4]) is liked by users, but it can be tedious or impracticable to demonstrate the range of responses required of the home. Tangible programming (e.g. ACCORD [11], CAMP [13], Media Cubes [1]) allows users to define rules using physical analogies such as jigsaw pieces or 'magnetic' words. Although users find these easy to use, the expressivity of these approaches is necessarily limited. Visual programming (e.g. iCAP [12], OSCAR [10]) allows rules to be defined graphically, but so far the approaches have been limited in application (e.g. to audio-visual devices).

1.3 Overview

Section 2 introduces the architecture and framework of the Homer system. The design of Homer components in general is discussed in section 3 along with specific examples of use in telecare. The policy approach described in section 4 allows the functionality of the home to be easily extended. The structure of policies is explained, and is illustrated with a range of policies relevant to telecare. A flexible approach to user interfaces is presented in section 5. To demonstrate the extensibility of the system, rather different interfaces using the Apple iPhone and iPad are presented. Section 6.1 discusses Homer in the context of related work. Various systems are evaluated against criteria that are desirable for the kind of support needed for telecare.

2. HOME SYSTEM

This section overviews the Homer system for telecare and home automation.

2.1 Overview

It is common for existing home systems to limit the possible technologies, devices, sensors and actuators to those explicitly supported by the system itself. This requires component developers to track changes in the system framework, and to support new devices as they become available. Connect 4 (see section **??**) deals with this by offering a middleware platform that requires third-party developers to write plug-ins for their own devices to ensure compatibility with Connect 4. This means that Connect 4 does not need to take responsibility for device updates or additions.

As discussed in section 3, Homer follows the same philosophy by offering developers the functionality to write plug-ins that expose device capabilities to Homer. Section 5 explains that the same philosophy is followed in end-user interactions with Homer, where developers write applications to expose functionality through any desired user interface.

Where Homer differs from existing work is in the seamless integration of policies (user-defined rules) for using and managing home components. When developers write plug-ins for Homer, they systematically what devices and services can do. These capabilities are then available to developers of end-user applications. Device and service capabilities can be used in policies as described in section 4. An example application developed by the authors offers a custom end-user programming experience on the Apple iPad, allowing end users to manage different functionalities and services in the home.

2.2 Architecture

Figure 1 shows that the Homer architecture comprises three main aspects: components, services and the internal framework.

Components: Within Homer, a 'component' represents devices, sensors, actuators and user services (such as email); Homer components are discussed further in section 3. The system has been developed using OSGi (Open Standards Gateway initiative, *www.osgi.org*) as a Service Oriented Architecture. This allows for a naturally modular system, where components are OSGi bundles (standalone modules) that can be installed, updated and removed dynamically.

Services: Services within Homer are not directly visible to the end user. Some are private (for internal Homer use) while others are public (acting as a library of services for component developers). Developers can write their own services, which can then be used by other components. Each service is an OSGi bundle, and thus benefits from modularity, loose coupling, etc.

Framework: The Homer framework is where the core functionality of the system resides, bringing together components, the home, end users and the interfaces they see. The framework is discussed in more detail in section 2.3.

2.3 Framework

The Homer framework in figure 2 acts as a central bridge between the components and services within the system. The framework

