Compatibility Issues between Services supporting Networked Appliances

Mario Kolberg, Evan H. Magill, Michael Wilson

Department of Computing Science and Mathematics
University of Stirling, Stirling FK9 4LA
United Kingdom
{mko,ehm,mew}@cs.stir.ac.uk

In the near future general household appliances such as televisions, refrigerators, alarm clocks, stoves, and even lights, will be supplemented with a network interface connecting the appliance to the Internet. Homes are being equipped with such networked appliances to allow a more convenient way of living. Such extensive automatic control of appliances leads to the concept of the *smart home*.

Behind such automation, there is a lot of software controlling the appliances. This software, which is often referred to as *services*, *applications*, or *bundles* is supplied by a range of service provider businesses. Hence in a single home, appliances may be controlled by a multitude of services, which are offered by a wide variety of different providers. Moreover, some services may require the use of other services. Importantly, these businesses are completely independent and may not even be aware of one another or their products. Hence appliances may be controlled by more than one service, and indeed these controlling services are often trying to achieve different goals. This causes compatibility issues, which need to be resolved for networked appliances to be successful in the mass market. This problem is well known in telephony and historically is referred to as the feature interaction problem.

This paper discusses the issue of compatibility between services in a home environment. Reasons why *and* how services interact are discussed, and a taxonomy of interactions is presented. Finally, an approach is presented which prevents interactions. The approach presented uses accepted and known device and protocol interworking techniques. Throughout the paper, a number of example scenarios are used to illustrate the issues. However, the emphasis of the paper is not only to present sample services for controlling home appliances or identifying specific interactions between such services, but on finding a *general* solution to the feature interaction problem that can *automatically* detect interactions between services in the home.

1. Introduction

1.1 Services for Networked Appliances

Networked appliances are attracting an increasing interest and the first selected products are now available off-the-shelf (e.g. IBM Home Director). Networked appliances are dedicated consumer devices which contain at least one network processor. As such, an increased number of networked appliances may assume a networked home with an always-on Internet connection. Examples include fridges, stoves, lights, curtain drawers (puller), security cameras, stereos, and TVs. A number of industrial trials have been conducted, such as the OnStar at Home trial launched by the Internet Home Alliance and involving over 70 homes in the Chicago area, the Telia Connected Home initiative in Europe, the Net@Home trial by France Telecom and Thomson, and Philips' HomeLab in Eindhoven, the Netherlands.

Even though some networked appliances offer additional functionality, they are largely expected to behave in a similar manner to their traditional counterparts. While some appliances may

contain embedded software inside the device, significant added value originates from the possibility of being networked, which allows for interworking between appliances and intelligent control by means of external software services. It is expected that services control a number of appliances and thus are rather rich in functionality. However, not all service features may be enabled and so services may be offered with various levels of functionality. For instance, a security service provides simple burglar alarm functionality. However, for a premium, the service will also record the picture from the entrance camera, or even switch on lights and the stereo at random times to give the impression to passers-by that somebody is at home. In other words, rather than features associated with a single appliance (e.g. software for a washing machine), the emphasis in this paper is on third party services controlling multiple appliances and offering value added functionality to the customer.

Commonly the provisioning of appliances and services is separate and introduces more flexibility to the market place. Thus consumers may buy appliances from a number of suppliers and also subscribe to services from a variety of providers. However, it is also expected that there will be packages offered to customers, comprising of appliances *and* services.

Added value will come from appliances cooperating. Consequently, services will be communicating with other services, either directly or implicitly via the controlled appliances. For example, both the home climate service and the security service may control windows. Most interworking is beneficial, however, not all interworking is desirable.

Imagine the following scenario in a typical home: During their summer vacation, the home owner sets the burglar alarm service to protect their home from intruders and the home climate control service to consume as little energy as possible as nobody is at home. The burglar alarm service monitors the state of the house, e.g. it detects movements in the house and monitors for the use of appliances. If something suspicious is detected, the police are notified. The climate control service controls the heating of the house, the air-conditioning, the window blinds and also the windows. The cheapest way of lowering the temperature in the house is to lower the blinds during the day and to open the windows in the evening. However, this movement will be registered by the security service. Consequently, when the climate control service lowers the blinds or opens the windows, the burglar alarm service will be triggered resulting in a call-out. Obviously, this behavior is undesirable.

1.2 Service Interactions

From the experiences in the telephony environment it is known that the interworking of services can lead to a phenomenon called *feature interactions* [1]. Services which work perfectly in *isolation* do not exhibit the desired behavior if other services are active in the same session or call. The services involved can be distributed in the network and be active on different endpoints. While particular interactions may be benign, in general, interactions can be extremely damaging to system performance and user expectations.

The term feature originates from telephony where it refers to a component of additional functionality - additional to the core body of software. For the purposes of this paper, a *service* is a piece of stand-alone software which is provided to the user. Services may contain a number of (optional) *features* which extend the basic functionality of the service.

Interactions may occur at any point during a service. This means that with an increasing number of services, there is a combinatorial explosion in the number of scenarios with the potential for interactions. In general, neither manual inspection, nor simple testing offer tractable solutions. More effective approaches addressing the special requirements of the networked appliances domain are needed. Commonly, when a feature interaction is known it is relatively easy to mend the problem. Hence the hard problem is to *automatically* detect *unknown* interactions. Thus the interaction scenarios given in this paper should only be understood as examples to illustrate the problem and not taken as an exhaustive list.

Services may interact with each other but also features may interact within a single service. For instance, a security service which monitors movement in the house to detect intruders, may also contain the functionality discussed above of switching on lights and the stereo to deter intruders. This will trigger appliances in the house, such as lamps, curtain drawers, or audio/visual equipment to give the impression somebody is at home. Movements created in this way may be picked up by the main component of the security service and an alarm may be raised. Clearly, this type of problem needs to be solved by the service vendor or provider. In this paper we refer to this problem as an *intra-service* interaction. This type of interaction is caused by the fact that services are frequently developed out of independent components. If these components are not correctly integrated they may compromise each other. Interactions between two independent services are referred to as *inter-service* interactions. While examples are provided of intra-service interactions, they are not the emphasis of this paper. Intra-service interactions can be detected during testing within a single organization. However, inter-service interactions cannot and are thus much more challenging, and this is the focus of this paper.

Previous work in the telephony domain [2, 3] has shown that feature interactions are hard to detect, and even harder to resolve. It is important to note that feature interactions are not about badly written services or software bugs, but conflicting service actions. Services are usually developed in isolation, i.e. independently of each other by separate businesses. Thus, services will 'meet' for the first time in the network. Thus, the paper will focus on the issue of service and device interaction, issues such as security and user focus are not addressed in this paper.

