Electronic Communications of the EASST
Volume 11 (2008)

Proceedings of the
First International DisCoTec Workshop on
Context-aware Adaptation Mechanisms for
Pervasive and Ubiquitous Services
(CAMPUS 2008)

Feature Interaction in Pervasive Computing Systems

Yu Liu and René Meier

6 Pages

Guest Editors: Romain Rouvoy, Mauro Caporuscio, Michael Wagner
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Feature Interaction in Pervasive Computing Systems

Yu Liu and René Meier

Lero @ TCD, Department of Computer Science,
Trinity College Dublin, Ireland
{yuliu, rene.meier}@cs.tcd.ie

Abstract: Feature interaction describes a situation where the combination of two or more services that individually perform correctly results in unexpected and possibly adverse behaviour. Such feature interaction issues have first been identified in telecommunication systems and are now beginning to be considered in other distributed software systems. We expect significant feature interaction research in pervasive computing where very many applications collaborate and adapt to changes to their environment or to user needs in order to provide tailored services to users. This paper presents a classification of feature interaction issues in pervasive computing systems. The classification captures, with a focus on automotive systems and systems for smart homes, feature interaction issues related to types of interaction, channels of interaction, and user needs. The classification aims to aid the understanding of feature interaction in pervasive computing systems, and to serve as a guideline for designers of pervasive applications.

Keywords: Feature Interaction, Pervasive Computing, Adaptation

1 Introduction

The complexity of pervasive applications has grown dramatically, with the involvement of a diverse set of devices, the dynamic nature of the environment that these applications monitor and control, and the changing user needs. Examples of pervasive applications can be found in areas such as automotive systems and systems for smart homes. Today's luxury cars are equipped with a multitude of software intended to assist car users to enhance safety, to strengthen security, and to improve comfort. A car essentially becomes a smart object, reacting dynamically to changing driving conditions as well as to the condition of the vehicle and the driver. Smart homes include services for controlling lighting, heating, and ventilation as well as for delivering information and entertainment. These services become networked smart objects to adjust to the needs of residents.

Generally, pervasive computing systems exist in a dynamic environment that has a range of parameters that must be catered for. Pervasive applications need to be context aware so that information about the context of a device can be captured and utilized. For example, in smart homes, relevant pattern of occupancy and alternative means of heating, such as sunlight, are important parameters a heating control application should capture and adapt to. The changes in the context ought to be dynamically accommodated and as such pervasive applications must be adaptable.

Collaboration and adaptation of pervasive applications is hindered by feature interaction issues, which were first identified in telecommunication systems [1] and are now considered relevant to other distributed software systems. Feature interaction is a situation where a number of services work properly in isolation, but exhibit undesired behaviour when combined. A wealth of literature [7] on feature interaction can be found in the telecommunication domain, but relatively little work has focused on addressing the distinct requirements of pervasive computing systems. A notable distinction, between telecommunication systems and pervasive computing systems, is that they have very different notions of context. Telecommunication applications treat the signalling channel as their context whereby the set of status signals of a call is transmitted and shared. Pervasive applications, on the other hand, consider the context in terms of the physical environment and of the needs of individual users. Consequently, pervasive computing systems have to deal with a much more complex context, and therefore face greater possibilities of feature interaction.

This paper discusses feature interaction in pervasive computing systems presented in the form of a classification. The classification captures types of interaction, channels of interaction, and user needs. Types of interaction describe how pervasive applications interact with their context and with each other. Channels of interaction are concerned with identifying the pathway whereby pervasive applications may interfere with each other. User needs are important inputs to pervasive computing systems and aim to describe the overall system behaviour. However, individual users might have conflicting goals that may not be satisfied at the same time or users have quality of service constraints that can not be met at runtime, thus leading to feature interaction. Calder [2] proposes a classification of telecommunication features, and his classification serves as basis for our types of interaction. In the pervasive context, however, both physical environment and user needs play a crucial part. Our classification extends his classification to deal with physical environment and user needs.

The remainder of the paper is organized as follows: Section 2 introduces the three major categories of our classification and uses them to discuss feature interaction in pervasive computing systems; Section 3 describes related work; Section 4 concludes this paper and outlines future work.

2 The Classification

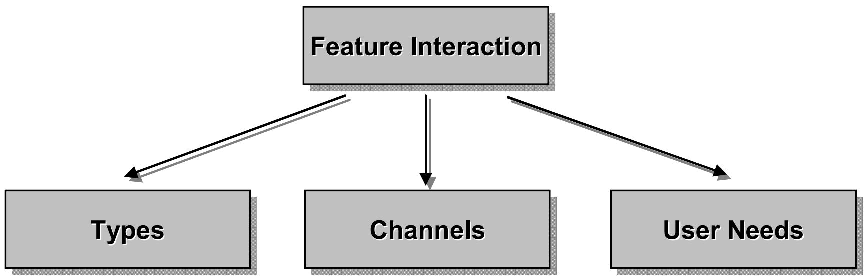

Figure1. Root of the Classification

As shown in Figure 1, Feature Interaction in pervasive computing systems can be described in three categories, namely, types of interaction, channels of interaction, and user needs.