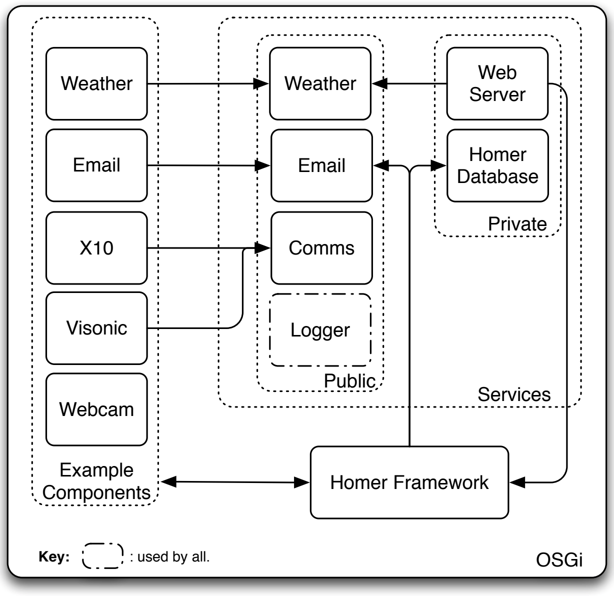


Figure 1: Homer Architecture

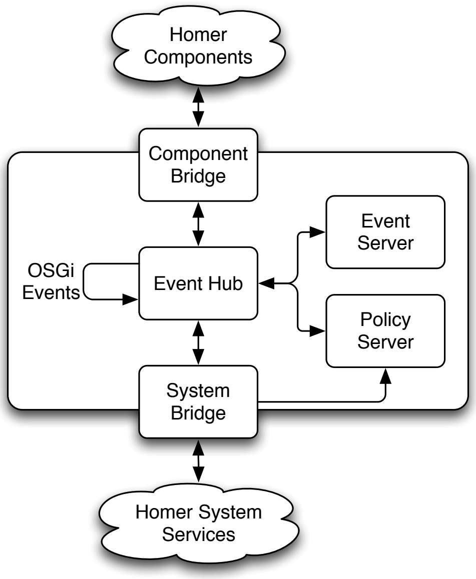


Figure 2: Homer Framework

is implemented as a single OSGi bundle, wrapping the core functionality of the system into one module. The framework offers the following capabilities:

Component Bridge: The component bridge is used for communication between components and Homer. Components register themselves with the bridge, which then manages future communication with components. The bridge is responsible for relaying messages between components and Homer, e.g. requests to turn on a particular device or reporting that a particular sensor was fired. The bridge does not require pre-defined knowledge of particular components, and has no dependencies on them.

System Bridge: The system bridge acts as a gateway between Homer's private services and the framework, e.g. support for the Homer web server and database. The web server supports external end-user applications as discussed in section 5.

Policy Server: As described in section 4, the policy server manages and executes policies within the system.

Event Server: The event server offers more sophisticated sensor and actuator fusion, termed 'device services'. It allows component-level events to be mapped to/from policy-level events through external logic defined by web service orchestration [14].

Event Hub: The event hub supports central communication among all aspects of the framework. This uses OSGi events for flexible exchange of messages among bundles. Event properties are used to filter messages according to what is relevant for each component. As an example, the component bridge is interested only in messages between components and the framework. The component bridge therefore registers a listener for events of type 'action' (more information see sections 3 and 4).

3. HOME COMPONENTS

This section discusses Homer components and gives some examples of these. Homer supports many different devices and services for both telecare and home automation. These can range widely, from vital signs monitoring, to video recording, to Twitter. A home system must be able to accommodate the ever-changing and growing nature of technology and services for the home.

3.1 Component Design

Homer components are lightweight, loosely-coupled modules that can be installed, modified and removed from Homer at run time. This capability is intrinsic to OSGi. A component represents a device or a user service. As simple examples, a medication dispenser provides usage information, a thermostat can check the room temperature, and a lamp module offer actions such as turning a lamp on, off or to some dim level. Homer categorises these aspects of a component as triggers, conditions and actions.

A trigger reports something that happens externally to Homer, e.g. the front door is opened. A condition checks the state of a component, e.g. whether the front door is open. An action allows the user to request a change external to Homer, e.g. to lock the front door. Components state the triggers, conditions and/or actions they support.

Components are not allowed to communicate directly with other components; shared functionality must be provided by a Homer service. Components should be simplistic, with no intelligence or complex logic of their own. This cleanly separates the core devices and services from the logic and applications that build on these: the Homer framework acts as an intermediary between users and the home.

3.2 Sample Telecare Components

Homer can support any component which conforms to its API and event style. These are sufficiently simple and unrestrictive that almost any service or device hardware can be supported. The following component examples illustrate the kinds of capabilities that have been found useful in telecare:

Camera: The camera component allows for movement detection, photos and videos to be recorded on request; these can also be emailed or sent to a digital display within the home. This offers security features for residents, e.g. to check who is at the door or to check for a prowler outside the house. The camera offers communication features to allow residents to keep in touch with friends and family. It also offers peace-of-mind features, e.g. to allow informal carers to know that the resident is up and about the house.

