1 Introduction

Although our complex lives are not necessarily improved by each new technological widget we adopt, uncooperative devices and appliances with deficient user-interfaces can certainly conspire to frustrate us. Over the past three years I have experimented with a number of technologies that gave birth to the *information furnace* concept: a basement-installed PC-type device that integrates existing consumer home-control, infotainment, security, and communication technologies to transparently provide user-friendly access and synergistic value-added services. In the following sections we will examine the devices and appliances lurking in the modern home, overview the problems associated with the current breed of devices, and go over the basic elements of the information furnace concept and its prototype implementation.

This paper contains more material than what would be strictly necessary to present and substantiate my thesis for the information furnace. The reasons are twofold: firstly I tried to extensively document the main aspects of the prototype implementation, and, secondly, I wanted to demonstrate that all I really need to know about system administration I learned building the information furnace (with apologies to Robert Fulghum [15]).

2 The Modern Home

A modern home contains a large number of sophisticated devices and technologies. Current and near future technologies and respective devices can be roughly categorised into the categories of home control, infotainment, security, communication, and special-purpose devices.

2.1 Home Control

Contemporary central heating systems are regulated by one external and a number of internal temperature sensors in conjunction with a control unit occupants use to set the desired room temperature. The system compares the internal room temperature to the setting of the control unit and, using the external temperature as a compensating factor, regulates the temperature of the water produced by the local heat-generating plant or the valve bringing remotely-heated water into the home. Burners often have their own control circuits based on target temperatures for the burner and the circulating pump, but we can regard them as a black box for the purposes of this article. Convenience elements associated with control units involve the ability to maintain different temperature settings for day and night, manually set the system to day, night, or absence mode, keep a weekly schedule of automatic switchovers between these modes, and switch-off for the prescribed duration of a trip.

Instead of a burner, some systems are based on a heat pump and air circulation. They are controlled by the same principles, but can also lower the building's temperature during hot days. Split-type wall-mounted room air conditioners feature an integrated opaque control circuit adjusted individually through a remote control.

The provision of hot running water to the bathrooms and kitchen is often controlled together with the central heating system. The added complications this brings into the picture involve the possibility of heating the water on sunny days through a solar panel, an electrical heater used as a backup measure, a circulating pump to pass water through the solar panel, and a second pump to bring hot water near the taps. The first pump operates through a thermostat comparing the temperature difference between the hot water storage tank and the solar panel; we can again regard the system as a black box that absorbs solar energy. The operation of the second pump is more tricky: its intention is to save water by bringing the hot water close to the taps. When the central heating system is operating, having a secondary warm water circulating circuit in the house does not hurt; the floors and walls where the running hot water pipes run act as secondary radiators. When however the central heating is switched off (on warm days or during an absence) the circulator actually cools the stored warm water by continuously running it through the house. In my experience modern heating controllers do not deal with this complication.

The natural light entering a building is often controlled through external blinds or stores. These also play an important role in regulating the heat flowing into or out of the building. In addition, a *heliostat* device can be used to track the sun movement and actively reflect sunlight into the building. Artificial lighting can be electronically controlled through a system such as X10 or LonWorks in the United States and the European Installation Bus (EIB) in Europe. Perversely, in the case of the EIB at least, it is currently cheaper to control lights using 230V switches and individual switch-to-appliance power-carrying cables than to use a signal and power bus, cheaper control switches, and the associated electronics. This is clearly a case where the silicon economy has not yet done its work. Other interesting elements of modern artificial lighting include light fixtures with integrated motion and light detectors that are increasingly used outside homes as burglar deterrents, time switches used for the same purpose inside the house, and "economy"-type light bulbs that may take up to five minutes to reach their rated light output.

A case where the silicon economy *has* worked is exemplified by the availability of affordable devices to control plant and garden watering. These often sport a bewildering array of daily and weekly watering programs (apart from the one you really require, that is), can be directly fitted into a watering hose, or can control multiple valves, and can receive additional feedback from a soil humidity sensor.

2.2 Infotainment

The array of devices used for servicing our entertainment, and, supposedly, our information access needs (covering the so-called "infotainment" category) is bewildering. It involves CD, MP3, and DVD players, radios, the (increasingly digital and interactive [24]) television, tape or hard-disk based video recorders, digital photograph and video cameras, game consoles, and networked personal computers. Across those devices we typically witness a gratuitous duplication of functionality, and a lack of standardisation; both are exemplified by the growing array of remote controls adorning the typical lounge table. The last problem has spurned research [41] and development of universal, configurable remote controls.

2.3 Security

Home owners not wishing to trust their security of the prized possessions I outlined in the previous paragraph to the watchful eye of the local cop or a *bona fide* man-eating animal often end-up contributing to the bottom line of the burglar alarm and home monitoring industry. A modern burglar alarm consists of a control unit, an array of sensors, and facilities for alerting whomsoever the owner can afford. The sensors used include motion detectors based on passive infrared (PIR), microwave, or hybrid technologies, magnetic contacts that detect the opening of doors and windows, and glass vibration sensors. Sensors placed under mats and carpets, and light beam detectors are less often used. Contrary to the popular perception promoted by Hollywood films, visible red intersecting laser beams used to test a burglar's agility are *not* a popular sensor option.

The control unit is typically an overpriced, and underpowered microprocessor-controlled contraption. It monitors the sensors (due to a dearth of input ports these are often or-wired into "zones"), allows the owners to activate and deactivate it using a PIN, distinguishes between a normal entry (e.g. through a door) that provides a delay for deactivating the system and an unexpected event (e.g. motion, entry through a window) that immediately triggers an alarm, offers a facility for operating with the occupants inside the house ("night mode"), and controls the alarm triggering and rearming process. Alarms in most cases sound an internal siren that is supposed to frighten the burglars (but will in most cases only frighten the poor owners when set-off in a "night-mode" operation), activate an external siren—often coupled with a strobe light—that passers-by typically try to ignore, and notify via a modem or a recorded message a control centre or a list of pre-assigned phone numbers.

The whole system has some redundancy and self-monitoring capabilities. Many sensors and sirens are equipped with a normally-closed "tamper" switch; opening the device's cover, or cutting its connecting wire will be immediately registered by the alarm unit. The control unit is equipped with a battery which supplies power during a power failure. In addition, many outdoor sirens come with their own battery and are wired for stand-alone operation: if the power supplied by the control unit is interrupted or the siren's tamper switch is activated the siren will begin to sound. Some systems are also installed with wall mounted panic buttons or similar signalling tokens that an individual can wear. These are also useful when elderly or disabled people wish to signal they need attention. Some owners also combine their unit with fire-detection sensors; however, fire-detection equipment installed to satisfy building regulations falls outside the scope of this article.

Related to security are also the door phone (and sometimes a TV camera), the associated door opener, and the remote-controlled garage door opener. Note that the typical door phone and opener combination is an system cunningly designed to minimise the number of individual cables required for its installation. Interfacing with such a system can be very difficult; however many small private box exchanges (PBXs) offer a door phone / door opening option and can be easier to interface.

2.4 Communication

The modern home's communication needs are served by the phone and an Internet connection using POTS and a modem, ISDN and a terminal adaptor, or another digital network technology (e.g. *DSL) and the associated terminator box. Sharing of phone lines and internal communication can be facilitated via a PBX, while the corresponding sharing of data connections can be facilitated by a network and hub (or wireless network) and a router. PCs are also increasingly used to share network collections. Phone lines are often terminated on an answering machine and a fax; the more exotic ISDN offerings trumpeted by the incumbent telecom providers (videophones, digital faxes) have been persistently snubbed by consumers. Connected to the data lines are PCs, holding valuable personal data and in dire need of regularly scheduled backups, and connected to the PC are various PDAs holding the owner's telephone number directory and other personal data. A baby monitor typically functions independently of the above setup. A variety of wired and wireless home networking technologies aim at interconnecting the systems I described [9].

