Managing Policy Interactions in KNX-Based Smart Homes

Mohamed Shehata[1], Armin Eberlein[2], Abraham O. Fapojuwo[3]

[1] *Dept. of Electrical & Computer Engineering, Shoubra Faculty of Engineering, Benha University, 108 Shoubra Street, Cairo, Egypt*
[2] *Dept. of Computer Engineering, American University of Sharjah, PO Box 26666, Sharjah, UAE*
[3] *Dept. of Electrical & Computer Engineering, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada*

{msshehat; eberlein; fapojuwo}@ucalgary.ca

Abstract

Smart homes have enjoyed increasing popularity in recent years. In order for them to further expand their market share, users need to be able to fully control devices. Policies are one way for users to achieve such flexible control of devices. However, user policies often tend to interact in unwanted ways leading to unexpected behavior of devices. This paper describes the design of a run-time policy interaction management module (PIMM) that serves as manager for detecting and resolving interactions between user policies in KNX-based smart homes. This module extends the traditional KNX networking system with the ability to manage policy interactions. The module operates in the run-time S-mode of the KNX network and works as part of the Engineering Tool Software (ETS) used to configure and control the operation of the KNX network in smart homes. The proposed module serves as the first of its kind that can be implemented inside the KNX networking system to detect and resolve unwanted policies interactions.

1. Introduction

A policy is user-defined information that can be used to modify the behavior of a system [1]. In recent years, interest in policies has increased as they provide greater flexibility and customizability of systems. For example, a person might set a policy regarding incoming phone calls such that calls from his/her boss are forwarded to the cell phone, private calls from the family to the spouse's number, and the remaining calls to another predefined phone number. A good overview of the current research of policies in distributed systems can be found in [2-4].

A new application area of policies is the control of smart homes devices. However, due to the expected large number of policies within a smart home, their diversity and potential complexity, there is a high chance that policies will negatively interact.

So far, little work has been done to address the problem of interactions between policies in general and within a smart home environment specifically. For example, the work in [5] describes the use of policies in the telecommunications domain. It suggested the use of a feature interaction manager that uses policies to control the composition of telecommunications services and features to prevent their interaction. The work in [6] proposed a policy architecture for enhancing telephony features and even suggested the use of policies as the features of the future. In [7], Kolberg et al. addressed interactions between networked smart homes appliances. Metzger discusses in [8, 9] the problem of feature interactions in embedded control systems including control systems in buildings where the use of a systematic approach for the detection of interactions has been proposed.

However, most of the work done so far has not comprehensively addressed the problem of policy interactions especially in smart homes. For example, the approaches described in [5] and [6] have been limited towards the use of policies in the telecommunications domain. The work in [7] was based on a limited interaction taxonomy that addressed only four types of interactions. Finally, [8, 9] addressed the problem of interactions at the feature level using a limited set of features in buildings.

This paper presents a run-time policy interaction management module for detecting and resolving interactions among user policies in KNX-based smart homes. This module extends the traditional KNX networking system [10, 11] to include the management of policy interactions. This module operates during run-time of the S-mode of the KNX network and operates as part of the Engineering Tool Software (ETS) [12]. The new extended KNX system with policy interaction support serves as the first smart homes networking system capable of managing policy interactions in an interactive manner during run-time.

This paper is structured as follows: Section 2 explains the use of policies to control smart homes devices. In section 3 presents the proposed policy interaction manager module. Section 4 presents the conclusions.

2. Using Policies to Control Smart Homes

2.1 Overview

Consumers play a significant role in the success or failure of a new product. In the case of smart homes, users are demanding full control over all smart home devices. This means that users can customize behavior of devices according to their needs. This is particularly becomes essential for elderly or disabled people that require customized control over household devices with the help of the so-called *Assistive Technology* [13] that has been developed to help such people to lead a more independent life.

