Noname manuscript No.
(will be inserted by the editor)

Automating Unobtrusive Personalized Services in Ambient Media Environments

Estefanía Serral[1], Miriam Gil[2], Pedro Valderas[2], Vicente Pelechano[2]

Received: date / Accepted: date

Abstract In the age of ambient media, people are surrounded by lots of physical objects (media objects) for rendering the digital world in the natural environment. These media objects should interact with users in a way that is not disturbing for them. To address this issue, this work presents a design and automation strategy for augmenting the world around us with personalized ambient media services that behave in a considerate manner. That is, ambient services are capable of adjusting its obtrusiveness level (i.e., the extent to which each service intrudes the user's mind) by using the appropriate media objects for each user's situation.

Keywords Unobtrusive ambient media · Personalized services · Context awareness

1 Introduction

Ambient media are embedded throughout the natural environment of users in physical objects that surround them (e.g., in his mobile device, in his car, in the TV, in the lights, etc.) and stimulate their human senses. These physical objects are called *media objects*[16]. To promote a natural interaction between the user and the environment, media objects must behave invisibly and unobtrusively, and negotiate the volume and the medium of the interactions.

This work is focused on supporting unobtrusive personalized services using the appropriate media objects. To achieve more invisibility, these ambient

[1] Christian Doppler Laboratory "Software Engineering Integration for Flexible Automation Systems", Vienna University of Technology
Favoritenstrasse 9-11/188, 1040 Vienna, Austria
estefania.serral@tuwien.ac.at
[2] Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camí de Vera s/n, 46022 València, Spain
Email:{mgil, pvalderas, pele}@pros.upv.es

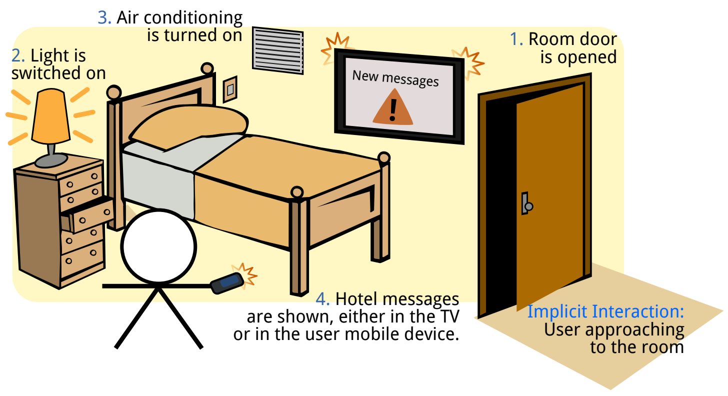

3. Air conditioning is turned on
2. Light is switched on
New messages
1. Room door is opened
4. Hotel messages are shown, either in the TV or in the user mobile device.
Implicit Interaction: User approaching to the room

Fig. 1 Example scenario of a personalized service adjusting the obtrusiveness level

services will be proactively executed in reaction to implicit interactions (i.e., those that occur without the explicit behest or awareness of the user) [13], based on the situational context such as location, surrounding environment or user's state [24]. For example, when a hotel guest approaches to his/her room (implicit interaction), a welcome hotel service will be automatically executed (see Figure 1): opening the door room to let the user come in, switching on the lights of the room, turning on the air conditioner to fit user preferences and communicating the new messages/news of the hotel to the user.

Our strategy to design unobtrusive personalized services is based on making ambient media more considerate [8] and less interruptive. For example, communicating the new messages using the ambient sound may be annoying either when the user is busy or with company. According to the Considerate Computing vision [8], user attention is a primary resource to be considered. Thus, ambient media services must behave in a considerate manner, demanding the proper level of user attention in each situation. Considering the service above introduced, each time it is performed, some of its actions could require a different degree of user attention. As a consequence, different media objects should be used to support each task at the appropriate obtrusiveness level (e.g., using either the TV or the mobile device to show the hotel messages).

The main contribution of this work is a design strategy and a software infrastructure for describing and automating unobtrusive personalized services in ambient media environments using high-level abstraction models. These models specified at design time are leveraged at runtime for supporting the execution of the described behaviour in an unobtrusive manner. In this way, the design effort made at design time is not only useful for producing the system, but also provides a richer semantic base for adaptation during execution. Also, user needs drive the design of the system providing users with personalized services with their appropriate degree of obtrusiveness and avoiding information overload.