2.5 Special-purpose Devices

Finally, inside a modern home there is a number of electronically controlled special-purpose devices. These include the humble vacuum cleaner, the microprocessor engineer show-off case microwave oven, and the increasingly clever refrigerator, oven, washing machine, drier, and coffee machine [16]. Unfortunately for this article's author and probably fortunately for their other users, none of these devices offers a viable interface for controlling their operation.

3 Modern Problems

The coexistence of the devices and systems I described in the previous section under the same roof is a sad story of unattained potential, lost opportunities, and waste.

3.1 User Interface

The most important problem inflicting the systems is their often dysfunctional (to put it politely) user interface. The reason behind this problem stems from the restricted human interaction devices the systems have at their disposal. In most cases interaction devices consist of small numerical LCD displays (sometimes capable of displaying some additional hieroglyphic symbols), and a few domain-specific buttons. The systems I described rarely follow the principles of a user-centred design [31, p. 188]. It is thus difficult to determine what actions are possible at any moment, the system's conceptual model and current state are hidden from the user, and there are no natural mappings between a user's intentions, the required actions, and the resulting effect. Similarly, many of the user interface design Golden Rules [38, pp. 74–75] are never followed: interfaces are inconsistent, require tedious sequences of data entry, and often lack shortcuts, informative feedback, and the ability to reverse actions. Other important user-interface problems include non-intuitive interaction sequences, the operation in various different "modes" [36, pp. 30–57], the overloading of buttons for different purposes, cryptic display messages, lack of localisation and accessibility for disabled people, and a non-ergonomic design.

Appreciating that I might be accused of shooting a lame duck, I illustrate these points with three representative examples.

Programming a Heating Controller

The room unit in question allows programming a weekly schedule for the controller's operation. Programming is performed by switching between 17 different modes (each indicated on the panel by a different number). The following excerpt from the operation manual outlines the weekly programming procedure [39, p. 5]:

> "With the heating program, you can predetermine the temperature switch-over times for one week. The weekly program consists of seven 24-hour programs. One 24-hour program may include up to three heating periods each of which is defined by a start and an end time. If you do not require a certain heating period, you need to enter the same time of day as start and end time.
>
> **4** Select the required day for the heating period (1 = Monday / 7 = Sunday)
>
> **5** Start of heating period 1: nominal operation
>
> **6** End of heating period 1: reduced operation
>
> **7** Start of heating period 2: nominal operation
>
> **8** End of heating period 2: reduced operation
>
> **9** Start of heating period 3: nominal operation
>
> **10** End of heating period 3: reduced operation"

Operating a Digital Answerer

The state of this particular digital answerer is indicated by a single "messages indicator" light. Its behaviour is to be interpreted as follows [45, p. 5]:

On Answerer is on and there are no messages.

Flashing Number of flashes indicates number of messages.

Off Answerer is off, but there might still be messages.

Flashing rapidly Outgoing announcement is invalid or memory is full.

For remotely accessing the messages the device's owner is provided with a paper cut out "remote access card' that lists the eleven different commands (five are to be used during message playback and six at all other times) the answering machine supports.

Programming a PBX

This low end PBX can be programmed from a DTMF phone connected to the extension 21 (only). The PBX allows the specification of different extension ringing patterns for day and night use. To specify the day or night starting time the following procedure has to be followed [27, p. 25]:

- To enter the system programming mode dial 6206206#.
- To set the day night switching time dial #71 A BB CC D where:
  - A is 1 to specify the day start time, to the specify the night start time.
  - BB is the hour 01-12.
  - CC is the minute 00-59.
  - D is 0 for AM and 1 for PM.
- To exit the system programming mode dial 6206#.

The examples we have seen, illustrate that in many cases the user interface of consumer-oriented home appliances and control devices is far from ideal. Clearly human interface studies and approaches towards better interaction paradigms [32, 33] have not yet found their way into widespread practice.

3.2 Lacking Functionality

One other problem with the devices we examined is that for a number of reasons they may impose arbitrary limits on their functionality, or lack support for useful functions. For many devices the available CPU power, RAM, or ROM are just not sufficient for implementing a given function. For others, the already complicated user interface would crumble under the cognitive load of the added functionality.

As an example, there is no reason why the heating controller I described should support only three heating periods per day, or not allow one to provide a schedule for the temperature of the running hot water as well as the temperature of the room. Similarly, a CD player may offer a facility to skip a boring track, but will not remember to skip the same track in the future. On another front, an alarm unit could provide a precise report of the alarm triggering circumstances, and allow its user to remotely probe and disable individual sensors. Finally, the PBX we examined could be more versatile if it supported different day and night mode start times for different days of the week.

A general lack of functionality witnessed in all the devices we examined is a facility to backup and restore the tediously entered program data. True, many devices have a power-backup system for their memory contents, but, in my experience, that inevitably one day will fail—typically long after the user has forgotten how to program the device and has lost the respective user manual.

3.3 Lost Synergies

I will fully expand the synergies made possible when all home systems communicate and cooperate with each other in Section 4.3; at this point I will illustrate my thesis with a simple example. The "blinking clock syndrome" refers to the myriads of device clocks flashing "12:00" all over our planet. Even in households where these are correctly set after a power failure, twice a year they need to be re-adjusted following the daylight savings time settings. However, a correct time signal enters a modern home from at least three different sources: RDS radio, teletext TV, and the Internet [28]. In addition, modern operating systems can correctly interpret and adjust the time following the local daylight savings rules. Our wonderful devices however, fail to cooperate to correctly set their time.

3.4 Provisioning

Related to the lost synergies is the duplication of hardware and functionality we witness in the modern home. Provisioning communication, user access, power, and space for all different devices is simply an unproductive use of resources.

Communication

The systems I outlined in Section 2 are typically implemented using the following distinct communication networks:

1. Voice
2. Data
3. Door interfacing
4. Heating
5. Security
6. Light control

There are (expensive) systems that integrate some of the above functions, but the general case involves a waste of resources.

User Interface

Each home system has its own user interface, with its ergonomically-challenged human interfacing devices. Humans have to learn different dysfunctional interfaces to perform a limited number of tasks.

Power

Each system needs line power, and, in the best case, also has a separate backup power system (typically a 9V battery). Apart from the nuisance of maintaining the tens of different backup power systems, the power requirements of all devices add-up to a sizable power drain which is both expensive, and environmentally unsound.

Space

Finally, many devices occupy space in living areas daily imposing their unsightly presence on us. The ubiquitous table with the telephone, answering machine, and fax is one example; the collection of the remote controls on the lounge table is another.

4 The Information Furnace

The Information Furnace, supporting the post-PC ubiquitous computing paradigm [49, 34], is a basement-installed PC-type device that integrates existing consumer home-control, infotainment, security, and communication technologies to transparently provide user-friendly access and synergistic value-added services. The use of integrated intelligent devices in the home automation area is not a new concept [30, 4, 7, 14] the information furnace differs however from other approaches by prescribing concrete architectural guidelines, expressly adopting a maximalistic approach towards its functionality, and aggressively targeting and exploiting the resulting synergies.

4.1 Architecture

The architecture of the information furnace is based on three basic premises. The device:

1. is located in the basement or in a cupboard,
2. acts as a central hub for content, communications, and control, and
3. offers multi modal user-friendly access to all its functions.