2. Expected Architecture for Networked Appliances

2.1 Location of Services

The overall architecture can be divided into two parts: firstly, the user domain or the home domain and secondly, the service provider domain or outside world. These two domains are depicted in Figure 1. Appliances inside the home are connected to a Local Area Network (LAN). Appliances communicate using various protocols, such as HAVi (Home Audio/Video Interoperability) [4], MHP (Multimedia Home Platform) [5], Jini [6], UPnP (Universal Plug and Play) [7], and X-10 [8].

The home is connected to the service provider domain. Various service providers offer their services in this domain, to be used with appliances inside the home. The domains are connected together by a Residential Gateway (RGW) with services such as Network Address Translator / Firewall functionalities. These provide protection from unauthorized access into the home and also a translation of addresses in the IP format to the format used inside the home. Currently, more sophisticated approaches such as the Open Services Gateway[9] by the Open Services Gateway Initiative (OSGi), are being specified, which allows services to be managed (installed,

started, stopped and updated) remotely and without the need to restart the gateway. Security is a prime concern of the initiative.

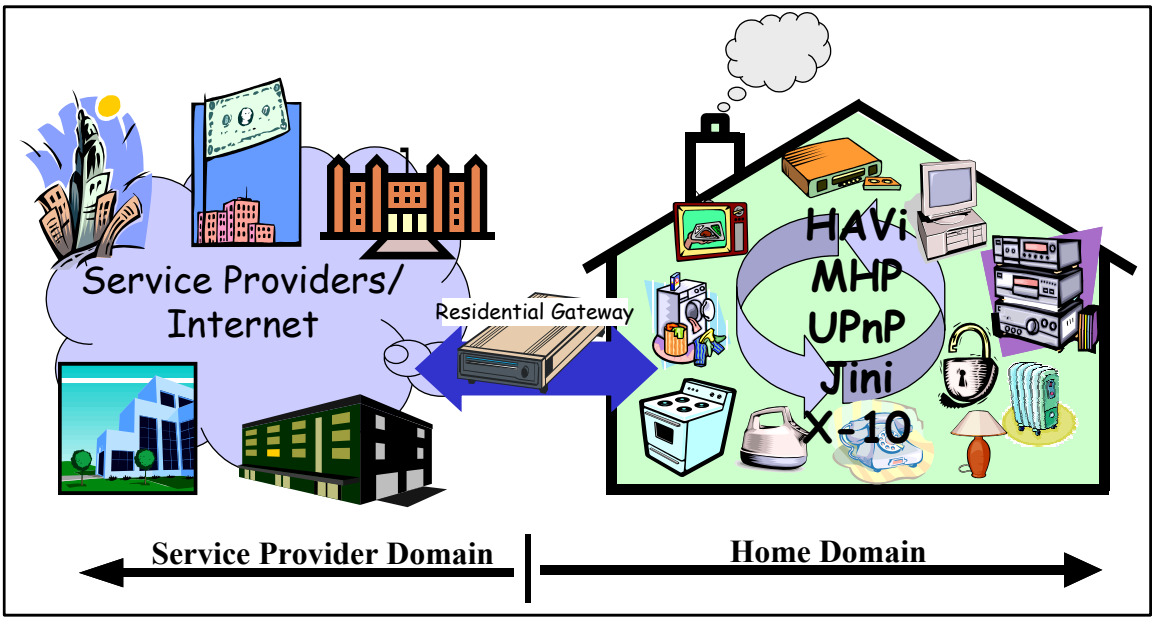


Figure 1: Home Domain and Service Provider Domain.

Services controlling appliances may be located in at least three different locations. The most likely location is in the Residential Gateway. Here, the OSGi Gateway provides a platform for service delivery and has specifically been designed to host services. Secondly, services may be located in the provider domain. As there will be a number of providers operating, a single home may be served by a number of providers, each supplying one or more services. A third likely option is that there also will be services residing within the home domain on devices. In fact, with more complex appliances, some service functionality will be contained in the appliance itself and the service will be very hard to split from the appliance. Whether services are supplied with an appliance will depend on its complexity, for instance, a networked washing machine will contain certain logic whereas a bedside lamp will not.

It is important to note that these options are not exclusive. In a typical home, services controlling the appliances may be deployed using all three options. The following section lists examples of services for networked appliances.

2.2 Services for Networked Appliances

As was already discussed in Section 1, bare networked appliances offer limited added value. It is the services driving these appliances which extend the functionality of the appliances. However, basic services controlling only a single appliance do not offer much added value compared to traditional appliances, and may not justify the investment. Thus services are likely to be powerful, integrating the control of multiple appliances. Service providers may extend the functionality of a service for additional charge. Hence some services are expected to consist of a number of components which can (optionally) be activated.

As the list of services below illustrates, services for appliances cover a wide spectrum [10]. Here, emphasis is given to the aspects concerned with controlling appliances.

Home Climate Control (HVAC) Service: This service integrates the control of the heating and air-conditioning. A thermometer is polled at regular intervals and transmits the current temperature to the service. If the temperature in the house rises to a certain level, the service will start the air-conditioning. If the temperature drops below a certain level the service starts the heating. The service also offers the option of energy efficient climate control. Here, the service includes the control of windows (open/close), an air fan, and also window blinds. The service has the ability to check when opening a window will affect the inside temperature in the desired way, i.e. the service is aware whether it is warmer or colder outside.

Power Control Service: The power control service enables the subscriber to get cheaper electricity by giving up some control to the power utility by using a separate on-switch. With this separate power switch, the user specifies that the appliance is ready to run, however, it is the power utility which actually switches the appliance on. This allows for the power utility to determine the exact time the appliance is used and hence to manage power consumption during peak demand. The service also offers an option of running the house efficiently. During these times, energy is saved by switching off unnecessary appliances, such as lamps and TVs.

Home Entertainment Service: This service controls entertainment devices in the home, such as the TV, stereo system, and VCR/DVD players. Pre-set TV programs may be recorded on the VCR.

Security Service: This service integrates various security and safety aspects. The basic service is of burglar alarm type functionality. It works together with movement sensors in the home. If any of the sensors detect any movement, the service is triggered. The service then records the picture from a security camera on the VCR and calls the police. An option for the security service is the "away from home" functionality. This aims at giving the impression that the house is occupied during absence of the owners. The service will turn on appliances, such as lamps or curtain drawers, at random or pre-set times.