Recently, users demanded even greater flexibility to specify complex system behavior that would accommodate their specific needs. Policies are one way of solving this dilemma between flexibility in specification and complexity of behavior. The current state of the art in smart homes allows a user to control devices individually (e.g. turn the lights on or off) but it is difficult to specify an overall behavior of the system. For example, let's consider a situation in which a user gets up at night to go to the bathroom. A user can set the following policy to describe a complex behavior that satisfy his/her needs: "If a person in room A gets out of bed between 11pm and 6am then the lights in room A and the hallway must switch on, initially at 50% of illumination then ramping up to 100% over 1 minute and the bathroom lights and fan must also switch on. After leaving the bathroom, the bathroom lights automatically switch off. After the person gets back into bed, the hallway and bedroom lights are dimmed from 100% to 50% over 1 minute and then switch off". This type of specification of system behavior is referred to as a *user policy*. The attractiveness of policies stems from the fact that a user does not need to deal with individual devices (e.g., lights, fans, sensors…etc). A policy manager determines what needs to be done in order to achieve the desired behavior of devices. Also, policies allow different users to customize their preferences by setting their own policies providing greater flexibility.

The challenge comes when people with different preferences live in the same home. If each person defines their own individual policies, chances that policies interact in an unwanted manner are very high. Policy interactions can be defined as the situation

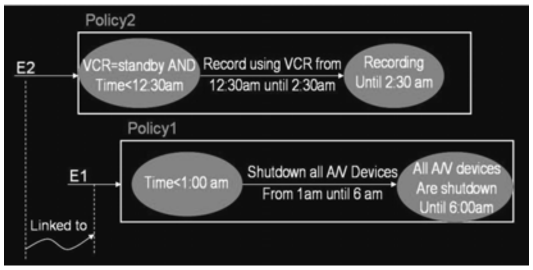

Policy2
E2
VCR=standby AND Time<12:30am
Record using VCR from 12:30am until 2:30am
Recording Until 2:30 am
Policy1
E1
Time<1:00 am
Shutdown all A/V Devices From 1am until 6 am
All A/V devices Are shutdown Until 6:00am
Linked to

Fig. 1. Screenshot of IRIS-TS identifying interaction between policy1 and policy2 [15]

when there is a negative relationship between two policies. A negative relationship can take several forms. For example, consider a policy regarding the Audio/Video control unit that was set by a user, e.g., a parent, which shuts down all audio and video devices between 1 am and 6 am daily. Now consider another Audio/Video-related policy that was set by another user, e.g., a son, which turns on the TV and the VCR (in recording mode) between 12:30 am and 2:30 am in order to record a specific TV show. Obviously, there is an interaction between the two policies.

To further demonstrate this negative relationship, the next three subsections discuss the above example from three different views. Subsection 2.2 shows how the above interaction can be detected by human experts. Subsection 2.3 discusses the occurrence and detection of the above interaction using a semi-formal interaction detection method called IRIS [14]. Finally, subsection 2.4 discuss the use of software simulation.

2.2 Detecting Policy Interactions using Experts

Experts are people who use their knowledge and expertise of the domain to detect interactions that might occur in the system. Considering the above example, an expert would be able to point out that these two policies interact based on his knowledge about the policies: At 12:30 am the Audio/Video control unit sends messages to the TV to switch on and the VCR to start recording as defined in the son's policy. The TV and VCR will work until the system clock is 1:00 am at which time the Audio/Video control unit will send shutdown messages to all audio and video devices in the house including the TV and the VCR according to the parent's policy. This means that the TV and VCR will shutdown even though the recording should have continued until 2:30am according to the son's policy. This means that the parent's policy overrides the son's policy before its completion and therefore the son's policy was not executed correctly.

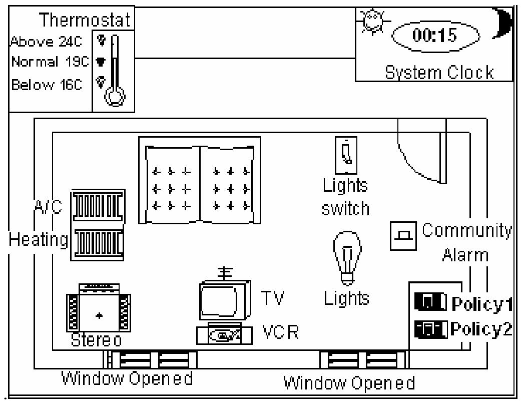


Fig. 2. Living room in a smart home

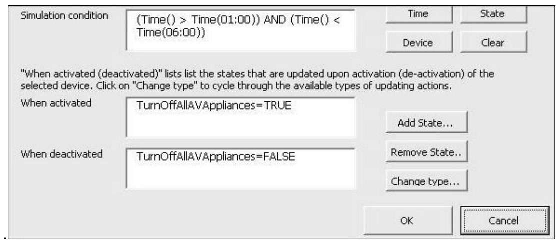
2.3 Detecting policy interactions using IRIS