The location of the device in a secure, non-accessible place is central to our design having a number of important repercussions. Firstly, the same location will be used to terminate the various connections. These often include home networks, telephone lines, reception antennas, network lines, and cable TV connections. The unsightly presence of all these cables can only be accommodated in a specially provisioned place. In addition, the noise the system will generate can be effectively isolated. Rotating hardware (hard disks and fans) and other noise-generating components such as electromagnetic relays can be brought together into a single place keeping the rest of the house serene. Furthermore, the system can be physically secured deterring burglars, minors, or even pranksters (have you checked your answering machine message lately?) Finally, an appropriate UPS can be provisioned to constantly maintain power without worrying about its size, noise, or appearance, or the distribution of power to multiple locations.

As the furnace acts as a central hub for content, communications, and control, we can eliminate wasteful duplication, provide universal access to all its functionality from any local or remote location, centralise our access and control policies, effectively backup all data, and, most importantly, exploit the synergies that the centralisation allows. A single modern CPU can easily handle all the functions I described in Section 2. Thus the numerous underpowered, specialised devices can be replaced with a single general-purpose one. When all functionality is housed in a centrally connected location, it can be accessed from all networked locations. Thus elements such as, the family's music and photograph collection, the answering machine messages, lighting controls, the burglar alarm log, and the heater programme are available from all rooms in the house, and also from remote locations. Naturally, the centralisation of these important functions entails considerable risks; these can however be effectively controlled if the associated policies are centralised, reviewable, and implemented under a reasonably secure operating system. In addition, all the programming and other information stored in the device can be centrally backed-up on a regular basis. Surprisingly, the information furnace concept, when applied as a replacement for the stand-alone provision of the functions it supports, increases all aspects of the *figure of merit M*, originally proposed for nomadic computers [26]:

$$M = \frac{\text{Intelligence}}{\text{Size} \times \text{Cost} \times \text{Power}}$$

However, the most important benefit of the centralisation is the synergies that can be exploited; we will examine this aspect in Section 4.3

The final element of the information furnace architecture concerns its user interface. I do not believe that a single user-interface is appropriate for all occasions. For this reason, the information furnace offers a number of different access modes. These can include web forms and Java applets, telephone-based DTMF commands, infrared remote controls, access via Bluetooth devices, or even a command-line interface. Thus, for selecting a song to hear one will use an infrared remote control, to start the hot-water boiler when returning from a trip one will issue DTMF commands over the cellular phone, to open the garage door one could use a Bluetooth interface of a PDA, and to program or review the activity log of the PBX or the burglar alarm one would prefer to interact with a web form. Ideally, all functionality should be available from all devices; at night one might prefer to use the bedside phone to check the burglar alarm sensors; when working on a PC, a web interface might be used to review the answering machine messages. Some of the access modes can be more user-friendly than others, however the processing and storage power of the information service means that there will be no artificial restrictions to the usability of a particular access mode. As an example, a complete answering machine help menu can be made available as a voice message over a phone connection without requiring the user to rely on cut-out cards or memorise the access commands.

4.2 Functionality

The functionality the information furnace provides encompasses everything it can reliably and safely accommodate. I take this maximalistic view, because, by my experience, every system and function moving to the information furnace automatically benefits from universal multi-modal access, user-friendly control, and data backup, while providing additional opportunities for synergies with other services. Thus, the information furnace can control:

- the home's climate and hot water provision,
- external and internal blinds,
- artificial lighting,
- alarm sirens, and
- door openers.

It should receive input from:

- the phone via DTMF commands,
- web forms and Java applets,
- remote controls,
- internal and external temperature sensors,
- motion detectors and opening sensors,
- rain and moisture sensors, and
- Bluetooth devices.

The information furnace should also act as the centralised repository for the home occupants data, in a manner analogous to the one suggested by the CyberAll project [2]. This includes:

- the music collection stored in MP3 form,
- synchronised copies of PDA contents,
- the digital camera photographs, and

- (in the future) the video collection.

Finally, the information furnace integrates the home's communication interfaces by performing the functions of a firewall, a router, an, intercom, and a PBX.

When some type of functionality can not be directly implemented by the hardware at hand, the information furnace shall at least communicate with the respective dedicated device so that it can indirectly control it. As an example, by communicating with a PBX using a simple modem, one can provide a decent user-interface to the functionality I described in Section 3.1.

All integration shall of course be performed with an eye on safety and security. Where appropriate the information furnace should work in parallel with dedicated hardware providing redundancy, or be isolated from it. Elevators, fire monitors, and emergency lighting should probably be left to work on their own; tapping an elevator's "call" button or a fire-alarm's output should be the limit to the type of coupling that should be considered safe. Similarly, control of mains voltages should be performed by dedicated hardware, leaving to the information furnace the task of issuing the respective commands [4].

4.3 Synergies

The most effective user interface is the one that does not exist. Centralising all home control, content, and communications in a single place allows us to exploit synergies that make many control functions redundant, or provide new and more versatile features. First of all, the collocation of all services in a powerful processing and storage device makes it possible to provide centralised backup, universal and multi-modal access to all functions, and user-friendly interfaces.

Consider the alarm-system motion detectors. These can detect activity in rooms and can therefore be used to:

- start the running hot water circulation pump when an occupant approaches the bathroom or kitchen,
- close the blinds when the owners are in a room to protect their privacy and, otherwise, open them in cold summer nights and sunny winter days to improve the home's climate control,
- turn artificial lighting on and off as the owners move across rooms—additional hints such as entertainment system or communications activity can be used to improve the heuristics of this approach,
- avoid ringing the phone in a bedroom with no activity (where presumably an occupant might be sleeping) when activity in another room indicates that someone else might prefer to pick-up the phone. (Note that reversing the above conditions does not yield a heuristic many adults would agree with.)

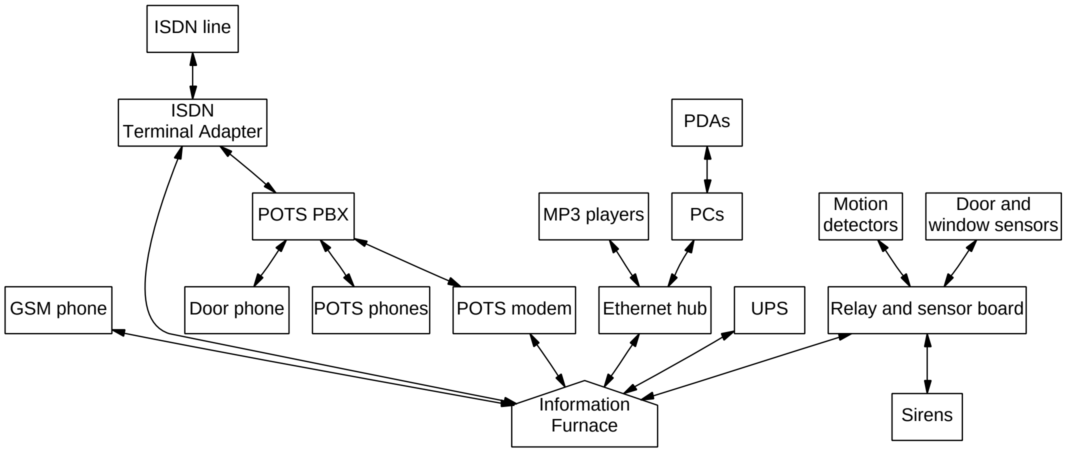

Figure 1: Information furnace connection diagram.

When leaving the home and on return the activities we perform can be comparable to walking through a jet-pilot's checklist. The information furnace can collectively perform these standardised activities through a single command. Thus the "leave-home" command will turn-on the answering machine, switch-off the internal artificial lighting and entertainment systems, lower the central-heating temperature, light the entrance, activate the burglar alarm, and open the garage door. On return a single (password protected) command will deactivate the burglar alarm, turn-off the answering machine, play-back any incoming messages, provide caller-id information on unanswered phone calls, switch-on the internal artificial lighting and entertainment systems, raise the central-heating temperature, switch-off the entrance light, and close the garage door. Similar sequences can be used for putting the house to sleep and preparing it for its owner's wakeup.