Email: This supports exchange of email on behalf of other components.

Infrared: Most audiovisual devices have infrared remote controls. With ageing, users may lose dexterity in their hands so that traditional remote controls become difficult to use. The Homer infrared controller extends the variety of home appliances that can be controlled. For example, programmes can be recorded automatically and appliances can be used through a simple touch screen.

Momento: A very important aspect of telecare is communication. Older people, on the whole, like to feel close to their friends and family – photos are a good way of doing this. The Homer component for the i-mate Momento wireless digital photo frame (*www.momentolive.com*) has its own email address, allowing friends and family to email photos for immediate display.

Nabaztag: The Nabaztag 'Internet rabbit' (*www.nabaztag.com*) has been adapted as a user-friendly interface device. As a non-threatening interface to technology, this is ideal for technophobic or technically inexperienced users. The rabbit provides an interface which supports speech recognition, RFID-tags recognition, text-to-speech conversion, and audible, visual or gestural alerts.

Oregon Scientific: Homer can monitor the home environment using wireless devices produced by Oregon Scientific (*www.oregonscientific.com*). These are mostly used for information such as room temperatures and humidity levels. This information can be used to control the household environment.

Tunstall: For telecare, Homer supports a range of home devices produced by Tunstall (*www.tunstall.com*). This includes basic devices such as flood detectors, gas detectors, movement detectors and pressure mats, as well as more specialised devices such as medicine dispensers and door entry systems.

Twitter: Support for Twitter (*twitter.com*) helps to maintain communication using short messages. These can be used for status updates and alerts.

SMS: Similar to the email component, this component supports sending and receiving SMS messages.

Visonic: These sensors (*www.visonic.com*) are mostly for monitoring home activity, including door, window, motion and gas sensors.

WiiMote: The WiiMote (a hand-held controller, *www.nintendo.com/wii*) has been given a Homer component wrapping. The WiiMote can be used for gestural input; for example, it can mimic nodding or shaking the head in response to questions. It also has buttons which can be used for control functions. This is a good example of how a mass-market device, originally for a completely different purpose, can be adapted for use in telecare or home automation.

X10: This widely used technology for controlling mains appliances and lighting allows Homer to manage many devices around the home.

4. POLICY-BASED CONTROL

The Homer policy server handles logic and automation within the home. Examples might be turning on the heating if a cold night is forecast, turning on lights for security if the house is unoccupied, or alerting a neighbour if the resident is late in rising. The approach builds on the earlier work of ACCENT (*www.cs.stir.ac.uk/accent*) as a way of controlling many kinds of systems using policies [16].

4.1 Policy Format

Policies traditionally have an event-condition-action form. An evaluation with users showed that they often do not understand the difference a trigger and a condition. For example, many users would not distinguish '**when** the front door opens' (trigger) and '**when** the front door is open' (condition). The Homer policy language therefore blurs the distinction between these. A single **when** clause introduces all triggers and conditions, followed by a **do** clause with a list of actions. This results in a simple but flexible language where triggers and conditions can be freely mixed. Policies have a tree structure that allows **when** nodes to be combined using **and**, **or** and **then** (the latter for sequencing), and **do** nodes to be combined using **and**. Two unique features of the Homer policy language are duration limits on the **when** part, and support of conditions in the **do** part.

Duration limits determine how closely events must occur in the **when** clause. As an example, consider a policy that detects night wandering: 'when the resident gets out of bed at night and opens the front door'. A time limit of ten minutes might be appropriate for this. Without such a limit, the user getting up and later checking if the milk has arrived could be misconstrued as night wandering.

Conditions are supported within the **do** part of the policy as it was found that users often wish to impose conditions on actions. A typical policy might be: '**when** I get home from work **do** play my favourite music **and if** it is dark outside **do** turn on the hall light'.