IRIS, *I*dentifying *R*equirements *I*nteractions using *S*emi-formal methods, is an approach that uses semi-formal methods such as tables and graphs to detect interactions between requirements [14]. This approach has been used to successfully detect interactions between smart homes appliances. IRIS uses different tables and graphs to detect possible interactions offline. In [15], Shehata *et al.* discuss how IRIS was applied to detect interactions in smart homes. Figure 1 shows the use of IRIS-TS (IRIS-Tool Support) on the above smart homes policies example to detect interactions. IRIS was able to point out that trigger event E2 is linked to trigger event E1, i.e., the occurrence of E2 is followed after some time by the occurrence of E1 (in this case the two events are called linked events). When applying IRIS, the interaction between parent's policy and son's policy has been detected successfully [15].

2.4 Detecting interactions using simulation

The simulation used in this research is a virtual execution of the behavior of smart homes using a smart home virtual environment created with Microsoft Visio. The above example was carried out in a smart home living room with the configuration shown in Figure 2. The system clock is an ActiveX control that increments the time. The devices are networked and behave like real physical devices. Policies 1 and 2 were defined as shown in Figures 3a and 3b. Policy 1 assigns a "True" value to a system variable called TurnOffAllAVAppliances between 1 am and 6 am daily. This variable affects all A/V devices. In order for any A/V device to work, this variable needs to have the value "False". Figure 3d is a system trace that shows the sequence of events: At 12:30am policy 2 sends a true value to the VCR and TV in the living room, i.e. both are activated. However, at 1:00 am, policy 1 shuts down the VCR and TV before the recording is complete. Even though policy 2 was still active until 2:30 am, the TV and VCR are shutdown.

There are numerous interaction situations in smart homes that need to be detected and managed. For instance, in the example mentioned above, when the son tries to set policy 2, the system should be able to detect that there is an interaction with policy 1 and ask the son to have the parents enter a special Administration PIN that would make an exception for the living room VCR and TV to work in the restricted time period from 1 am until 6 am. If this PIN is not entered, then the new policy is deleted.

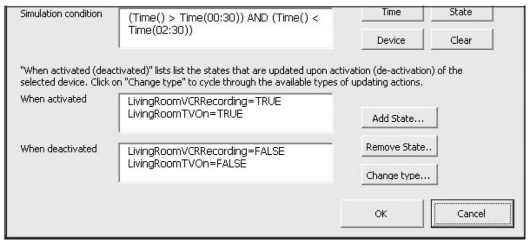

Simulation condition: (Time() > Time(01:00)) AND (Time() < Time(06:00))

"When activated (deactivated)" lists list the states that are updated upon activation (de-activation) of the selected device. Click on "Change type" to cycle through the available types of updating actions.

When activated: TurnOffAllAVAppliances=TRUE

When deactivated: TurnOffAllAVAppliances=FALSE

(a) Specification of Policy 1

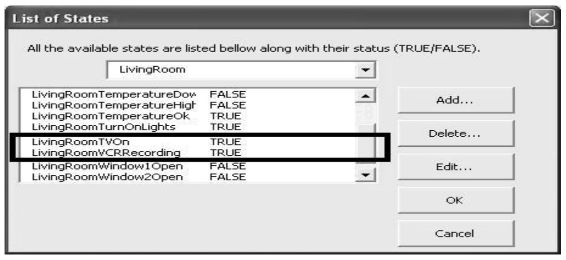

Simulation condition: (Time() > Time(00:30)) AND (Time() < Time(02:30))

"When activated (deactivated)" lists list the states that are updated upon activation (de-activation) of the selected device. Click on "Change type" to cycle through the available types of updating actions.