Other activities can trigger synergistic events. As an example picking up the phone can cause the entertainment system to pause the music or video playback; when the alarm system detects an unlawful entry it can begin flashing all the house's lights to frighten the burglar and attract neighbourhood attention; watering the garden should probably be avoided when the garden lighting indicates that a party is taking place; a visitor overstaying his welcome will cause a gradual lowering of the house's temperature and lighting.

5 Prototype Implementation

To experiment with the ideas I outlined in the previous sections I designed and implemented a prototype of the information furnace. (In all honesty this is not an entirely accurate description of the causal relationship between the two aspects of my work, but seems to be the generally-accepted politically-correct way of expressing it.) The implemented information furnace provides the functionalities of an alarm system, an answering machine, a fax server, a PBX interface, an internet firewall and router, a content management and distribution point, and a backup server.

5.1 System Structure

You can see a diagram of the information furnace connections in Figure 1. The information furnace consists of a low-end (100 MHz Pentium) PC equipped with a 40Gb hard disk, an additional serial port card, and an Advantech PCL-724 24-bit digital input / output card. For the PC I was fortunate to acquire a surplus IBM Personal Computer 340 unit; running FreeBSD 4.1 it proved to be a very stable platform with uptimes in excess of 200 days. The national telecom operator provides with each ISDN connection a terminal adapter with two POTS (plain old telephone system—traditional analogue phone) interfaces and an RS-232 or USB data port. The data port is connected to the information furnace for providing internet access and firewall functionality. The two POTS interfaces are connected to an entry-level analogue PBX. I decided to use an analogue PBX instead of an ISDN model to minimise the system's cost and by reasoning that new upcoming telephony offerings, such as xDSL or fixed wireless lines, might not be compatible with an ISDN PBX. The PBX connects to a number of plain phones, a door-entry phone, a relay-actuator for opening the door, and a POTS modem used for programming the PBX and providing a voice/DTMF interface. The 40Gb hard disk is used to store music content in MP3 form to distribute throughout the house and as intermediate storage for backup purposes. PCs and MP3 players connect to the information furnace via an Ethernet LAN. The GSM phone and a UPS, both connected to the information furnace via serial links, provide communications and power backup.

Connecting the alarm system devices to the furnace was more challenging. Alarm sensors and actuators typically work with 12V voltage, while the digital I/O card I used provided an 8255-compatible TTL type interface on a 50-pin ribbon-cable connector. To match the physical form and electrical characteristics of the two systems I designed and implemented a simple PCB (printed circuit board) circuit that converts sensor signals into TTL-compatible inputs, uses relays to activate external loads, and provides screw-clamp terminal blocks for connecting the sensors and sirens (Figure 2, right).

5.2 Home Security

The information furnace's alarm subsystem consists of a device driver that interfaces to the PCL-724 card and a daemon that monitors sensors and reacts to signals and commands.

The PCL-724 card emulates the Intel 8255A programmable peripheral interface chip running in mode 0 (simple I/O). It provides two 8-bit ports (port A and port B) and two 4-bit ports (port C upper, port C lower). Each port can be individually programmed for input and (latched) output and appears at a different offset of the device's base I/O address. One of the lines can also trigger an interrupt, but I did not use that feature. A separate register allows the configuration of ports for input or output. The device is so simple, that reliably probing for it when input data arrives at its terminals is impossible;

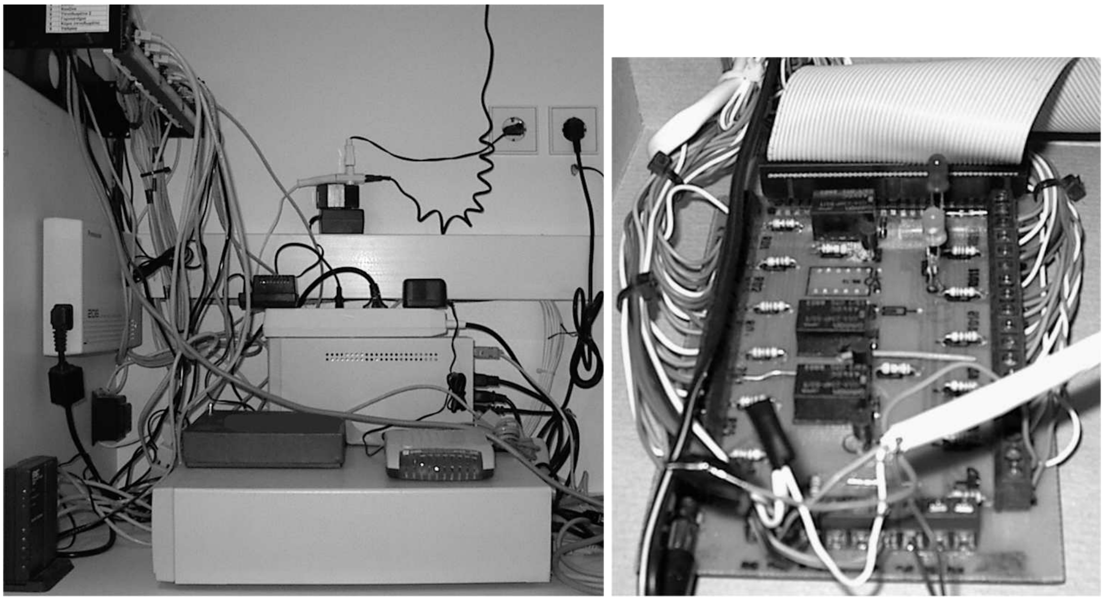

Figure 2: The information furnace (left) and the sensor connection PCB (right).

```
if (oflags & FWRITE)
    /* Writing == output; zero the bit */
    outb(scp->iobase + PBIO_CFG, scp->iomode = (ocfg & (~portbit)));
else if (oflags & FREAD)
    /* Reading == input; set the bit */
    outb(scp->iobase + PBIO_CFG, scp->iomode = (ocfg | portbit));
else
    return (EACCES);
```

Figure 3: Configuring the PCL-724 I/O ports.

therefore the kernel configuration has to specify the device's base address. The device driver provides four character devices that correspond to the card's I/O ports. Opening a device for read or write, automatically configures the corresponding hardware port for input or output, as illustrated in Figure 3. Initially all ports are set for input to avoid damaging external circuitry.

When designing the device driver, to minimise kernel context switches, I specified three different ways I/O would be performed:

Basic The read or write operation returns immediately after reading or writing the data to the port at bus speed.

Paced Data is transferred from or to the port at intervals specified by a separate *ioctl(2)* call.

Differential Only port values that differ from the previous port value are returned.

Modes can be set via *ioctl(2)* calls. However, experimenting with the basic mode I found that polling the input ports at one second intervals provided acceptable functionality (most alarm sensors have their own latches) with negligible impact on performance. I therefore decided to handle the rest of the complexity through user mode polling, realising however that other applications might benefit from a more sophisticated driver implementation.

The user-mode alarm daemon is structured around an event-driven driven loop. Three types of events are handled:

- external commands (e.g. arm, disarm, panic),
- sensor inputs, and
- elapsed timers (used for providing delays, automatic re-arming, and notification intervals).

Different levels of logging are provided by calls to the *syslogd(8)* daemon. Apart from triggering the various sirens, alarms cause the queuing of voice and data messages to kind (unlucky) individuals and the responsible authorities via the modem and the (backup) GSM phone.