2.1 Types of Interaction

Types of feature interaction describe how features, or services, are triggered. Features can trigger each other, or respond to the same external trigger. In telecommunication systems, external triggers reside in a two-way signalling channel and features usually detect their own triggering conditions. However, in pervasive computing systems, external triggers of a feature are defined by a possibly large set of context properties, which can be changed by the environment, which in turn can be influenced by the feature or indeed by other features.

Shared Trigger. Shared Trigger interaction takes place when more than one feature is activated by the same event trigger, and their responses conflict with each other. For example, consider the application that opens windows and an application that controls air conditioning [4] in a modern house. Both applications are triggered once temperature reaches a certain threshold value chosen by the occupant of the house. Opening the windows will compromise the effectiveness of air conditioning. This type of interaction can affect the environment and might result in a fluctuating room temperature.

Sequential Trigger. Sequential Trigger interaction occurs when the responses of one feature cause another feature to be triggered [2]. Sequential trigger takes two forms: 1) One feature sends a notification directly to another feature. 2) One feature affects environment variables, such as temperature and humidity, through actuators, for example, air conditioner and heater, and these changes to the environment variables then lead to other features being triggered.

Looping Trigger. There are situations where individual features may run correctly but multiple features as a whole are stuck in a loop [2]. This is due to cyclic generation of sequential events, whereby the features involved get triggered repeatedly. Looping Trigger is considered to be a special case of Sequential Trigger. For example, in order to keep humidity of a room at a constant level, the Humidifier application is activated once humidity drops below the constant level. Meanwhile, the Ventilation application is triggered by a high humidity level that is caused by humidifier. Hence, this may cause the Humidifier application and Ventilation application to be switched on and off repeatedly.

Missed Trigger. Missed Trigger interaction refers to a situation where the presence of one feature in the system prevents the second feature from operating. Some features are designed to use the same device; however, the undesirable situation could be that one feature has full control over the device or disable it, thus preventing other features from correctly functioning.

2.2 Channels of Interaction

Channels of feature interaction are concerned with the pathway whereby pervasive applications may interfere with each other. Pervasive computing systems can be modelled through three layers, namely, application layer, device layer and environment layer [8]. Feature interaction might occur at each layer for different reasons.

Application Layer. Experiences from telecommunication domain [1] have shown that distributed support for applications, such as information sharing or transactions, might be problematic and as such may give rise to unanticipated interaction between features. Two issues arising from distributed application support also find their way into pervasive computing systems. The first issue is concerned with the assumption of data availability of pervasive applications. Based on its context, one application assumes that certain data is available while the applications that hold the data keep it private. The second issue is transaction support of pervasive applications. Apart from the issues in distributed support for applications, dynamic adaptation of pervasive applications can also lead to unanticipated interactions, as some of the applications might collaborate for the first time and lack knowledge of each other's interface and behaviour.

Device Layer. The device layer comprises sensors, actuators and control devices. In this layer, feature interaction issues boil down to the conflicts in accessing control devices. Some control devices may only be used by one application at the same time, while other devices can be used by multiple applications simultaneously. Lack of control over the access to the shared devices can result in a deadlock situation for the applications attempting to use such devices. Another issue regarding application-device interaction is the absence of the notion of a session. Feature interaction in telecommunication applications is dealt with in a specific period of time, namely, a session. However, there is no indication of the completion of a control request delivered to a device [4]. For the devices controlled by a pervasive computing system, some of them can carry out a control request almost instantaneous while other devices carry out a request over a certain length of time, for example, to request a VCR to record a TV program. Hence, the start and endpoint to look for feature interaction are not well defined.

Environment Layer. The environment layer is the source of an implicit coupling between different applications in a pervasive computing system. Environment variables, such as temperature and humidity, can be changed by control devices. One feature might be triggered by the changes in an environment variable that other features have influence upon. Kolberg [8] advises to explicitly specify the links between control devices and environment variables. In addition, two seemingly irrelevant environment variables can affect each other, thus creating an implicit relationship between trigger conditions of several otherwise independent features. Metzger [5] proposes to build an environment simulator to establish relationships between environmental variables.

2.3 User Needs

An integral part of the context that pervasive applications deal with is the needs of user. User needs are arguably the important inputs to a pervasive computing system. User needs also constrain the overall behaviour of the system. User needs can be divided into behaviour constraints and Quality of Service (QoS) constraints. Behaviour constraints are used to specify the composite behaviour of features. QoS constraints are generally concerned with timeliness, resource consumption and other QoS metrics associated with features.