When activated: LivingRoomVCRRecording=TRUE
LivingRoomTVOn=TRUE

When deactivated: LivingRoomVCRRecording=FALSE
LivingRoomTVOn=FALSE

(b) Specification of Policy 2

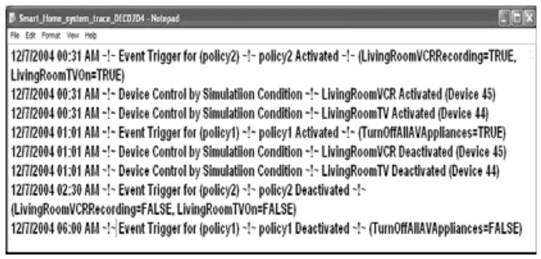

List of States

All the available states are listed bellow along with their status (TRUE/FALSE).

LivingRoom

LivingRoomTemperatureDow	FALSE
LivingRoomTemperatureHigh	FALSE
LivingRoomTemperatureOk	TRUE
LivingRoomTurnOnLights	TRUE
LivingRoomTVOn	TRUE
LivingRoomVCRRecording	TRUE
LivingRoomWindow1Open	FALSE
LivingRoomWindow2Open	FALSE

(c) Living room states taken at 00:45


Smart_Home_system_trace_DEC0704 - Notepad

```
12/7/2004 00:31 AM -!- Event Trigger for (policy2) -!- policy2 Activated -!- (LivingRoomVCRRecording=TRUE,
LivingRoomTVOn=TRUE)
12/7/2004 00:31 AM -!- Device Control by Simulatiion Condition -!- LivingRoomVCR Activated (Device 45)
12/7/2004 00:31 AM -!- Device Control by Simulatiion Condition -!- LivingRoomTV Activated (Device 44)
12/7/2004 01:01 AM -!- Event Trigger for (policy1) -!- policy1 Activated -!- (TurnOffAllAVAppliances=TRUE)
12/7/2004 01:01 AM -!- Device Control by Simulatiion Condition -!- LivingRoomVCR Deactivated (Device 45)
12/7/2004 01:01 AM -!- Device Control by Simulatiion Condition -!- LivingRoomTV Deactivated (Device 44)
12/7/2004 02:30 AM -!- Event Trigger for (policy2) -!- policy2 Deactivated -!-
(LivingRoomVCRRecording=FALSE, LivingRoomTVOn=FALSE)
12/7/2004 06:00 AM -!- Event Trigger for (policy1) -!- policy1 Deactivated -!- (TurnOffAllAVAppliances=FALSE)
```

(d) System trace of living room from midnight until sunrise

Fig. 3. Simulation results of the smart home living room with policy1 and policy2

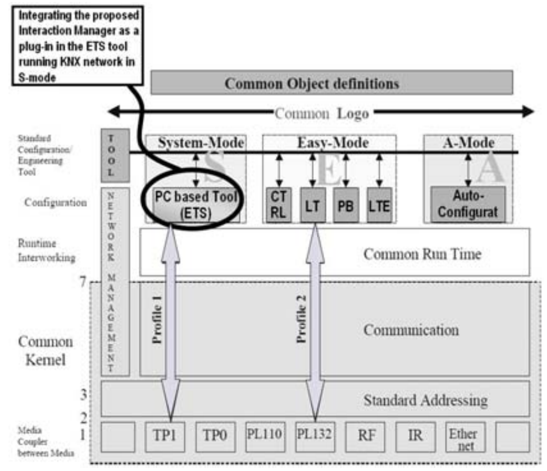


Fig. 4. KNX network model

3. Policy Interaction Manager Module

In this section we extend the traditional KNX networking system to address the problem of policy interactions. We propose the use of a run-time *P*olicy *I*nteraction *M*anager *M*odule (PIMM) that operates in the run-time S-mode of the KNX network and works as part of the Engineering Tool Software (ETS) used to control the configuration and operation of the KNX network in smart homes.