The actual behaviour of the alarm is specified using a domain-specific language [44, 42, 43]. A domain-specific language (DSL) [35] is a programming language tailored specifically for an application domain: rather than being general purpose it captures precisely the domain's semantics. Examples of DSLs include *lex* and *yacc* [20] used for program lexical analysis and parsing, HTML [5] used for document mark-up, and VHDL used for electronic hardware descriptions. Domain-specific languages allow the concise description of an application's logic reducing the semantic distance between the problem and the program [3, 44]. As a design choice for implementing safety-critical software systems DSLs present two distinct advantages over a "hard-coded" program logic:

```
leave:
    | set_sensor_active(ALL, OFF)
    | set_sensor_active("Door", ACTIVE)
    > wait_for_door_open
    ;

wait_for_door_open:
    | syslog(LOG_INFO, "Waiting for door open")
    ActiveSensor > door_open
    ;

door_open:
    | syslog(LOG_INFO, "Door opened")
    10s > day_arm
    ;
```

Figure 4: DSL specification of the “leave” command.

Concrete Expression of Domain Knowledge Domain-specific functionality is not coded into the system or stored in an arcane file format; it is captured in a concrete human-readable form. Programs expressed in the DSL can be scrutinised, split, combined, shared, published, put under release control, printed, commented, and even be automatically generated by other applications.

Direct Involvement of the Domain Expert The DSL expression style can often be designed so as to match the format typically used by the domain expert. This results in keeping the experts in a very tight software lifecycle loop where they can directly specify, implement, verify, and validate, without the need of coding intermediaries. Even if the DSL is not high-level enough to be used as a specification language by the domain expert, it may still be possible to involve the expert in code walkthroughs far more productive than those over code expressed in a general purpose language.

The DSL used for specifying the alarm daemon behaviour describes a state machine. Each state description consists of its name, actions to perform when it is entered (written on lines starting with a $|$ symbol, and events that lead to other states (denoted using a $>$ symbol). Actions are simply C function calls. To enhance the DSL’s expressiveness a state can also transfer immediately to another state without waiting for an event; I use this feature to modularise the specification by defining “subroutine” states. As an example, the sequence in Figure 4 is used to specify that a “leave” command will arm the system 10s after opening the door: A small Perl [47] script transforms the alarm specification into an efficient C loop structure.

5.3 Telephone Integration

The answering machine, fax server, PBX programming, and alarm-notification functions of the information furnace are handled by software written on top of the *vgetty* [10] extension: a voice-handling add-on to the *mgetty* [8] package, which in turn replaces the Unix *getty(8)* terminal handler to handle data and fax calls. I wrote the incoming and outgoing modem interaction scripts in Perl using the *Modem::Vgetty(3)* [21] Perl package.

We often access the answering machine through the phone using a DTMF voice menu. The interaction of the answering machine software with the PBX poses an interesting problem. During normal use we want to be able to access the answering machine voice menu, but we do not want it to answer incoming calls. This was solved by having the answering machine setup the PBX to direct external incoming calls to all extensions but the modem during normal use, and only to the modem extension when the answering machine is enabled.

The PBX provides a global 100 phone quick access memory feature. By using these memories one can access the same number from all extensions, without having to individually program and maintain the memories of each different telephone. Apart from offering a centralised point for storing the quick-dial numbers, this approach obviates the need to handle the disparate user-interface each device has for storing phone numbers (is the programming sequence "*code* AUTO *number store*" or "MEM *code number hangup*"?) Of course, this approach solves one user-interface problem by replacing it with another, since the quick-access programming sequence for the PBX is (hold your breath) "6206206# #00*code*0*number*# 6206#". Thankfully, having the PBX connected to the information furnace, one can easily package this functionality as a shell script

```
vm shell -l ttyd1 -S /usr/bin/perl call.pl "#00${1}0${2}#"
```

and have another script program the PBX quick-access memories to a known state:

```
# John Doe Home
setmem 10 0105554321
# John Doe mobile number
setmem 11 0935551234
# John Doe Office
setmem 12 0105556789
```

Since this script is rarely used, I did not provide a more elaborate interface to it, although the script could easily be generated by mining the PDA phone database backups, or through web forms. However, even in the format it currently is, it proved a time-saver when the country's numbering plan changed: a simple global replace operation in the editor resulted in a new script that when run correctly programmed the PBX memories for the new plan.

You will probably have noticed that many different operations may compete for accessing the modem at a given time. To solve this resource contention problem, I wrote a command queue handling daemon as a simple shell script, illustrated in Figure

```
#!/bin/sh
VMDIR=/var/spool/voice/vmq
cd /
exec <&- >&- 2>&-
while :
do
	for i in $VMDIR/vm.*
	do
		$i && rm $i
	done
	sleep 10
done &
echo $! >/var/run/vmd.pid
```

Figure 5: File identification tags used in revision control systems.

5. Modem accessing commands are deposited in the form of small shell scripts in the voice shell spool directory. The queuing function is provided by Perl and C language libraries, so that individual requests are uniformly named according to the date and time they were generated and therefore processed in the correct order. The final name of each script is given using a *rename(2)* call, ensuring the operation's atomicity.

5.4 Content Distribution

A motivating requirement that led to the information furnace's conception was the ability to access our music collection from any networked place in the house. Converting CDs into a collection of MP3-compressed files is a relatively easy task these days. I used *dagrab* [46] and *cdparanoia* [29] to extract raw content from audio CDs, and the encoders *bladeenc* [19] and *notlame* [37] to convert that content into MP3 form. More difficult were the tasks of organising the transfer of a set of CDs into MP3 format (the so-called "ripping" operation), systematising the material's storage and access, providing useful metadata, and setting up an appropriate content directory.

Although a number of programs for ripping CDs and organising collections exist, most of them appear bloated and resource hungry. I decided to handle the task flexibly by piecing together existing tools and see where this approach would lead me. I ripped CDs by piping the output of an audio extraction program into an MP3 encoder. I encoded most material at 192kbps; some older historic CDs were of such low quality that they could be encoded at 128kbps without audible problems (variable rate encoding was not universally supported at the time of the ripping operation). The ripping script saved each track into a separate file named `track`*NN*`.mp3` and also created an text file containing the CDs track information. To increase the throughput of the ripping process I used the script shown in Figure 6 to rip each CD into a directory named with an ascending integer number. After each CD was ripped, the disk was ejected and the script would wait for a new disk to be inserted. Thus, whenever we would see an ejected CD in the tray we could just feed the furnace with a fresh CD.

```
#!/bin/sh
NUM=50
while :
do
     NUM=`expr $NUM + 1`
     mkdir $NUM
     cd $NUM
# Loop waiting for a disk to be inserted
while cdcontrol -f /dev/acd0c status media | grep -q '^No media'
do
            sleep 60
done
     cdrip
     cd ..
done
```

Figure 6: Automating the ripping process.

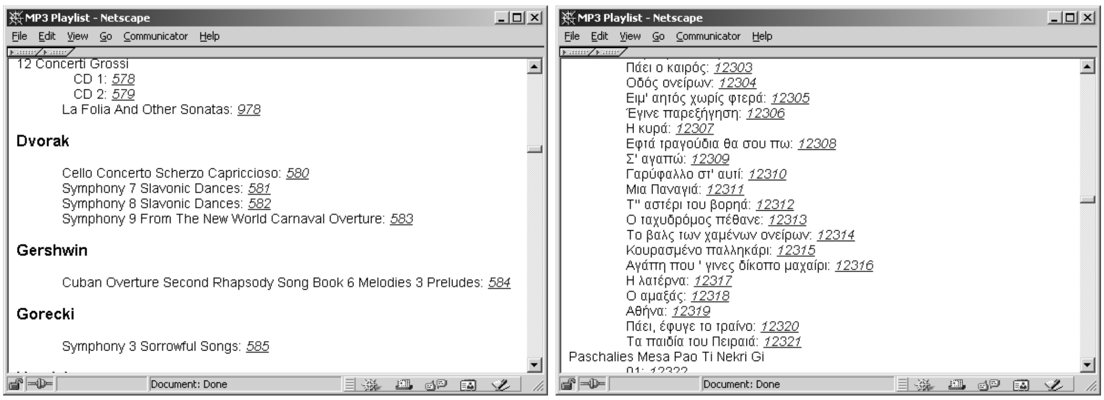

Figure 7: CD and track-level HTML playlists.