Communications Support Service: This service supports the use of email and the telephone. The user can be notified by a message displayed on the TV when a phone call or an email arrives. The message will contain the caller for a phone call and the sender and subject line for an email. Optionally, users may subscribe to a feature which turns down the volume of the TV and the stereo when a phone call arrives.

3 Service Interactions

3.1 Interaction Examples

Interactions can occur for a number of reasons. The most common are conflicting goals and broken assumptions. Services pursue specific goals, e.g. to switch off unnecessary appliances to save electricity. Another service may start the TV or switch on lights to pretend the home is occupied. Clearly, the goals of both services clash. Similarly, services need to make certain assumptions about their environment. For instance, with a security service it is assumed that when set nobody is at home and hence no appliances should be used. However, another service may control the blinds preventing the sun from unnecessarily heating up the home. Here the assumption of the first service, that no appliances will be used, is violated by the second. Both conflicting goals and broken assumptions may result in inter-service and intra-service interactions.

Intra-service interactions should be dealt with during testing by the service provider. However, inter-service interactions are harder to deal with as they will only show up in a customer's home when services from different providers interwork.

The following section provides some example interactions between the services outlined above. Section 3.2 suggests a taxonomy to help to better understand the complexity of the problem. Section 4 introduces an approach to handle service interactions between services controlling networked appliances. From the list of services in the previous section, a number of example interactions due to conflicting goals are apparent. This includes both intra-service and inter-service interactions and both types are included in the following list of examples.

1. Imagine the situation where the owner of the house is absent and the **away from home** feature of the **security service** and the **environmental control** feature of the **power control service** are activated. The environmental control will switch off lamps and TVs, however, the security service will switch lamps and the stereo on to give the impression that somebody is at home. The problem are the two overlapping requests from the two services trying to control appliances in conflicting ways. This is an inter-service interaction.
2. The **security** and the **entertainment services** both try to control the VCR. Once triggered, the security service records the picture from a camera on the VCR. However, the entertainment service may try to record a TV show at the same time – disabling the security service. Unlike the former example where the first action (switching off lamps) was finished before the second occurred, the problem in this example occurs because the first action is not yet completed (recording the camera picture) when the second is triggered. Clearly, this interaction is also between two independent services, and hence of the inter-service type.
3. An intra-service interaction may occur within the **climate control service**. A thermometer is periodically polled for the current temperature by the service. If the temperature rises above a certain value and if the user has activated the energy efficient climate control option, the service may start the air conditioning *and* open the windows. Clearly, these two actions are not compatible as the open windows compromise the correct and efficient functioning of the air conditioning. This interaction is caused by the energy efficient component of this service, which will in fact waste energy if not correctly integrated.
4. An interaction with a series of events is caused by the **climate control service** and the **security service**. If movement in the house is detected, the security service is notified which triggers the alarm. The environmental control service may lower the blinds, start the ventilation fan or even open windows. All of these actions will trigger a movement sensor and consequently the security alarm. This is an inter-service interaction.
5. An infinite loop can be caused by an intra-service interaction within the **climate control service**. Assuming the service consists of a heating component controlling the heating and a cooling component controlling the air-conditioning, the following interaction may occur. Reaching a certain temperature will trigger the climate control service resulting in the air conditioning being activated. However, this may cause the temperature to drop below the pre-set temperature for the heating to be activated. The heating will increase the temperature, however, this again may result in the temperature being too high and the air conditioning being started again!

Many more examples can be constructed between the simple services presented here. This shows that interactions are a serious obstacle to the deployment of networked appliances. However, the examples above have been selected because they exhibit particular properties, and not because of their likelihood or nuisance value. A classification of interactions can be created

by grouping interactions according to their properties. The next section presents such a classification to help the understanding of the problem and to serve as a measure of the success of future approaches to detect or solve interactions.

3.2 A Taxonomy for Interactions between Services for networked appliances

The taxonomy for Internet Appliances is based on a classification for telephony feature interactions [11, 12]. All interactions listed above can be associated with one type of interaction as defined by the taxonomy. However, it is noted that this is an empirical study and so no claim of completeness can be made. An initial version of the taxonomy for networked appliances has been presented in [10].

Multiple Action Interaction (MAI)
This type of interaction occurs when two services are controlling the same appliance. Figure 2 shows this type of interaction, with *S* symbolizing a service and *A* an appliance. The diagram shows two services carrying out *actions* on the appliance. The number in front of *action* specifies the order. Instances for this type of interactions are examples 1 and 2 in the previous section. The second action request might arrive while the first one is still being processed (VCR example), or the first request is fully executed but the first service still assumes exclusive control over the appliance (switched off lights). All such examples are 'bad' interactions, they either compromise at least one of the services involved, or bring the appliance to an undefined state.

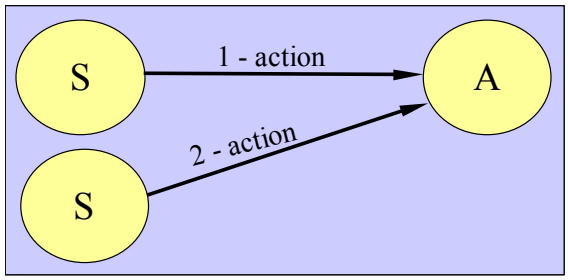


Figure 2: Multiple Action Interaction.

Shared Trigger Interaction (STI)
This type of interactions is populated by scenarios where an event is sent to two different services which perform conflicting actions, (cf. Figure 3, *trigger* means an appliance sending some triggering information to the services). The third example in Section 3.1 is such a case. Interactions belonging to this category are also 'bad'. However, their direct influence is restricted to effects on the environment. For example, the room temperature is wrong. They do not directly affect the operations of the services. Indirectly, however, they may cause an interaction. Importantly, this interaction may not be 'visible' in the network and can only be verified by the effects on the environment.