In the following, we use a top-down approach to demonstrate how the proposed interaction manager module is implemented within the KNX network system. Figure 4 shows a model of the KNX system describing its internal structure including the three modes of configuration and management. The proposed policy interaction manager module is implemented in the S-mode as part of the ETS suite. The S-mode was chosen because:

- The configuration and management of devices working in the S-Mode are shifted from manual configuration and management to a PC-based tool set called Engineering Tool Software ETS.
- S-Mode supports free programming.
- S-Mode ETS uses a database for storing information about all devices (e.g., home appliances, switches, sensors, etc.) in the proprietary format VD3.
- The S-Mode, unlike E-mode and A-mode, can have 3rd party plug-ins.

As stated earlier, the ETS is used to design and configure intelligent homes and buildings that are based on the KNX system. ETS modules rely on a product database with detailed information about each product. This description allows ETS to show the product, its available application programs and their associated parameters, and the corresponding possibilities for Group Address binding in a graphical way to the user.

The basic architecture of the ETS consists of the following components:

- ETS User Interface: This user interface is used to configure the devices of the smart home.
- Plug-in Components: They allow third party plug-ins to be added to the ETS software suite.
- Hawk: This is a database-based download manager (e.g. S19 file format) that can be used to download software for new KNX compatible devices from the different vendors and suppliers.
- Falcon: This is a DCOM (Distributed Component Object Model) based 32-bit access library that allows Windows applications to access the KNX network. Falcon offers several APIs to support all different aspects of bus access and device management.
- Eagle: This is a component similar to Falcon but used to access the standard ETS database (ETS DB). The ETS DB contains information about different KNX devices such as their description, their application programs, and their functionalities.

To extend the ETS software suite to be able to manage policy interactions, the following components have to be added to the ETS software suite:

- Policy-enabled ETS user interface: This is an enhanced interface that allows the user to manage devices in the smart home using policies rather than the standard ETS user interface.
- Policy Interaction Manager Module (PIMM): This is the core component and is responsible for decomposing user policies into atomic policies. An atomic policy is a policy that consists of only one functionality of one device. Usually a user policy is complex and consists of more than one atomic policy. For example, a user policy can be "at 8:00 am switch off living room lights and open the curtains". This policy can be decomposed into two atomic policies which are "switch off living room lights" and "open curtains". These two atomic policies are triggered by the event: time = 8:00am. After that the PIMM checks that no atomic policy interacts with any other policy set by other users. If an interaction is detected, it has to be resolved based on the priorities associated with the atomic policies involved in the interaction.
- Pre-defined Smart Homes Interaction Data Base (PSHI DB): This is a database in which interactions

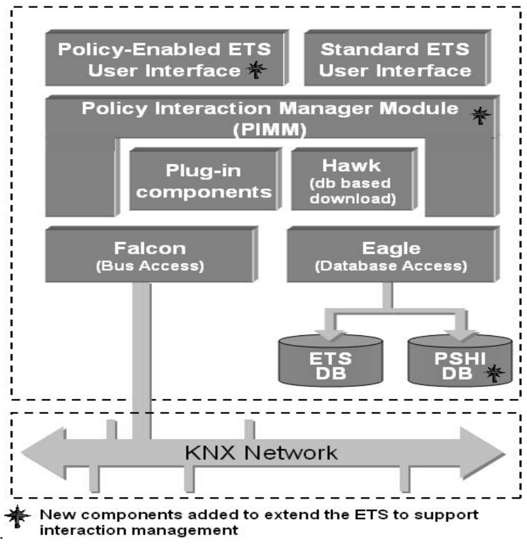

Policy-Enabled ETS User Interface
Standard ETS User Interface
Policy Interaction Manager Module (PIMM)
Plug-in components
Hawk (db based download)
Falcon (Bus Access)
Eagle (Database Access)
ETS DB
PSHI DB
KNX Network
New components added to extend the ETS to support interaction management

Fig. 5. Integrating the interaction manager in the ETS suite

that are known to occur between smart homes devices are stored. The PSHI DB is essential for the operation of the PIMM because the PIMM consults with this database to determine whether or not there exists an interaction between two atomic policies. The PSHI DB is accessible through the EAGLE database access component to provide consistency of the ETS software suite.