At periodic intervals we would move the ripped directories into a hierarchical structure made up out of the music type, the composer, performer, or band name, the album name, and the CD number. Also a script would crawl the directory structure and create a metadata file for each CD (`info.txt` by pulling information out of the *cddb.com* (and later the *freedb.org*) server using the *CDDB* Perl module [6]. I decided to store the metadata into a separate file and not use the ID3 standard, because the type of data available from the public CD directories does not exactly match that maintained in the MP3 ID3 structures. A separate Perl script crawls the content directories gathering metadata and creating the content directory in plain text, HTML, and LaTeX file formats. Each CD is identified by a three digit number and each individual track is identified by a five digit number (Figure 7). These numbers are again stored in one plain-text file (`index.txt`) for each CD. The numbers increase monotonically as new CDs are added and are never reused thus providing numbering persistency, so that bookmarks and music collections are not rendered invalid when CDs are added or deleted. The CD and track identification numbers are also needed for selecting a particular CD or track using a simple remote control. I reserved two-digit numbers for creating bookmarks to particular songs, and single-digit numbers for identifying a music type (e.g. Rock, Jazz, or Classical) from which an MP3 player would randomly shuffle tracks. The last option proved to be the most popular. The plain-text file forms the track database. It simply contains track and CD identification numbers as comments, followed by the respective file name:

```
# 521
# 10297
/vol/music/Classical/Bach/FrenchSuites/cd1/track01.mp3
# 10298
/vol/music/Classical/Bach/FrenchSuites/cd1/track02.mp3
# 10299
/vol/music/Classical/Bach/FrenchSuites/cd1/track03.mp3
```

This format allows simple *sed(1)* scripts to select data based on a CD or track identification number and feed the results directly as a playlist to MP3 players such as *mpg123* [18] and *mad* [25].

The first MP3 player connected to the information furnace was a network computer. In 1997, Digital Equipment Corp. (now part of Compaq Computer Corp.) produced the DIGITAL Network Appliance Reference Design (DNARD), and published the hardware specifications for free use. DEC used the code-name “Shark” to refer to these NCs—probably due to the plastic fins used to make them stand in an upright position (Figure 8). The DNARD exploits the power of the StrongARM microprocessor combined with the flexibility and economy of industry standard busses and chips. With the sale of Digital Semiconductor to Intel in early 1998, ownership of the StrongARM passed to Intel. Since that time, the DNARD design has no longer been supported by Digital or Compaq. Using however, a DNARD as an MP3 player connected to the information furnace was an attractive proposition, because of the DNARD’s attractive slim design, silent operation (it does not contain a disk or a fan), infrared port, and audio hardware. The Shark runs NetBSD 1.5 patched with Mark Foster’s AV package to support the audio hardware

Figure 8: The Digital Network Appliance Reference Design—DNARD (left) and the Shark as an MP3 player (right).

and the infrared port [13]. The Shark gets its initial configuration from the information furnace *dhcpd(8)* server and boots using TFTP [12, 40]; it subsequently mounts its file systems and the MP3 disk volume over NFS. A small shell script, run at startup time, allows us to use a remote control to select music. The *irw* command from the LIRC distribution [1] reads remote control messages. These can be a number forming a CD, track, or music-type code, play, stop, previous, next, or pause. The play command starts an MP3 player process. All other commands are handled by sending signals to the MP3-player process: stop kills the player process; pause pauses it; previous and next send it the USR1 and USR2 signals respectively. I contributed patches to the MAD and *mpg123* development efforts that enable both players to recognise these signals to move backwards and forwards in the current playlist. Playlists are generated by a *sed(1)* command that prints the master playlist from the music part selected until its end. As music is sorted and traversed according to its content, when the player finishes the selected track or CD, it will continue playing roughly similar content. Shuffling of music tracks is simply accomplished using the NetBSD *shuffle(1)* command.

The two other MP3 players we deployed use similar concepts, but run on less polished hardware and software configurations. One consists of an Intel 100MHz Pentium PC that boots a copy of FreeBSD diskless from the information furnace using etherboot [17]; the other is an old laptop running SuSE Linux 7.0. To minimise the noise of the PC I switched its fans to 5V, reasoning that with no hard disks its power demands were orders of magnitude lower than its rated capacity. Having the information furnace utilise simple standards for organising and disseminating the content (a text index and MP3 files exported via NFS as a directory tree) allowed me to choose the operating systems opportunistically: I selected FreeBSD to avoid the burden of configuring, maintaining, and provisioning disk space for another operating system (the player shares the read-only partitions of the information furnace), and SuSE Linux because it was the first OS installation to run correctly on the laptop's idiosyncratic hardware.

5.5 Security and Availability

The information furnace is secured in a place that is not easily accessible, continuously performing a number of critical functions. For these reasons it is imperative that it runs unattended and recovers gracefully after any problem. First of all, I configured the machine's BIOS to let the machine boot without a keyboard and mouse. However, the most important part of the machine's reliability is the correct installation of a UPS. I installed an additional serial card to connect the machine to the UPS. The sharing of interrupts for serial cards is a notorious problem. Thankfully, the serial card I used was a 1985-vintage ISA card, build around standard TTL and controller chips. It was therefore relatively easy to follow the traces from the motherboard connector to the DIP switch and, by cutting a trace and a bit of soldering, rewire the interrupt line to a different, free PC interrupt.

More tricky was the software configuration of the UPS, a task that has not yet been solved to my satisfaction. To power-down a UPS after its batteries signal that they are close to empty by expecting it to switch-off when the batteries are completely drained is incorrect due to a number of subtle race conditions. Consider first of all the following innocent scenario:

1. Mains power is interrupted
2. Computer is now powered by the UPS
3. UPS batteries signal a low condition
4. Computer gracefully halts
5. UPS dies as batteries are completely drained
6. Computer switches off as UPS power is interrupted
7. Mains power is restored
8. Computer restarts

Consider now the first race condition: mains power is restored between steps 4 and 5. The UPS will restore power and the computer will wait idly in its halted state. One can counter, that many computers have automatic power management so that (in step 4) they can be shut off instead of halted; when power is restored the computer will correctly restart. Enter now the second race condition: power is restored *during* the shutdown sequence (this sequence can last for several minutes on servers running database applications). The computer will now complete its shutdown sequence and switch itself completely off despite being fed with mains power.

How can one handle these problems? The communication protocol of most UPSs supports a software command to switch-off the UPS. Thus the last action of step 4 is to soft switch-off the UPS (and consequently the computer) if the UPS is running on batteries, or restart the computer if the UPS is at that point running on mains power. If the UPS is switched-off, when power is restored both the UPS and the computer will correctly restart. Note that the implementation of this sequence is not trivial: the excellent NUT [22] UPS software I used, operates as a user process; when the computer is ready to halt, user processes have died and filesystems are unmounted making it difficult to send that last command to the UPS.