The paper is organized as follows. Section 2 describes the high-level abstraction models for describing unobtrusive personalized services. Section 3 explains how these models are used at runtime for providing the functionality of these services. Section 4 describes the evaluation of the proposal by means of a case study. Section 5 describes the related work. Finally, Section 6 discusses this work and presents conclusions and further work.

2 Modelling Unobtrusive Personalized Services

The goal of the design strategy presented in this work is to manage the media objects of the environment to augment the world around users and provide them with unobtrusive personalized services. This interactivity with the physical objects should be adapted to the attentional resources and needs of each user in order to avoid overwhelming him/her. Thus, we propose personalizing services in ambient media environments by using the most appropriate media object for each situation according to users needs and preferences. For defining such services, designers need to perform the following steps (see Fig. 2):

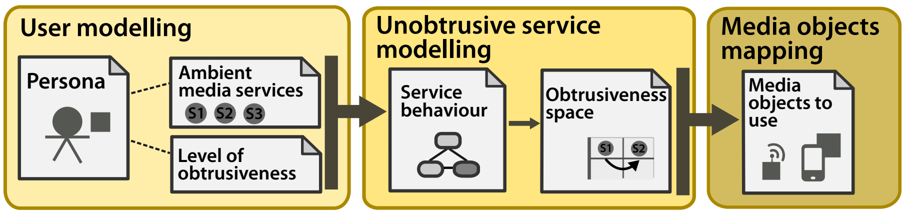


Fig. 2 Overview of the design strategy

1. ***User modelling***: Detect user needs and preferences to determine the needed ambient media services and the obtrusiveness level required for the interaction with them.
2. ***Unobtrusive service modelling***: Define the ambient media service behaviour and interaction style based on user's needs and obtrusiveness to make use of the appropriate media objects for each situation.
3. ***Mapping obtrusiveness to media objects***: Define the media objects (physical objects of the environment) most appropriate for each service in each situation.

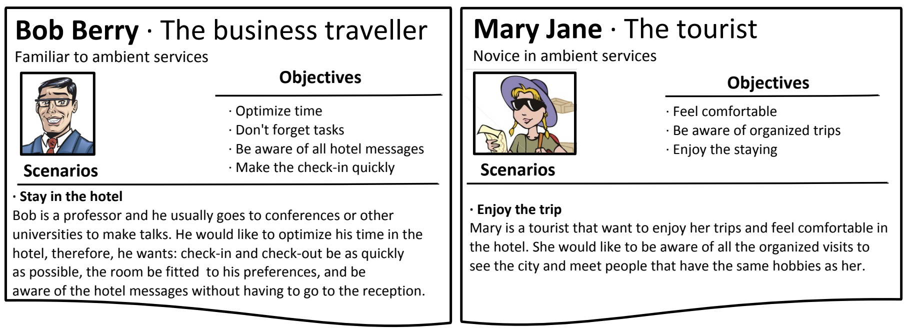
These steps and how they are supported by the proposed models are further described in the next subsections.

2.1 User Modelling

To detect user needs and preferences for achieving the personalization of ambient media services we make use of *personas* (also known as user profiles). A persona is a summary representation of the system's intended users, often described as real people [3]. Personas provide a framework for describing the target audience in a way useful to design systems. In this way, ambient services can be personalized according to these descriptions.

Personas are used to gather the relevant information of the audience to help drive design and detect common functionalities between users. Personas are usually used in the design of user-centered approaches. According to users, personas give a much more concrete picture of typical users providing features that directly address specific user needs [9]. Thus, it is interesting the use of them in this work where we have directly addressed specific user needs.

Personas describe target users of the system, giving a clear picture of how they are likely to use it, and what they will expect from it [3]. It has become a popular way for design teams to capture relevant information about customers that directly impact on the design process: user goals, scenarios, tasks, and the like. Scenarios are little stories describing how typical user tasks are carried out. They help to identify both the decisions that a user will have to make at each step in their experience and his/her preferred interaction styles.



Bob Berry · The business traveller
Familiar to ambient services

Objectives
· Optimize time
· Don't forget tasks
· Be aware of all hotel messages
· Make the check-in quickly

Scenarios
· Stay in the hotel
Bob is a professor and he usually goes to conferences or other universities to make talks. He would like to optimize his time in the hotel, therefore, he wants: check-in and check-out be as quickly as possible, the room be fitted to his preferences, and be aware of the hotel messages without having to go to the reception.

Mary Jane · The tourist
Novice in ambient services

Objectives
· Feel comfortable
· Be aware of organized trips
· Enjoy the staying

Scenarios
· Enjoy the trip
Mary is a tourist that want to enjoy her trips and feel comfortable in the hotel. She would like to be aware of all the organized visits to see the city and meet people that have the same hobbies as her.