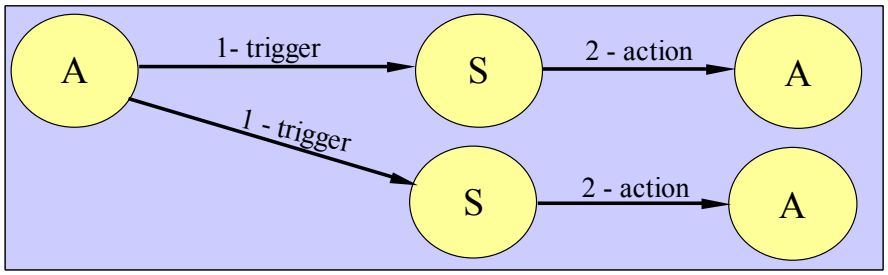
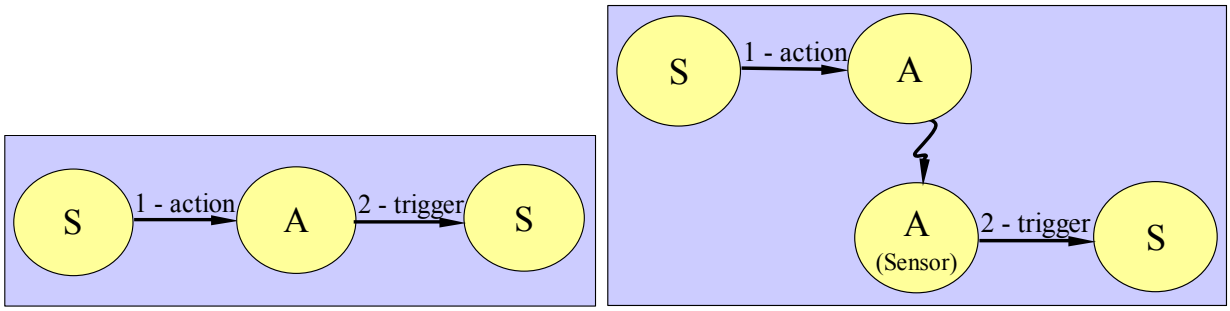


Figure 3: Shared Trigger Interaction.

With this type of interaction, a single condition (e.g. incorrect temperature) is treated. As the services controlling appliances are assumed to be rather powerful and are concerned with one particular aspect of the home (e.g. security, climate) it is unlikely that interactions of this type will occur between different services – unless the user has deployed, for instance, two climate control services. Interactions of this type are expected to occur between two features within a single service.

Sequential Action Interaction (SAI)
If a service sends requests to an appliance which causes the appliance to send notifications to yet another service, then a Sequential Action Interaction occurred (cf. Figure 4(a)). However, unlike the previous two categories, the trigger for the second service may also be caused by the environment. In other words, the appliance controlled by the first service does not need to communicate a trigger directly to the second service. If the appliance controlled by the first service changes the environment, the change may be picked up by a sensor and communicated to the second service. For instance, the fourth interaction in the list in Section 3.1 is an example of this category. The blinds are lowered by the climate control service. This creates movement and the movement sensor triggers the burglar alarm service (cf. Figure 4(b)).



(a) (b)

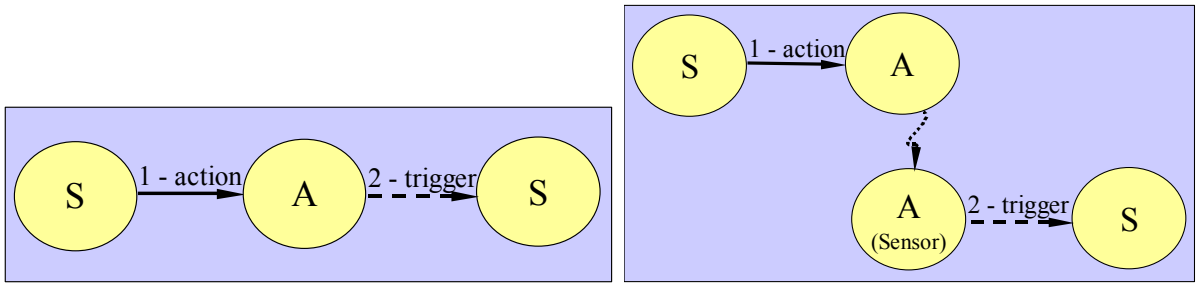
Figure 4: Sequential Action Interaction with direct trigger (a) and trigger via environment (b).

The fifth interaction in Section 3.1 is a special case of this type of interaction. This is because the actions of the climate control service create an endless loop and the optimal temperature will never be reached. In the looping case, two different values of the *same* variable trigger different actions. As services will control certain aspects of the house, it is likely that looping interactions will occur within one service, rather than between services. In short, looping interactions will commonly be intra-service interactions.

Sequential Action Interactions can be either 'good' interactions, i.e. the behavior is desired, or 'bad' interactions whose behavior is undesirable. However, except for the loop-type interactions, even 'bad' interactions in this category do not cause damage to the appliances or services involved. 'Bad' interactions of this type compromise the user's expectations.

Missed Trigger Interactions (MTI)
The final class of interactions are characterized by the fact that one service prevents a second one from operating. In other words, the first service performs an action on an appliance which prevents a trigger message being sent to the second service (cf. Figure 5(a); the dashed line symbolizes 'Not Sent'). As with the previous category, the appliance controlled by the first service does not necessarily need to be linked directly to the second service, but may change the environment and so only indirectly link to the second service. Thus, when an interaction ocurs, the first service sends a request to the appliance which has an impact on the environment. This

change will be picked up by a sensor and because of the changed environment the sensor will not trigger the second service (cf. Figure 5(b)). As with Sequential Action Interactions, the resulting behavior is not necessarily undesired. Missed Trigger Interactions may be desirable.



(a) (b)

Figure 5: Missed Trigger Interaction with direct trigger (a) and trigger via environment (b).

In fact, while this type of interaction occurs in the telephony context [11, 12], no example interaction that causes undesired behavior could be identified in the appliance context. There are, however, missed trigger interactions which do not exhibit undesirable behavior. For instance, the security service may lower the window blinds to give the impression that someone is at home. During a hot summer day this will prevent the temperature inside the home from rising to a level where the climate control service would switch on the air-conditioning. Thus the climate control service is not triggered because of the actions of the security service. However, the resulting behavior is not undesired. What has happened is that the security service causes an action, which has a side effect that also works towards the goal of the climate control service, namely keeping the temperature below a certain value. Unless services can completely disable appliances and thus prevent trigger messages from being sent, no other reasons for missed trigger interactions could be identified in the case study. Consequently, Missed Trigger Interactions are not considered further in this paper.

3.3 Summary of the Problem

From the previous section it can be seen that interactions in an appliance environment are a real issue. It needs to be tackled if the overall success of networked appliances is not to be jeopardized. Compared with a telephony environment where the number of services and users involved in an interaction is small (usually not more than two or three), the number of services and especially networked appliances affected by an interaction in an appliance environment can be much larger. While these interactions may occur between different services (inter-service) and also between features *within* a single service (intra-service), only inter-service interactions are considered further.