4.2 Policy Grammar

The policy server can handle relatively complex policies. However, the user interfaces that are built to support the writing of policies for users can choose to support any level of policy complexity. The policy grammar supported by Homer is described fully in [7]; in outline, policies have the following structure:

```
policy: scenario actions ;
scenario: "when" when_node ("within" duration)? ;
when_node: when_and | when_or | when_then | trigger | condition ;
when_and: "and" when_node+ ;
when_or: "or" when_node+ ;
when_then: "then" when_node+ ;
actions: "do" do_node ;
do_node: do_and | do_if | action ;
do_and: "and" do_node+ ;
do_if: "if" (condition_node) ("else" do_node+)? ;
condition_node: condition_and | condition_or | condition ;
condition_and: "and" condition_node+ ;
condition_or: "or" condition_node+ ;
```

The policy server executes policies by following their tree structure. Policies are loaded into memory and represented as a hierarchy of nodes. A node is satisfied when all its children are (*and*), one of its children become satisfied (*or*) or each child has become satisfied in the required order (*then*). Finally, when the top level node (the whole **when** clause) is satisfied then the policy actions can be carried out.

4.3 Telecare Policy Examples

The following illustrates how policies can support telecare:

Sleeping Problems:

- **when** Tom gets out of bed at night **and** opens the front door within 5 minutes **do** activate his neighbour's bedside alarm
- **when** John gets out of bed at night **do** turn on the hall and toilet lights

Memory Problems:

- **when** Brian is late in taking medication **do** provide a reminder
- **when** Brian does not take medication for a whole day **do** send an email alert to the surgery **if** it is a weekday
- **when** the time is between 0500 and 1200 **and** Mary is in bed **and** the diary has an event in an hour **do** activate Mary's alarm clock
- **when** the living room is unoccupied for 5 minutes **do** turn off the television and the radio

Mobility Problems:

- **when** a person with a valid RFID tag arrives at front door **do** open the door **and** alert the user **and** display the visitor's photo

Hearing Problems:

- **when** music is playing anywhere **and** (the telephone rings **or** the doorbell rings) **do** reduce music volume by 90%

Comfort Features:

- **when** the living room is occupied **and** the living room light level falls below 60% do turn on the standard lamp
- **when** Mary is getting up **or** Mary is going to bed **do** set the bedroom temperature to a comfortable level
- **when** movement is detected in a room **do** set the room temperature to a comfortable level
- **when** the weather forecast predicts very cold weather during the night **do** turn on the heating
- **when** the left side of bed becomes unoccupied **and** the time is after 0730 **do** turn on the coffee machine
- **when** an SMS is received from Mary saying 'warm the house' **do** turn on the heating
- **when** the TV in the living room is turned off **and** the time is after 9:30pm **do** turn on the electric blanket in the master bed **and** tell Mary 'electric blanket has been turned on'

Safety Features:

- **when** the fire alarm is activated **and** no one is home **do** send an SMS alert to a neighbour
- **when** movement is detected outside **and** the time is between 2300 and 0600 **and** the house is in sleep mode **do** turn on the outside light **and** turn on the outside security camera
- **when** flooding is reported in the bathroom **do** turn off the water **and** send a recorded message by phone to a neighbour

5. HOME USER INTERFACES

Homer is exposed through a HTTP interface using JSON (JavaScript Object Notation, *www.json.org*). This supports information exchange in a neutral format, allowing any approved application to access Homer from a wide range of possible technologies and platforms. With a valid public application identifier and a secret key, an application can have access to all the devices within the home. An application can receive triggers, request actions, and view, edit, delete and add policies. This allows different types of external applications to be build for Homer. Many platforms and technologies could be used for Homer applications. These include Web technologies, Google Web Toolkit and Flash. Mobile devices can be used such Android, Apple and Nokia or systems. Desktop applications can be built for Linux, MacOS and Windows.

A survey on how users would like to control and program their home [8] revealed that this should be possible from 'anywhere'. Most users would prefer touch control than any other input method. For these reasons it was decided that sample applications for Homer should use a mobile phone and a tablet PC – the Apple iPhone and iPad were chosen as popular examples of these.