The information furnace acts as an internet router and as a firewall by means of the native FreeBSD user-mode *ppp(8)* package running with network address translation (NAT) enabled. This approach while not perfect is adequate for the profile of the users living inside the firewall. Configuring the filters was relatively easy, once I had reference [50] at hand. Despite my earlier thoughts to the contrary, I found that protecting a dial-up connection can be worthwhile. I do not have time to maintain the various MP3 players with the latest security patches and, as the following excerpt from the information furnace's *apache* log shows, dial-up connections *are* actively scanned for security holes:

```
[Tue Sep 18 20:35:49 2001] [error] [client 195.158.192.25]
File does not exist: /usr/local/www/data/scripts/..\xc1\x9c
../winnt/system32/cmd.exe
```

6 Facilitators and Roadblocks

In the previous section we saw that the concept of the information furnace is indeed realisable, if only with a subset of the functionality we prescribed in Section 4. Here I will describe the most important factors that facilitated and hampered the development and are likely to affect future similar endeavours: open-source software, hardware standards, cost, and maintenance.

6.1 Open Source Software

Clearly, the most important aspect that affected the development was the availability of open-source software. The information furnace and its appendages were based on three different open-source operating systems. The stability and clear structure of FreeBSD provided the platform for the main unit, NetBSD with its multiple architecture support was at the time the only OS that supported the Shark's StrongARM architecture, while the aggressive development model of Linux resulted in an installation procedure and the existence of device drivers that could revive an old laptop as an MP player. The existence of these systems in source form allowed me to easily write and add a device driver to support the PCL-724 I/O card under FreeBSD, and Mark Foster to patch NetBSD to provide audio and infrared support. A number of times I found myself going over the source code to verify elements that were not clearly documented—documentation can not possibly cover everything. Some early failed experiments for diskless-booting the Shark were based on an old Linux platform; by comparing the NFS implementation of the Linux version I was using with that of the Shark's NetBSD I quickly found out that the configuration would never work since the two were supporting different versions of the NFS protocol.

No less important were the various add-on packages I used. In some cases I experimented with more than one package for a given task. It was clear that the co-existence and evolution of competing packages created evolutionary pressure that resulted in better overall offerings. A clear example of this case was the area of MP3 encoders and decoders. The Shark, with its StrongARM processor lacking floating-point support, is a tough platform for MP3 decoders. I fortunately was able to choose and test several different packages until I settled for the MAD MP3 decoder; the only one that run successfully on the Shark. A counterexample was the *vgetty* package: as far as I could determine it is the only viable offering for handling voice modems, and it has a lot of room for improvement. At the start of the project I was somewhat ambivalent on binary and package distributions. However, I found that being able to quickly try packages without having to go through the configuration and manual compilation process outweighed the opacity problems of this distribution process.

6.2 Standards and Costs

The existence of open standards proved to be a blessing for the project's success; the lack of standardisation a curse. Specifically, the lack of open standards ruled out having the information furnace controlling the home's heating in the form I described: the heating controller was clearly attached to a form of a network bus, but its operation at all the network stack levels was apparently a secret closely guarded by its manufacturer (i.e. the standard was not available on the web). Similarly, in the domain of artificial lighting controls, it being an area where a number of incompatible proprietary standards compete, there are expensive solutions that do not deliver the economies they could. Efforts for integrating arbitrary communication protocols such as [23] could help, so would adopting TCP/IP for communicating with all devices [11].

On the other hand, the standards and the resulting economies of PC manufacturing coupled with the rapid obsolescence of PCs provided me with a number of cheap and viable platforms for deploying the infrastructure I described. While scavenging obsolete hardware can be a viable strategy for a researcher or a hobbyist, it can not form a long-term technology adoption plan. However, the above forces can also result in the development of affordable hardware platforms based on established components and processors like the DNARD Shark. These platforms, based on cheap industry-standard busses and chips can form the base of the future's mass-produced information furnaces.

6.3 Deployment and Maintenance

Perhaps the biggest roadblock to the universal deployment of information furnaces is their installation, testing, and maintenance. The subtle interactions of multiple systems can result in many subtle and difficult to find bugs. Some of them can be amusing: the first visitors to ring the doorbell after the information furnace was deployed were greeted with an telephone answering machine message! More than a year after the furnace's deployment we are still tuning its operation and correcting (minor by now) inconveniences.

Most people would not regard the existence of a resident system administrator an acceptable solution to this problem. The availability of stable software (rather than its organic home-growth), the adoption of the domain-specific languages we saw in Section 5.2, and the initial configuration of the furnace by a qualified professional can help in this direction. However, the above process, although similar to other processes followed for building homes, is completely different from the *ad hoc* procedure typically employed when purchasing and deploying consumer-oriented hardware (plug it in, play with the buttons, avoid reading the manual). Unless the widespread deployment of information furnaces is coupled with an appropriate installation and maintenance process, significant problems will ensue.

7 Conclusions

There is tremendous scope for making the devices found in a modern home more user-friendly and synergistic. The isolated location, all-inclusive scope, and multi-modal interface attributes of the information furnace offer a potential roadmap for achieving these goals. By implementing a prototype I discovered the pivotal role that open-source software, standards, and the deployment process will play in such an endeavour.

Some may counter that my thesis for a centralised information furnace contradicts the proposed move from a complex general purpose personal computer towards user-friendly simple, and versatile "information appliances" [32, 48]. I can defend my position on two grounds. Firstly, the systems my proposed information furnace is set to replace do not exhibit any of the information appliance design axioms: simplicity, versatility, pleasurability [32, p. 67]. Secondly, my solution, although based on personal computer technology, does not entail (at least in the form I designed it) the two damning characteristics of PCs: creeping featurism and an application-oriented mindset [32, pp. 80–87]. I propose that application furnaces be individually configured by experts to match the needs to a home's occupants in the same sense as the house itself is architected. Secondly, the information furnace I propose is in fact an information, appliance albeit one with a rather large scope: to integrate the home's control, information, and communication systems. This integration aspect—necessary to exploit the synergies I discussed—is the diametrical opposite to the PC's "one application for each task" design philosophy.

While my prototype implementation proves the concept, its piecemeal implementation by a single developer has resulted in a wanting (to put it politely) software architecture. If the information furnace concept is to be widely adopted major architectural challenges have to be overcome. Already, research approaches such as *iRoom* [14], demonstrate how the task of developing such an architecture could be approached. The software architecture of a consumer-oriented information furnace should: be extremely reliable, allow installator and end-user customisation, provide means for interfacing to many different proprietary devices, and integrate the above with a modular, multi-modal, and user-friendly interface. What is not needed is a repeat of the PC usability and reliability debacle in a scale that will affect our entire family, lives, and home.

Acknowledgements

Compaq Research contributed (as a prize of the 2000 Usenix technical conference "win a pet Shark contest") the Digital Network Appliance Reference Design—DNARD that I used as the system's first MP3 player. Jeffrey Mogul kindly handled the tricky logistics for distributing the contest's Sharks and saved the day by explaining to me how a keyboard could be essential for its operation. Eliza Fragkaki contributed the server's processing unit, literally provided a helping hand during the CD ripping operation, and patiently endured the prototype system's alpha and beta testing period. Lorenzo Vicisano came up with the idea of using the Shark as an MP3 player, Isidoros Kouvelas and Vasilis Prevelakis offered encouragement, help, and interesting ideas during the prototype's implementation, and Giorgos Gousios contributed valuable comments on an earlier draft of this paper. Finally, my colleagues at the Athens University of Economics and Business eLTRUN research group provided me with numerous opportunities to enrich my view and expectations of ubiquitous computing appliances and applications.

Software Availability

The source code for the PCL-724 device driver and the Shark MP3 player script is available at <`http://www.dmst.aueb.gr/dds/sw/ifurnace`>.

References

[1] Christoph Bartelmus, Pablo d'Angelo, Heinrich Langos, Tom Wheely, Karsten Scheibler, Jim Paris, Pawel T. Jochym, and Milan Pikula. LIRC: Linux infrared remote control. Online. http://www.lirc.org/, 2002. Current March 2002.

[2] Gordon Bell and Jim Gray. Digital immortality. *Communications of the ACM*, 44(3):28–30, March 2001.