Requests delivered to appliances using the same message may be of two basic types: firstly, the request can be carried out almost instantaneously (open the door, turn on the light, etc), and secondly, the request will be carried out continuously over a certain length of time (record a TV show). These are quite different kinds of actions triggered by the same message.

Interactions in the telephony domain are detected during a call or a session. However, in an appliance environment no concept of a closed period of time exists. There is no closure. Furthermore, an appliance interacts with its environment, by either monitoring or changing it. For instance, a heater increases the room temperature, or opening a window creates movement. Unlike in telephony, where services only interwork and consequently interact by network

messages, in an appliance environment, services also interact via the environment. There are not always explicit network messages! Consequently, approaches applied in a telephony setting are not applicable here. Approaches need to consider *what* effect appliances have on their environment, to detect and avoid interactions.

It is important to note that the hard problem is to identify interactions, and even more so to *automatically* detect interactions. Once an interaction is known, generally, it can be fixed easily. The taxonomy has been used to better understand the problem in networked appliances. At the end of the paper it will be used to measure the effectiveness of the approach to detect and resolve interaction, which is presented in the following.

4. Avoiding Interactions between services for appliances

Feature interaction approaches targeted at the telephony domain have been service centric, they have focused on the *behavior* of the service controlling the resource [2]. The approach advocated here to handle interactions, employs a model which focuses on devices and their surrounding environment. It shows *what* devices do to their environment and thus allows interactions to be avoided. Inspiration is drawn from the Operating Systems (OS) and distributed computing domain, where concepts of locking and restricting access are commonplace.

As the services operating in the home may originate from a number of sources and may interwork in the home for the first time, the approach operates during run-time of the services. That is it constantly monitors the actions of services to prevent incorrect behavior. Therefore services must be able to interwork correctly. Although some protocols have used the environment for device communication, for example, EHS (European Home System) [13] and CEBus [14], they have not tackled the root cause of service (or feature) interaction.

The model employed by the approach consists of three layers: service, device and environment layer (cf. Figure 6):

- *Service Layer* – the services a users has subscribed to or purchased. These may include security services, entertainment services or climate control services.
- *Device Layer* – the physical devices, such as lamps, motion sensors, or doors. Devices can be split into two categories: Firstly, *input devices* which are sensors monitoring the environment. These devices will trigger services. The second group are *output devices* which are controlled by services. Output devices may alter *environment variables*.
- *Environment Layer* – are the variables within the room or home environment. These variables are monitored or altered by services via devices. Examples include room temperature, movement, or noise. For a specific environment the set of variables is fixed, as are the links to particular devices. Only the access status of the variables will change depending on which devices are used by the triggered services.

4.1 Static Model

The model has a layer for the environment, which has not been used in previous feature interaction approaches. In this layer, attributes of the room or home, can be expressed. It then becomes possible to express how devices in the device layer relate to, affect, or are affected by the environmental variables. For instance, a security service will monitor movement in the

home, however, the fan and also the window link to the movement variable (cf. Figure 6). Thus, the key to the success of this model is creating correct relationships between layers.

Figure 6 shows the services, devices, and the environment variables used in the interaction examples discussed in Section 3.1. The relationships between devices and environment variables are shown by black arrows. These black arrows show the direction of information, for example, an air conditioner has an impact on the temperature variable and temperature will then affect the thermometer device. These relationships are static and determined prior to the deployment of services and operation of the approach. These relationships would be agreed by industry. For example, a heater will affect temperature regardless of who manufactures the device. In the service layer there are five services; in the device layer there are thirteen devices (incl. two sensors marked with a double line rectangle) and four variables in the environment layer.

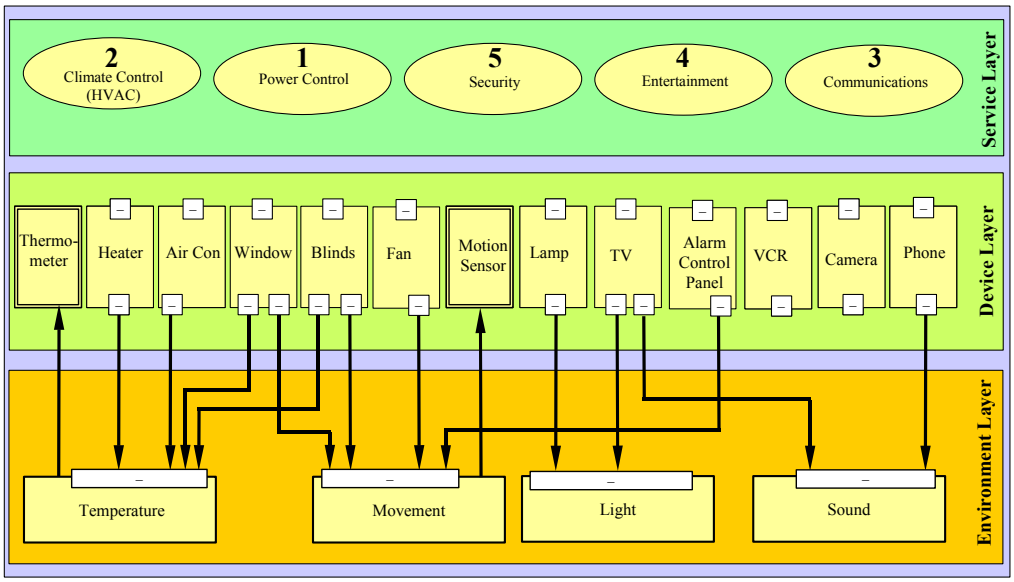


Figure 6: The Static Part of the Environmental Model incl. Example Services.

Each service is assigned a priority, which is shown as a number within each service oval. Priorities are assigned by numbering all services in the system. Each number can only be assigned to a single service. The service with the highest number is the most important and the one with lowest number is the least important one to the running of the house. This ranking of the services is expected to be carried out by the service provider on behalf of the user. If the user deploys a service, it can be expected that the user is capable of ranking the services themselves.

Other than sensors, each device has two sets of attributes: the access attribute of the device, and an attribute specifying the access to each environmental variable accessed by the device. It is these two types of attributes which are used to apply the operating system concept of resource locking to this model. All access attributes to devices and environmental variables are specified by the services during run-time. Hence the boxes showing the attributes in Figure 6 are left blank. The environmental variables also hold an access attribute during run-time. This is the access attribute used by the device accessing the variable.

4.2 The Approach in Action

During run-time the attributes are set by the service. The access attributes to output devices can be set to two possible values, *NS* or *S*. *Not Shared* is represented by *NS*: a device can only be

used by one service at any one time. *S* means *Shared*: a device may be used by multiple services concurrently. Services may change the access attributes during run-time. This may reflect a service using a device for a different purpose. Sensors are slightly different, in that access to them is always shared, i.e. a number of services may be notified of a certain event concurrently.