5.1 iPhone Application

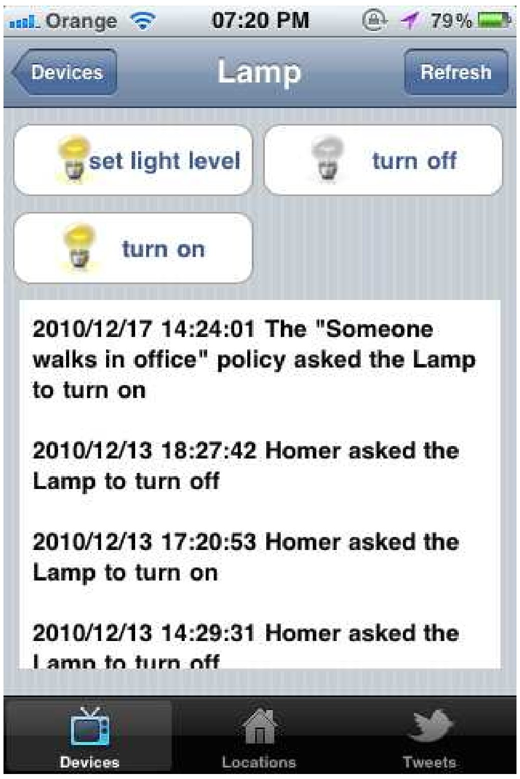
An Apple iPhone application was developed to demonstrate a simple home control application for Homer. This simple application demonstrates one possible means of accessing and controlling Homer. It can be used by resident, friends or caregivers. The user can browse devices by location (e.g. living room or bedroom) or by type (e.g. television or lamp). When viewing a device, users can see the recent events involving this, the current device state (e.g. on or open). Device state can be changed if this is supported (e.g. turning off a lamp is possible, whereas closing a window may not be). The device view is illustrated in figure 3.

5.2 iPad Application

The Apple iPad was chosen as the main device to demonstrate the full functionality of Homer. The application is currently under development, but is planned to offer a fully interactive view of the home. This will show the live status of devices and offer control over these. Two modes of operation are supported: integrated use of devices and policies, and separated use of these in two different applications. This allows more technically capable and interested users to have access to policies, allowing these to be viewed, edited and deleted (according to set permissions). For telecare, polices are written and maintained by a formal carer or family member (with appropriate training).

In many systems, writing policies requires specialised technical knowledge (and often programming skills). Homer policy support has therefore been made as simple as possible, as it is likely a carer will not have technical expertise. Current research approaches to end-user programming include tangible and visual programming, programming by demonstration, and use of natural language. These vary in degrees of success, but there is certainly no universally accepted solution.

The iPad application has been developed using a hybrid of natural language and visual programming [7]. An important benefit of the approach is that users define policies in ways that are meaningful to them. For example, the same underlying policies can be defined from the perspective of locations, devices, people or time. Users are also able to refer to devices as they wish, and even to use multiple names for the same thing (e.g. 'TV', 'lounge television').



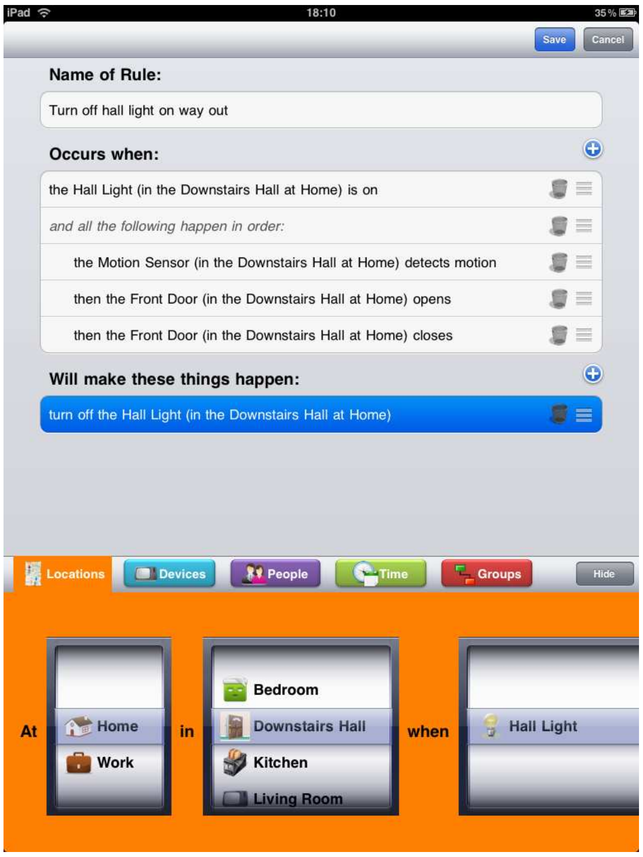
2010/12/17 14:24:01 The "Someone walks in office" policy asked the Lamp to turn on