[3] J. Bell, F. Bellegarde, J. Hook, R. B. Kieburtz, A. Kotov, J. Lewis, L. McKinney, D. P. Oliva, T. Sheard, L. Tong, L. Walton, and T. Zhou. Software design for reliability and reuse: a proof-of-concept demonstration. In *Conference on TRI-Ada '94*, pages 396–404. ACM, ACM Press, 1994.

[4] Stewart Benedict. X-automate. *Linux Journal*, 1999(57es):7, 1999.

[5] T. Berners-Lee and D. Connolly. RFC 1866: Hypertext Markup Language — 2.0, November 1995.

[6] Rocco Caputo. CDDB—high-level interface to the internet compact disc database. Online. http://www.cpan.org/modules/by-module/CDDB/, 2002. Current March 2002.

[7] Goran Devic. Home entertainment linux MP3 player. *Linux Journal*, 2000(71es):8, 2000.

[8] Gert Doering. Mgetty+sendfax archive/documentation centre. Online. http://alpha.greenie.net/mgetty/, 2002. Current March 2002.

[9] Amitava Dutta-Roy. Networks for homes. *IEEE Spectrum*, 36(12):26–33, December 1999.

[10] Marc Eberhard. Vgetty documentation center. Online. http://alpha.greenie.net/vgetty/, 1998. Current March 2002.

[11] Robert E. Filman. Editor's introduction: Embedded internet systems come home. *IEEE Internet Computing*, 5(1):52–53, January / February 2001.

[12] R. Finlayson. RFC 906: Bootstrap loading using TFTP, June 1984.

[13] Mark J. Foster. AV: An audio/visual equipment device driver for NetBSD. Online. ftp://ftp.talix.com/pub/av/, 1999. Current March 2002.

[14] Armando Fox, Brad Johanson, Pat Nanrahan, and Terry Winograd. Integrating information appliances in an interactive workspace. *Computers Graphics and Applications*, 20(3):54–65, May/June 2000.

[15] Robert Fulghum. *All I Really Need to Know I Learned in Kindergarten: Uncommon Thoughts on Common Things*. Ivy Books, reissue edition, 1993.

[16] Fotis Georgatos and Annie Pinder. Coffee-HOWTO. Online. http://www.linuxdoc.org/-HOWTO/mini/Coffee.html, 2000. Current March 2002.

[17] Markus Gutschke, Gero Kuhlmann, Jamie Honan, Martin Renters, Bruce Evans, Rob de Bath, and et al. Etherboot open source code for creating boot ROMs. Online. http://etherboot.sourceforge.net/, 2002. Current March 2002.

[18] Michael Hipp. mpg123: A fast MP3 player of Linux and Unix systems. Online. http://www.mpg123.de/, 2001. Current March 2002.

[19] Tord Jansson. bladeenc MP3 encoder. Online. http://bladeenc.mp3.no, 2002. Current March 2002.

[20] Stephen C. Johnson and Michael E. Lesk. Language development tools. *Bell System Technical Journal*, 56(6):2155–2176, July-August 1987.

[21] Jan Kasprzak. Modem::vgetty perl module. Online. http://www.cpan.org/authors/id/Y/YE/YENYA/, 1998. Current March 2002.

[22] Russell Kroll. Network UPS tools. Online. http://www.exploits.org/nut/, 2002. Current March 2002.

[23] Markus Lauff and Hans-Werner Gellersen. Adapation in a ubiquitous computing management architecture. In *Proceedings of the 2000 ACM symposium on Applied computing 2000*, pages 566–567. ACM Press, 2000.

[24] Giorgos Lekakos, Kostas Chorianopoulos, and Diomidis Spinellis. Information systems in the living room: A case study of personalized interactive TV design. In *Proceedings of the 9th European Conference on Information Systems*, Bled, Slovenia, June 2001.

[25] Robert Leslie. MAD: MPEG audio decoder. Online. http://www.mars.org/home/rob/proj/mpeg/, 2002. Current March 2002.

[26] Tsugio Makimoto, Kazuhiko Eguchi, and Mitsugu Yoneyama. The cooler the beter: New directions in the nomadic age. *Computer*, 34(4):38–42, April 2001.

[27] Matsushita Electric Industrial Co., Osaka, Japan. *Panasonic Electronic Modula Switch System Modem KX-T206: Installation Manual*, 1993. Document code PSQX1158YA KW0796KM1027.

[28] David L. Mills. RFC 1305: Network time protocol (version 3) specification, implementation, March 1992.

[29] Monty. Cdparanoia—an audio CD reading utility. Online. http://www.xiph.org/paranoia/manual.html, 2002. Current March 2002.

[30] Elizabeth Mynatt, Douglas Blattner, Meera M. Blattner, Blair MacIntyre, and Jennifer Mankoff. Augmenting home and office environments. In *Proceedings of the third international ACM conference on Assistive technologies*, pages 169–172. ACM Press, 1998.

[31] Donald A. Norman. *The Psychology of Everyday Things*. BasicBooks, New York, NY, USA, 1988.

[32] Donald A. Norman. *The Invisible Computer*. MIT Press, Cambridge, MA, USA, 1998.

[33] Catherine Plaisant and Ben Shneiderman. ON-OFF home-control devices: Design issues and usability evaluation of four touchscreen interfaces. *Interacting with Computers*, 3(1):9–26, 1992.

[34] Larry Press. Personal computing: the post-PC era. *Communications of the ACM*, 42(10):21–24, October 1999.

[35] J. Christopher Ramming, editor. *USENIX Conference on Domain-Specific Languages*, Santa Monica, CA, USA, October 1997. Usenix Association.

[36] Jef Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley, 2000.

[37] Conrad Sanderson. Notlame: LAME command-line front end. Online. http://hive.me.gu.edu.au/not_lame, 2002. Current March 2002.

[38] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer-Interaction*. Addison-Wesley, third edition, 1998.

[39] Siemens Building Technologies. *Room Units for Use with Heating Controllers: QAW70*, June 1999. Document code CE2N1637E.

[40] K. Sollins. RFC 1350: The TFTP protocol (revision 2), July 1992.

[41] Diomidis Spinellis. Palmtop programmable appliance controls. *Personal Technologies (Personal and Ubiquitous Computing)*, 2(1):11–17, March 1998.

[42] Diomidis Spinellis. Reliable software implementation using domain specific languages. In G. I. Schuëller and P. Kafka, editors, *Proceedings ESREL '99 — The Tenth European Conference on Safety and Reliability*, pages 627–631, Munich-Garching, Germany, September 1999. ESRA, VDI, TUM, A. A. Balkema.

[43] Diomidis Spinellis. Notable design patterns for domain specific languages. *Journal of Systems and Software*, 56(1):91–99, February 2001.

[44] Diomidis Spinellis and V. Guruprasad. Lightweight languages as software engineering tools. In Ramming [35], pages 67–76.

[45] Thomson Consumer Electronics, Indanapolis, USA. *GE Digital Answerer*, 1996. Document code 2-9865.

[46] Marcello Urbani. dagrab: Read audio tracks from a CD into wav sound files. Online. http://web.tiscalinet.it/marcellou/dagrab.html, 2000. Current March 2002.

[47] Larry Wall, Tom Christiansen, Randal L. Schwartz, and Stephen Potter. *Programming Perl*. O'Reilly and Associates, Sebastopol, CA, USA, second edition, 1996.

[48] Roy Want and Gaetano Borriello. Survey on information appliances. *Computers Graphics and Applications*, 20(3):24–31, May/June 2000.

[49] M. Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):74–84, October 1993.

[50] Elizabeth Zwicky, Simon Cooper, and D. Brent Chapman. *Building Internet Firewalls*. O'Reilly and Associates, Sebastopol, CA, USA, second edition, 2000.