The attributes specifying access from a device to environmental variables may be assigned one of four possible values. Besides *NS* (a variable can only be altered by one device at a time) the values *S*±, *S*+ and *S*- are defined. *S*± means that a device changes the environmental variable in an unknown way, e.g. either increases or decreases the temperature. *S*± is also used for accessing environmental variables which cannot be influenced in a directed way, for instance movement cannot be increased or decreased. S+ and S- allow devices with the same goal to work together, i.e. devices changing the environment in the same way. For example, if a heater is accessing the temperature variable, other devices increasing the temperature are also allowed, as the access attribute from the heater to temperature is set to *S*+. Similarly, if the air conditioning is accessing the temperature variable, other devices lowering the temperature are also permitted (*S*-). However, using heater and air conditioning at the same time is not allowed as the temperature is influenced in a conflicting manner. Consequently, shared accesses with the same attribute are allowed, however, accesses with different attributes are *not* allowed.

An example of this behavior is depicted in Figure 7. Successful accesses to devices and environmental variables are shown by solid (blue) arrows. Unsuccessful attempts are shown by dashed red lines. The security service controls the alarm control panel and the access attribute is set to *NS* as no other service should be able to change the state of that panel at the same time. The attribute of the link to the movement environmental variable is also set to *NS*, as no other service should cause movement. Otherwise the security service could be set off by another service. As a consequence, the environmental variable is locked, shown by the *NS* attribute of the movement variable. On the other hand, the climate control service accesses the blinds. As it requires exclusive access to the blinds to achieve its goal (lower them), the attribute is set to *NS*. However, the attribute of the link between the blinds and the movement environment variable is set to *S*±. That means that the climate control service is happy that other devices also cause movement. However, since the security service accessed the movement variable first and locked it, the request by the climate control service via the blinds is denied. This interaction is example four listed in Section 3.1.

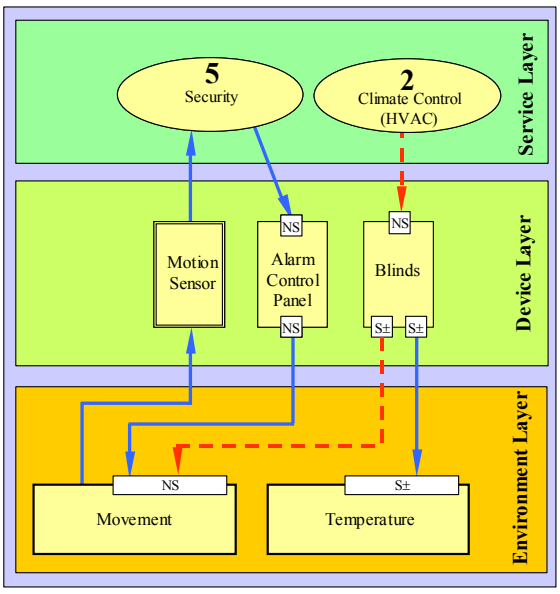


Figure 7: An Interaction between Security and Climate Control Services.

It is important to note that the access to a device by a service only succeeds if the device can access all required environmental variables. For this reason, the climate control service cannot access the blinds because the device only gets access to the temperature variable, but not to the movement variable.

Since sensors are expected to deal with only one particular aspect of the environment, e.g. temperature or movement, there is no access attribute from an environmental variable to a sensor. In other words, a sensor will only link to a single environmental variable. Figure 7 shows an example with movement and a movement sensor.

However, the interaction may also occur the other way around, i.e. the climate control service accesses the blinds before the security service. This scenario is shown in Figure 8(a). As the climate control service is first, the access to the movement variable by the alarm control panel is denied. In other words, the climate control service takes precedence over the security service. Arguably, this is not appropriate. The security service should take priority – what use is a perfectly acclimatized home which has been burgled?

To resolve this kind of conflict, priorities have been introduced. Services with a higher number have a higher priority which forces those with a lower priority to give up their resources. Figure 8(b) shows an example of this mechanism. In this case study, the security service has been assigned priority 5, which is the highest possible. Thus, the climate control service needs to give up control of the movement variable and consequently the blinds do not lower (dashed yellow lines). Afterwards, the security service can access the movement variable through the alarm control panel (dashed blue line). Consequently, the security service has been given priority over the climate control service.

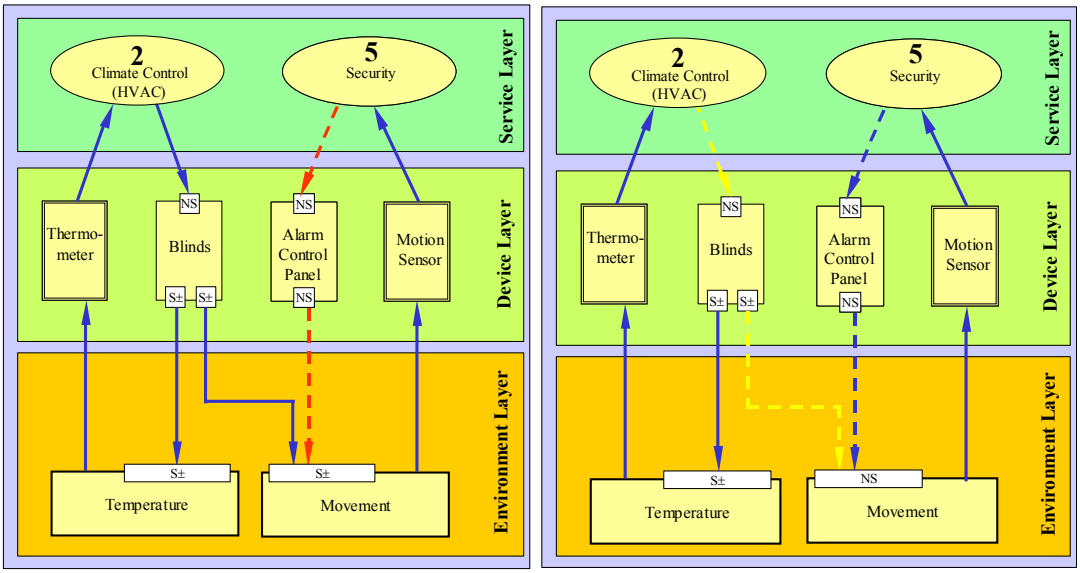


Figure 8: Interaction between Security and Climate Services revisited (a) and resolved (b).