2010/12/13 18:27:42 Homer asked the Lamp to turn off

2010/12/13 17:20:53 Homer asked the Lamp to turn on

2010/12/13 14:29:31 Homer asked the Lamp to turn off

Figure 3: Homer iPhone Application



Name of Rule:

Turn off hall light on way out

Occurs when:

the Hall Light (in the Downstairs Hall at Home) is on

and all the following happen in order:

the Motion Sensor (in the Downstairs Hall at Home) detects motion

then the Front Door (in the Downstairs Hall at Home) opens

then the Front Door (in the Downstairs Hall at Home) closes

Will make these things happen:

turn off the Hall Light (in the Downstairs Hall at Home)

Figure 4: Homer iPad Application

Figure 4 shows an example of defining a policy from the location perspective (chosen here as what happens at home in the hall). This policy aims to save energy by not using unnecessary lights. The **when** part of the policy defines a scenario as a combination of triggers and conditions. Here, the policy applies if the hall light is on, there movement in the hall, and the front door is opened then closed. The **do** part of the policy turns the hall light off.

6. EVALUATION

Home trials to evaluate Homer in practice are planned. This section assess policy support in Homer through comparison with other approaches.

6.1 Related Systems

ACCENT (Advanced Component Control Enhancing Network Technologies, *www.cs.stir.ac.uk/accent*) is a comprehensive policy-based management system that is applicable in a number of areas. A policy server manages and executes user-defined policies. The underlying language which represents the goals and policies is APPEL (Adaptable and Programmable Policy Environment and Language, *www.cs.stir.ac.uk/appel*). Complementing the policy server is a goal server that allows the user to define high-level objectives [15]. Various policy wizards allow non-technical users to define policies easily. Conflicts among policies are automatically detected and resolved (e.g. the user wishes the house to be warm, but also wishes to save energy). The policy system is interfaced to the target underlying system to be managed. The ACCENT policy system is comprehensive but complex, and more work is needed on making it user-friendly (e.g. easy definition of rules, supporting fuzzy policies, and explaining policy actions to the user).

The ACCORD project [11] used physical jigsaw pieces that can be combined to form rules. It is a framework for allowing dynamic configuration of a library of components to form policy-style rules within the home. User trials described in [11] were very successful. They highlighted that users were able to easily grasp the notion of jigsaw pieces representing various devices or functions that could be combined. Users were able to create sets of interconnected components to solve example problems, and also suggested further components with sample applications.

CAMP (Capture and Access Magnetic Poetry [13]) used magnetic poetry to provide users with a flexible, yet computationally constrained, means of natural language programming. The aim was to support automated capture and playback of home activities. The system allows users to define goals and rules at a very high level by piecing together words from a library in any desired order. The system then parses the natural language rules into a lower level intermediate representation, interpreted by the underlying capture and access system INCA. Preliminary user evaluation reaffirmed the expectation that CAMP's interface was extremely simple to use and allowed users flexibility to express their intentions in a way that made sense to them.

Ponder2 (*www.ponder2.net*) is a reimplementation of Ponder [2], which was a popular example of a policy-based system. Ponder2 is still work in progress, but improves implementation and design issues. Work is ongoing on redesigning and integrating each aspect of Ponder. Ponder2 has been used in many different application areas including robots, body sensor nodes and mobile telephones. Research applications have included health monitoring, unmanned autonomous vehicles, and large web-based infrastructures. The work presented in [17] explores the use of Ponder2 for autonomous pervasive environments.