Behaviour Constraints. Users of a pervasive computing system typically define specific behaviour constraints. These constraints range from simple tasks such as controlling devices individually, e.g., switch on the heater at 9pm, to situations that involve very complex user activities, such as controlling the level of illumination of a room according to user profiles. Behaviour constraints typically have been embedded in the implementation of applications. This is at the expense of customizability highly demanded by users. We believe that behaviour constraints are part of the context and should be separated from the implementation of the applications. Policy based solutions [3] are a promising approach and have already been applied in smart homes, whereby users can explicitly specify the behaviour constraints they can accept in terms of policies as opposed to embodying these constraints inside implementation of features. Some industrial strength platforms for smart homes, such as KNX

[9], make it easier to integrate devices from different vendors and suppliers. By employing this capability, users are writing their own policies using an elementary set of basic built-in features of a platform. Policy based solutions consider feature interaction as two policies having inadvertent embedded conflicts. In adopting policy based solutions, there are two advantages: 1) Users have greater flexibility to express system behaviour constraints in terms of policies. 2) Conflicting behaviour constraints can be detected through policy conflicts.

QoS Constraints. QoS constraints such as timeliness, resource consumptions of individual features are essential considerations for dynamic adaptation. For instance, time-bounded adaptation in automotive systems is a pressing issue [6]. Adaptation typically involves swapping in new features or swapping out existing features to suit the current context. The issue of feature interaction comes down to how to guarantee QoS constraints of existing features when new features are dynamically activated. Very little current work on feature interaction considers QoS constraints of a system. This must be addressed, especially in mission-critical pervasive computing systems, such as automotive systems.

3. Related Work

Feature interaction issues in telecommunication systems have been extensively studied. Cameron [1] identifies three dimensions for feature interaction: 1) The kind of features involved in the interaction (customer features, system features). 2) The number of users involved (single user, multiple users). 3) The number of network components involved in the interaction (single component, multiple components). Calder [2] proposes a newer classification of telecommunication features, and his classification contains four types of interaction using trigger condition as the criteria. Our classification extends Calder's classification to deal with a much more complex context which incorporates the physical environment and user needs.

Feature interaction issues have also been researched in other domains. Metzger [5] discussed feature interaction issues in embedded control systems introducing the notion of physical environment and proposes to build an environment model to reveal implicit relationships of environment variables. Kolberg [4] [8] addresses only four types of interaction in smart homes, and he proposes a simplified model of environment variables. Shehata [9] recognises a paradigm shift from feature interaction to policy interaction in smart homes, so that users can explicitly specify behaviour constraints and interaction detection can be carried out at both design time and runtime. However, we believe that in addition to behaviour constraints, QoS constraints must also be considered for (mission critical) pervasive computing systems.

4. Conclusion and Future Work

This paper proposed a classification of feature interaction issues in pervasive computing systems. This is achieved through extending existing work on feature interaction in telecommunication systems. The classification comprises three categories, namely, types of interaction, channels of interaction, and user needs. These categories have been discussed with an aim to foster understanding of feature interaction in pervasive computing in light of current research in the area. We intended to further diversify our classification with a special focus on distinguishing between desired and unwanted interactions, on challenges driven by the physical environment and on detecting and resolving conflicts at the device layer.

Acknowledgement

The work described in this paper is partly supported by Lero - the Irish Software Engineering Research Centre.

References

[1] E.J.Cameron, N.Griffeth, Y.J.Lin, M.E.Nilson, H.Velthuijsen and W.K.Schnure. A feature interaction benchmark for IN and beyond. In *Feature Interactions in Telecommunication Systems*, IOS Press, pp.1-23, May 1994.

[2] M.Calder, M.Kolberg, E.Magill, D.Marples, and S.Reiff-Marganiec. Hybrid Solutions to the Feature Interaction Problem. In *Proc. 7th. Feature Interactions in Telecommunications and Software Systems*, IOS Press, June 2003.

[3] P.Dini, A.Clemm, T.Gray, F.J.Lin, L.Logrippo, and S.Reiff-Marganiec. Policy-enabled mechanisms for feature interactions: reality, expectations, challenges. In *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Volume 45, Issue 5, August 2004.

[4] M.Kolberg, E.Magill, D.Marples, and S.Tsang. Feature Interactions in Services for Internet Personally Appliances. In *Proceedings of IEEE International Conference on Communications 2002, New York, USA, vol. 4, pp. 2613-2618.*

[5] A.Metzger. Feature interactions in embedded control systems. In *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Volume 45, Issue 5, August 2004.

[6] S.Fritsch, A.Senart, D.C.Schmidt, and S.Clarke. Time-bounded Adaptation for Automotive System Software. In *Proceedings of the 30th International Conference on Software Engineering*, 2008.

[7] M.Calder, M.Kolberg, E.H.Magill, S.Reiff-Marganiec. Feature Interaction: A critical review and considered forecast. In *Computer Networks*, Volume 41, Number 1, 15 January 2003, pp. 115-141(27).

[8] M.Kolberg, E.H.Magill, and M.Wilson. Compatibility issues between services supporting networked appliances. In *Communications Magazine, IEEE*. Volume 41, Issue 11, 2003.

[9] M.Shehata, A.Eberlein, and A.O.Fapojuwo. Managing Policy Interactions in KNX-Based Smart Homes. In *Proceedings of the 31st Annual International Computer Software and Application Conferences,* Vol 2, 2007.