Fig. 3 Excerpt of two personas

In this work, we follow the notation defined by [3] to determine the needs of each user and the functionality required in a specific domain. Figure 3 shows an example of two personas for a Smart Hotel system. These personas give a detailed picture of a *business traveller* and a *tourist*. Both have different objectives and have different scenarios according to their needs. For example, the *business traveller* is a busy person that wants optimize his time, be productive and be aware of the hotel messages in all moment. In contrast, the *tourist* wants to feel comfortable during the staying in the hotel and be aware of the organized trips. Thus, these different personas require different services and different interaction styles with the services according to their needs. By the definition of personas, analysts must detect the services needed for interacting with the system. For example, the *business traveller* would need an ambient service executed when he enters the room in charge of fitting the room to his preferences and remember him the missed hotel messages. On the contrary, the *tourist* could require a tourist guide service that informs her the organized visits and a waking up service that makes her staying more comfortable. In this way, personas will guide subsequent adaptations in information presentation, modality and media objects chosen.

2.2 Unobtrusive Service Modelling

Once the ambient media services and the functionalities are identified according to the user needs, analysts have to determine the behaviour of the services to give support to the functionalities. The way to describe these services' behaviour in an unobtrusive way is explained in the following subsections.

2.2.1 Describing the service behaviour.

In order to describe ambient service behaviour we use task models inspired by *Hierarchical Task Analysis*, which hierarchically refines a high-level task into more specific ones by building a task tree.

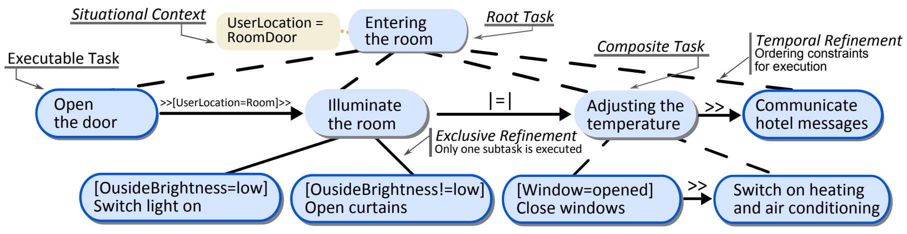

Situational Context
UserLocation = RoomDoor
Entering the room
Root Task
Executable Task
Composite Task
Temporal Refinement
Ordering constraints for execution
Open the door
>>[UserLocation=Room]>>
Illuminate the room
\|=\|
Adjusting the temperature
>>
Communicate hotel messages
Exclusive Refinement
Only one subtask is executed
[OusideBrightness=low] Switch light on
[OusideBrightness!=low] Open curtains
[Window=opened] Close windows
>>
Switch on heating and air conditioning

Fig. 4 *Entering the room* service

Figure 4 shows the modelling of the behaviour for the *entering the room* service (used as example in previous sections) using the proposed task model. This service is modelled for the needs of *the business traveller* persona. The root task represents the service and has an associated situational context whose fulfilment starts the execution of the service (*UserLocation=RoomDoor*). This situational context defines the context conditions that are produced in an implicit interaction. The root task is broken down into more specific tasks until they are executable tasks (i.e., tasks that can be executed by a media object). For each executable task, it is indicated in which obtrusiveness levels can be executed, and also, which media objects must be used in each one of those obtrusiveness levels to carry out the task (this is next further explained).

A task can have a context precondition (represented between brackets), which defines the context conditions that must be fulfilled so that a task is performed (if the precondition is not fulfilled, the task will not be executed). For instance, the *Switch light on* task has as a precondition that the outside brightness is low.

To break a task down, *temporal* or *exclusive* refinements can be used. On the one hand, using *temporal* refinements, all the subtasks are executed following a temporal order; in this case, tasks are related between them using temporal restrictions based on the ones proposed in [20]. For instance, as shown in Fig. 4, the service is decomposed into four subtasks using temporal refinements. This means that, when user is detected in front of his/her hotel room, its door will be opened; then, when the user enters inside, the room will be illuminated and the temperature will be adjusted; and finally, the hotel messages will be communicated to the user. On the other hand, using *exclusive* refinements only one subtask will be executed. For instance, the *illuminate the room* composite task is decomposed into two subtasks by using exclusive refinements; therefore, to illuminate the room either the lights will be switched on or the curtains will be opened.