6.2 Comparison with Related Work

For policies to be useful in telecare and home automation, they must be part of a sophisticated home system or be simple to integrate into existing home systems. They need to offer immediate benefits in a simple way. Home policies should offer the ability to piece together small capabilities so as to usefully automate a variety of tasks. This might involve mixing hardware devices and software services in a flexible and reconfigurable way. The following comparison criteria, assessed in table 1, are relevant:

Easy system integration: Can the approach readily be integrated into a new system? The extent to which Homer can be integrated with other systems is that it is fairly easy to give existing components an appropriate wrapping. Ponder2 is a policy language and framework rather than a home system, so it is expected that it could be integrated fairly readily. Systems that cannot be easily integrated should ideally be already embedded in a sophisticated home system.

Existing system embedding: Has the approach already been deployed in a home system? Only ACCENT and Homer are home systems. Accord is part of a simplified home system, with a limited set of supported devices and services. CAMP is limited to a small subset of the home technology: simple capture and access devices.

Easy device/service extension: Does the approach support the ready addition of new components? It is extremely important in telecare (and home automation) that new devices be easily added as they become available, and that these can be automatically supported by the policy server. Only Homer and Ponder2 offer a policy server that handles devices abstractly, and are therefore able to support new devices easily.

Easy policy definition: Is it easy for end users to write policies? Normally the definition of home logic involving multiple devices is hard-coded during development. This is costly and hard to maintain and customise. It is very desirable to make it easy for end users, caregivers, etc. to define policy. Only Accord, CAMP and Homer address this fully.

Flexible interface support: Is the approach designed to support multiple user interfaces so as to offer a choice? This is desirable, though not essential. ACCENT Homer and Ponder2 can offer this as they abstract policies to a level below the user interface. Accord and CAMP offer a unique and hand-built interface that is tightly coupled with their approach.

Telecare device compatibility: Are existing telecare products supported? Only ACCENT and Homer have been designed explicitly to support telecare.

Policy conflict handling: Are conflicts among policies detected and resolved? Policy conflicts are inevitable in telecare since different stakeholders can define different policies for the resident. ACCENT and CAMP offer full support for this. Homer support is being adapted from the ACCENT work, while Ponder2 support is being absorbed from Ponder.

The comparison in table 1 shows no system fully meets all the criteria. However, there is evidence that Homer is currently the best match to the requirements.

7. CONCLUSION

This section summarises the work and points to future developments.

7.1 Summary

It has been explained that Homer aims to meet the needs of both telecare and home automation through core capabilities coupled with support for more specialised devices and services. A flexible architecture has been introduced that allows components to describe key features of themselves. For example, the triggers, conditions and actions supported by a component makes user control and configuration easy. New and existing devices can readily be added or removed at run time, allowing the home system to evolve. Components are also well integrated with policies as a means of letting users manage how the home should behave. The functionality of Homer is exposed through a platform-neutral interface that makes it possible to develop a wide range of user interfaces.

The application to telecare has been described. Components appropriate to home care have been illustrated. Sample telecare policies have been given. Two sample interfaces have been presented, specifically designed for non-technical users and therefore suitable for telecare. Homer and similar systems have been assessed against criteria appropriate to a telecare system. It has been seen that Homer offers a number of benefits.

7.2 Future Work

Homer scales well to hundreds of policies. However, a potential issue with many policies is that they contradict with each other. Work is under way to adapt conflict handling techniques from ACCENT [16] for use with Homer.

Another possible problem with many policies is that it might become difficult to discover why the home took certain actions. Techniques from expert systems are being investigated as a means of explaining to the user the reasoning that led to particular actions. This will be particularly important when conflict handling is introduced, since the user might wonder why some policy was not applied.

8. ACKNOWLEDGEMENTS

Claire Maternaghan is supported by the Scottish Informatics and Computer Science Alliance (SICSA), the University of Stirling, and the MATCH project (Scottish Funding Council, grant HR04016). The authors are grateful to Phil Gray and Marilyn McGee-Lennon (University of Glasgow) for advice on user interface aspects.