Figure 5 shows a model for the ETS software suite extended to manage policy interactions. The new components that were added are marked with an asterisk each. As seen from Figure 5, the user has a choice to work with the standard ETS interface or to work with the enhanced policy-enabled ETS user interface. In the latter case, the user is able to set complex policies regarding the behavior of the smart home. In either case, all policies have to go through the PIMM to check if there is a possible interaction between the new policy and an existing policy. For this reason the position of the two components Hawk and plug-ins were rearranged to be part of the PIMM to support interaction management.

To filter any unwanted interactions, the policy interaction manager works as a middle layer between the user interface through which the user sets his preferences and the execution of these preferences. Below the PIMM are the Falcon and Eagle components. The PIMM communicates with the ETS DB and the PSHI DB via the Eagle component when performing the interaction detection operation for new policies. Similarly, the PIMM communicates with the KNX network, via the Falcon component, to execute policies if no interactions were detected or after detected interactions were resolved. This means that all policies executed over the KNX-network devices are guaranteed to be interaction free.

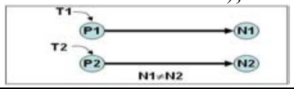
The task of the PIMM is to detect and resolve unwanted interactions between new policies and already existing policies set by other users. The following outlines the operation of the interaction manager module when a user sets a new policy:

- STEP 1: “Input user policy and decompose it into atomic policies A_i where i=1…n (n = number of atomic policies)”.
- STEP 2: “Set i=1”. This step is to initialize a counter i that increases from 1 to n to perform a loop over all the atomic policies A_1 to A_n (atomic policies of the new user policy A).
- STEP 3: “Check if any of the new atomic policies A_i has the same or a linked trigger event with an already existing atomic policy B”.
- STEP 4: “If step 3 does not find any same or linked trigger events with other policies, go to step 11, else go to step 5”. That is, if no identical or linked trigger events are found, no interactions will occur and we can accept the new atomic policy (step 11)
- STEP 5: “Compare the two atomic policies A_i and B for interactions based on the PSHI DB”. This step compares the two atomic policies for interactions using the predefined scenarios in the PSHI DB.
- STEP 6: “If the result of step 5 is null then go to step 11 else continue with step 7”. This step decides if there are no interactions between the two atomic policies. If this is the case, the new atomic policy can be accepted as an active policy (step 11). If an interaction is detected, continue with step 7.
- STEP 7: “If atomic policy A_i has a higher priority than atomic policy B, go to step 10, else continue with step 8”. This step compares the priorities of the two atomic policies.
- STEP 8: “Output a message to the user: atomic policy A_i interacts with atomic policy B. Enter an override PIN or cancel atomic policy A_i”.
- STEP 9: “If an override PIN is not entered, then delete atomic policy A_i and go to step 12. Else continue with step 10.
- STEP 10: “Delete atomic policy B”. This step is carried out if a PIN is entered to delete policy B.
- STEP 11: “Accept atomic policy A_i as an active policy”. This step means that atomic policy A_i has been accepted into the system and is ready to execute whenever it is triggered.

- STEP 12: “Increment i. If i is less than or equal to n, then go to step 3. Else go to step 1”.

In step 5, the PIMM analyzes the two atomic polices for interactions according to all interaction scenarios stored in the PSHI DB. For example, two atomic policies interact if they control the same device, have the same trigger event, have the same prestate, but have contradicting next states. The preceding definition of interaction is called non-determinism scenario of interaction [19]. Using this interaction definition, an interaction scenario can be defined as shown in Table1. Currently there are 30 different interaction scenarios stored in the PSHI DB to be used by the PIMM in smart homes [19].

Table1: Non-determinism interaction scenario

Scenario ID	SCR1
Interaction Scenario Body	(P1.Device)=(P2.Device) AND (P1.TriggerEvent=P2.TriggerEvent) AND (P1.PreState=P2.PreState) AND (P1.NextState ≠ P2.NextState)} 

5. Conclusion

This paper presented the problem of policy interactions that occurs in smart homes. This problem has been discussed from different viewpoints including experts, semi-formal methods, and software simulation which have highlighted the effects of interactions.