The second interaction listed in Section 3.1 is between the security and entertainment services. The entertainment service is recording a TV program on the VCR. However, the security service is triggered and wants to record pictures from the security camera on the VCR. Similar to the example given in Figure 8, the VCR device is already locked (*NS*) by the entertainment service (cf. Figure 9(a)), however, due to the higher priority of the security service, the entertainment

service has to give up the VCR which can then be reassigned to the security service. Thus the picture of the burglar can be recorded on tape. This is shown in Figure 9(b). For the opposite case, where the security service acquires the VCR first (*NS*), the entertainment service will not be able to access the VCR, due to its lower priority. Thus this scenario is resolved successfully.

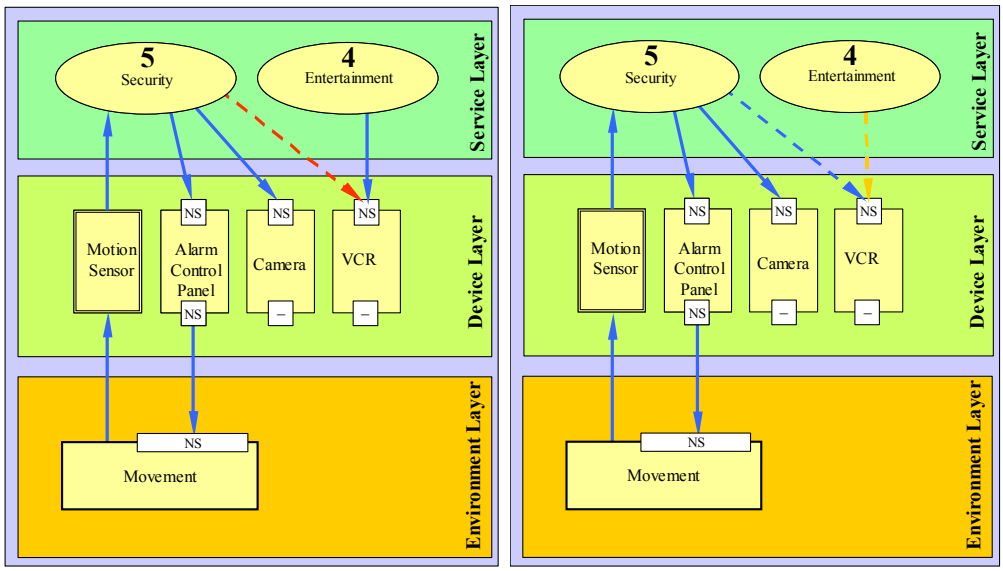


Figure 9: Interaction between Security and Entertainment Services (a) and resolved (b).

The first interaction listed in Section 3.1. between the security service and the power control service can also be handled by the approach. The security service turns appliances on to give the impression that someone is at home, whereas the power control service turns appliances off to save energy. The conflict between these services is depicted in Figure 10. Assuming the security service is running first, it accesses the TV and lamp (*NS*) therefore no other service can access them. Consequently, when the power control service tries to gain access to turn the devices off, its request is denied (cf. Figure 10). If, however, the power control service is accessing the devices first, the power control service will need to give up control of the appliances due to the higher priority of the security service.

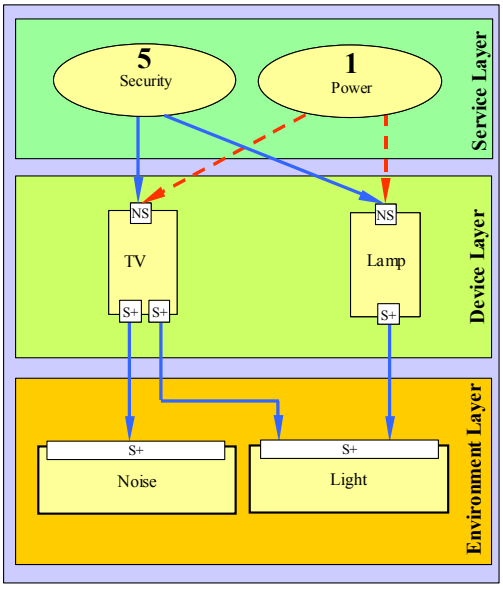
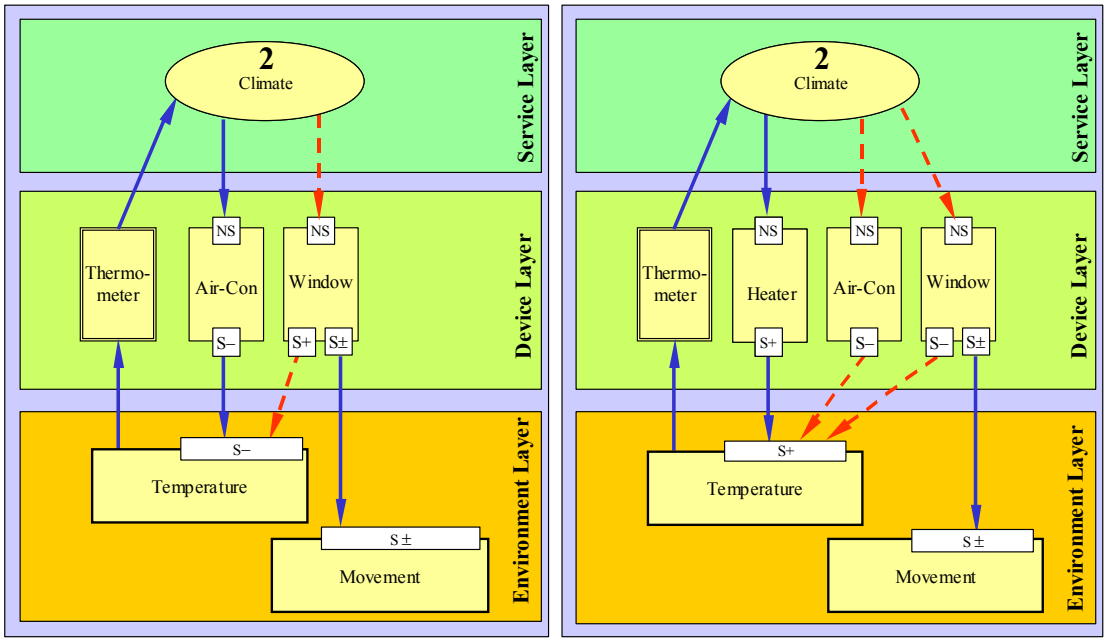


Figure 10: Interaction between Security Service and Power Control Service.