9. REFERENCES

[1] A. F. Blackwell and R. Hague. AutoHAN: An architecture for programming the home. In *Proc. Symp. on Human Centric Computing Languages and Environments*, pages 150–157. Association for Computing Machinery, New York, USA, Sept. 2001.

[2] 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, Jan. 2001.

[3] A. K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu. A CAPpella: Programming by demonstration of context-aware applications. In *Proc. Conf. on Human Factors in Computing Systems*, pages 33–40. Association for Computing Machinery, New York, USA, Apr. 2004.

Criterion	ACCENT	Accord	CAMP	Homer	Ponder2
Easy system integration	partly	no	no	partly	yes
Existing system embedding	yes	no	no	yes	no
Easy device/service extension	no	no	no	yes	yes
Easy policy definition	partly	yes	yes	yes	no
Flexible interface support	yes	no	no	yes	yes
Telecare device compatibility	yes	no	no	yes	no
Policy conflict handling	yes	no	yes	pending	pending

Table 1: Policy System Comparison for Home Use

[4] K. Gajos, H. Fox, and H. Shrobe. End user empowerment in human centered pervasive computing. In F. Mattern and M. Naghshineh, editors, *Proc. 1st Int. Conf. on Pervasive Computing*, number 2414 in Lecture Notes in Computer Science, pages 134–140. Springer, Berlin, Aug. 2002.

[5] 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.

[6] C. Leong, A. 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.

[7] C. Maternaghan. The homer home automation system. Technical Report CSM-187, University of Stirling, UK, Dec. 2010.

[8] C. Maternaghan. How do people want to control their home? Technical Report CSM-185, University of Stirling, UK, Dec. 2010.

[9] M. C. Mozer. The neural network house: An environment that adapts to its inhabitants. In M. Coen, editor, *Proc. AAAI Symp. on Intelligent Environments*, pages 110–114. AAAI Press, Mar. 1998.

[10] M. W. Newman, A. Elliott, and T. F. Smith. Providing an integrated user experience of networked media, devices, and services through end-user composition. In *Proc. Symp. on Human Centric Computing Languages and Environments*, number 5013 in Lecture Notes in Computer Science, pages 213–227. Springer, Berlin, May 2008.

[11] T. Rodden, A. Crabtree, T. Hemmings, B. K. J. Humble, K.-P. Åkesson, and P. Hansson. Configuring the ubiquitous home. In *Proc. 6th Int. Conf. on The Design of Cooperative Systems*, pages 215–230. IOS Press, Amsterdam, May 2004.

[12] T. Sohn and A. K. Dey. iCAP: An informal tool for interactive prototyping of context-aware applications. In *Proc. Int. Conf. on Human Factors in Computing Systems*, pages 974–975. Association for Computing Machinery, New York, USA, Apr. 2003.

[13] K. Truong, E. M. Huang, and G. D. Abowd. CAMP: A magnetic poetry interface for end-user programming of capture applications for the home. *UbiComp 2004: Ubiquitous Computing*, pages 143–160, 2004.

[14] 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, USA, May 2010.

[15] K. J. Turner and G. A. Campbell. Goals for telecare networks. In A. Obaid, editor, *Proc. 9th Int. Conf. on New Technologies for Distributed Systems*, pages 270–275. Université de Québec à Montréal, Canada, July 2009.

[16] K. J. Turner, L. S. Docherty, F. Wang, and G. A. Campbell. Managing home care networks. In R. Bestak, L. George, V. S. Zaborovsky, and C. Dini, editors, *Proc. 9th Int. Conf. on Networks*, pages 354–359. IEEE Computer Society, Los Alamitos, USA, Mar. 2009.

[17] K. Twidle, N. Dulay, E. Lupu, and M. Sloman. Ponder2: A policy system for autonomous pervasive environments. In *Proc. 5th Int. Conf. on Autonomic and Autonomous Systems*, pages 330–335. IEEE Computer Society, Los Alamitos, USA, 2009.