A policy interaction manager module (PIMM) was proposed to extend the traditional KNX networking system. This policy interaction manager module was introduced as part of the Engineering Tool Software suite (ETS) to work as a run-time interaction manager to detect and resolve any unwanted interactions between policies. The module detects interactions between policies based on 30 interaction scenarios that define when two policies interact. These interaction scenarios are stored in the pre-defined smart home interactions database (PSHI DB).

REFERENCES

[1] E. C. Lupu and M. Sloman, "Conflicts in policy-based distributed systems management," IEEE Transactions on Software Engineering, vol. 25(6), pp. 852-869, 1999.

[2] J. B. Michael, J. Lobo, and N. Dulay, Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks: IEEE Computer Society, Los Alamitos, California, USA, June 2002.

[3] H. L. Lutfiyya, J. Moffett, and F. Garcia, Proceedings of 4th IEEE International Workshop on Policies for Distributed Systems and Networks: IEEE Computer Society, Italy, 2003.

[4] D. Verma, M. Devarakonda, and a. M. K. E. Lupu, Proceedings of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks: IEEE Computer Society, New York, USA, 2004.

[5] A. D. Marco and F. Khendek, "eSERL: Feature interaction in Parlay/OSA using composition constraints and configuration rules," in Feature Interactions in Telecom. and Software Systems VII, D. Amyot and L. Logrippo, Eds. Amsterdam: IOS Press, June 2003.

[6] S. Reiff-Marganiec and K. J. Turner, "A policy architecture for enhancing and controlling features," in Feature Interactions in Telecommunications and Software Systems VII, D. Amyot and L. Logrippo, Eds. Amsterdam: IOS Press, 2003, pp. 239-246.

[7] M. Kolberg, E. H. Magill, and M. Wilson, "Compatibility Issues between Services Supporting Networked Appliances," IEEE Communications Magazine, vol. 41, pp. 136 - 147, 2003.

[8] A. Metzger and C. Webel, "Feature Interaction Detection in Building Control Systems by Means of a Formal Product Model," 2003.

[9] A. Metzger, "Feature interactions in embedded control systems," Computer Networks, vol. 45, pp. 625-44, 2004.

[10]S. D. Bruyne, "Finding your way around the KNX Specifications", presented at KNX Technology Tutorial Workshop, Deggendorf, Germany, October 6th, 2004.

[11]KNX Technology, http://www.konnex-knx.com/, Last viewed on January 10th, 2005

[12]Engineering Tool Software - ETS, http://www.eiba.com/en/ets3/, Last viewed on January 10th, 2005

[13]G. Dewsbury, B. Taylor, and M. Edge, "DESIGNING DEPENDABLE ASSISTIVE TECHNOLOGY SYSTEMS FOR VULNERABLE PEOPLE," Health Informatics Journal, ISSN 1460-4582, vol. 8, Number 2, pp. 104-110, June 2002.

[14]M. Shehata, A. Eberlein, and A. Fapojuwo, "IRIS: a semi-formal approach for detecting requirements interactions," presented at Proceedings. 11th IEEE International ECBS, 24-27 May 2004, Brno, Czech Republic, 2004.

[15]M. Shehata, A. Eberlein, and A. O. Fapojuwo, "Feature Interactions between Networked Smart Home Appliances," presented at QSSE, 4th ASERC Workshop on Quantitative and Soft Computing Based Software Engineering, Banff, Alberta, Canada, Feb. 16-17 2004.

[16]BatiBUS technology, http://www.batibus.com/, Last viewed on January 10th, 2005

[17]EIB Technology, http://www.eiba.com/, Last viewed on January 10th, 2005

[18]EHS Technology, http://www.ehsa.com/, Last viewed on January 10th, 2005

[19]M. Shehata, A. Eberlein, and A. O. Fapojuwo, "Using Semi-Formal Methods for Detecting Interactions among Smart Homes Policies," Submitted for Journal of Systems and Software, August, 2004.