As discussed earlier in the paper, intra-service interactions should be picked up during testing of the service prior to deployment. Nevertheless, the approach is able to pick up this type of interaction as well. This is true for both intra-service interactions listed in Section 3.1, interactions three and five.



(a) (b)

Figure 11: Two intra-service interactions in the Climate Control Service.

The third example from Section 3.1, where the air conditioning is switched on at the same time as the window is opened is shown in Figure 11(a). Here it is assumed that as the air conditioning is switched on, it is warmer outside than inside the house. The air conditioning accesses the environmental variable temperature with S-, and the window with S+. Hence the concurrent access is not allowed. The same holds for the second intra-service interaction in the climate control service, where heater and air conditioning operate in an infinite loop. Again, this interaction is detected as only one device, either the heater or the air conditioning can get access to the temperature variable as they have conflicting access attributes to the temperature variable. This scenario is depicted in Figure 11(b). Also note that the window accesses the temperature variable with the attribute S-. This is because if the heater is used it can be assumed that it is colder outside than inside.

As can be seen, all interaction examples listed in Section 3.1 can be resolved with the approach presented here. Importantly, all scenarios above are resolved in a way that security and comfort functions are not compromised. This is an essential condition for a deployment of the system. However, not all interworking of services is bad. For example while the entertainment service controls the TV, the communications system is able to display a message on the TV that a phone call has arrived. This is because both services are happy that the TV is accessed by multiple services concurrently (cf. Figure 12). Clearly, shared access to devices offers increased flexibility and hence if the goal of a service is not compromised by sharing access to devices the attribute should be set to *S*.

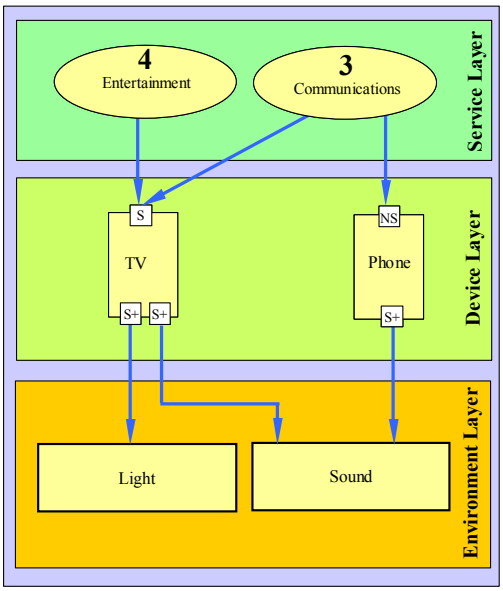


Figure 12: Successful interworking between Entertainment and Communications Services.

5. Conclusions

The feature interaction problem is threatening the success of networked appliances. If networked appliances are to be a success, the feature interaction problem needs to be solved. Lay users will buy home appliances off-the-self and subscribe to services. 'Surprises' as discussed in this paper will confuse them at best! Key is that devices are automatically configured and fully functional right out of the box.

It is important to point out that once an interaction has been identified it is often straightforward to fix it. The difficult task is to find the interactions in the first place – before the user becomes aware of a potential problem. This paper attempts to find a solution to this general problem, identifying specific interaction scenarios is not aim of this paper.

This paper highlights the problem and presents a number of illustrative example interactions in the home domain. The examples have been used to illustrate different classes of interactions presented in the taxonomy. The taxonomy was helpful to understand the problem and it is hoped that it will be useful in measuring the effectiveness of approaches. With the approach presented in this paper, Shared Trigger Interactions and Sequential Action Interactions are detected at the environmental layer. Multiple Action Interactions are detected at the device layer. This shows the importance of the environmental layer. Two types of interactions could not be detected without it.

Example interactions have been used to show the effectiveness of the approach presented in resolving conflicts between services in the home. The approach is novel in two ways; firstly it is the first attempt at tackling interactions between services controlling networked appliances in the home, and secondly because it employs a model of the environment. This is used to show the effects of services and devices on aspects of the environment. Interactions are detected as conflicts when accessing a device or an environmental variable. Hence the approach is device-centric rather than service-centric as commonly used in traditional feature interaction approaches. Using the approach to detect and avoid interactions has proven to be extremely powerful. Currently, the approach is implemented as a service on an OSGi platform.

References:

[1] E. J. Cameron, N. Griffeth, Y.-J. Lin, M. E. Nilson, W. Shnure and H. Velthuijsen, Towards a Feature Interaction Benchmark for IN and Beyond, IEEE Communications Magazine, Volume 31, Number 3, March 1993, pp. 64-69.

[2] M. Calder, M. Kolberg, E. Magill, S. Reiff-Marganiec: Feature Interactions: A Critical Review and Considered Forecast, Computer Networks Journal, Elsevier Science, Amsterdam, The Netherlands, Volume 41, Number 1, pp 115-141, 2003.

[3] D. O. Keck and P. J. Kuehn, The Feature and Service Interaction Problem in Telecommunications Systems: A Survey, IEEE Transactions on Software Engineering, Volume 10, Number 10, Oct. 1998, pp. 779-796.

[4] HAVi, http://www.havi.org (viewed: 8/18/2003)

[5] MHP, http://www.mhp.org (viewed: 8/18/2003)

[6] Jini, http://www.jini.org (viewed: 8/18/2003)

[7] UPnP, http://www.upnp.org (viewed: 8/18/2003)

[8] X-10, http://www.x10.org (viewed: 8/18/2003)

[9] Open Services Gateway Initiative: OSGi Service Platform (Release 2), IOS Press, 2002

[10] M. Kolberg, E.H. Magill, D. Marples, and S. Tsang. Feature Interactions in Services for Internet Personal Appliances, Proceedings of IEEE International Conference on Communications 2002 (ICC 2002), New York City, USA, Volume: 4, pp. 2613-2618.

[11] D. Marples, Detection and Resolution of Feature Interactions in Telecommunications Systems during Runtime, PhD Thesis, University of Strathclyde, Glasgow, UK, 2000.

[12] M. Calder, M. Kolberg, E.H. Magill, D. Marples and S. Reiff-Marganiec, Hybrid Solutions to the Feature Interaction Problem, In D. Amyot and L. Logrippo (eds.), Feature Interaction in Telecommunications and Software Systems VII, IOS Press, Amsterdam, 2003 pp. 295-312.

[13] EHS, http://www.ehsa.com (viewed: 8/18/2003)

[14] CEBus, http://www.cebus.org (viewed: 8/18/2003)