To specify the context conditions (in the situational context, task preconditions and relationships), we use logical expressions grounded on elements of an ontology-based context model (we use the one presented in [25]). The logical expressions combine any number of basic expressions linked by the following logical connectives: and (AND), or (OR), equalities (=), inequalities (!=) and greater (>), or less than (<). Note that these context conditions allow services to be adaptive to user's context by means of specifying the different variations in a context-adaptive way.

2.2.2 Specifying the obtrusiveness space.

To adjust the obtrusiveness of the ambient services, we make use of the conceptual framework for defining implicit interactions presented in [13]. Specifically, we determine the obtrusiveness level of each executable task of the services. This framework defines two dimensions to characterize interactions: *initiative* and *attention*. According to the *initiative* factor, interaction can be *reactive* (the user initiates the interaction) or *proactive* (the system takes the initiative). With regard to the *attention* factor, an interaction can take place at the *foreground* of user attention (the user is fully conscious of the interaction) or at the *background* (unadvised interaction). For this work, we have divided the attention axis in three segments as shown in Figure 5 which are associated with the following values: *invisible* (there is no way for the user to perceive the interaction), *slightly-appreciable* (usually the user would not perceive it unless he/she makes some effort), and *completely-awareness* (the user becomes aware of the interaction even if he/she is performing other tasks). This conceptual framework defines the space of possible interaction styles (e.g., those that require more or less attention and more or less initiative).

There is different ways to accomplish a service task with different degrees of attentional demand and initiative. The selection of one will depend on the user needs and preferences. Furthermore, all the tasks do not have sense in all the quadrants. For example, the task *Switch light on* has more sense in reactive levels (e.g., when the user turns on the lights or the lights are turned on when the user enters the room). Designer not only decides what action needs to

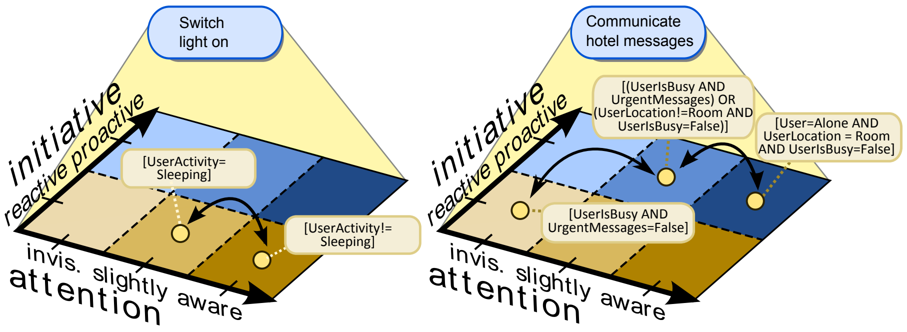

Switch light on
initiative
reactive proactive
[UserActivity=Sleeping]
[UserActivity!=Sleeping]
invis. slightly aware
attention
Communicate hotel messages
[(UserIsBusy AND UrgentMessages) OR (UserLocation!=Room AND UserIsBusy=False)]
[User=Alone AND UserLocation = Room AND UserIsBusy=False]
[UserIsBusy AND UrgentMessages=False]

Fig. 5 Selection of the obtrusiveness levels for two tasks

occur by means of tasks, but also, very importantly, the manner in which it should take place by means of selecting the possible obtrusiveness levels in which each executable task can be performed. Thus, designers have to describe the obtrusiveness space for each different executable task by defining in which obtrusiveness levels it can be executed, and the context conditions that make the system to choose the adequate level at each moment. By means of the obtrusiveness space, designers can better match an appropriate interaction to the situation at hand. It is worth noting that the obtrusiveness space is specified for each executable task individually, independently of the service to which it belongs, and considering the objectives specified in each persona.

Figure 5 illustrates the linkage between the executable tasks "Switch light on" and "Communicate hotel messages" (shown in Figure 4) and the obtrusiveness space for the interactions that support these tasks. On the one hand, the task to "switch light on" is performed in different levels of attention depending on the user activity. Note that this task can be executed in other services (such as for waking up the user), and the obtrusiveness space depends only on the task and the persona, not on the service in which the task is performed. Thus, this task can regulate the light using the gradual lighting when the user is sleeping. However, when the user is not sleeping, the task is performed at the completely-awareness level of attention and, therefore, it uses the regular lights. This task is always performed in a *reactive manner* with regard to the initiative axis since the task is executed in reaction to the user detection. On the other hand, the second task provides the hotel messages to the user. Depending on the urgency of the messages, the location of the user and the engagement of the user in other activities and his/her company, the task is executed in a different obtrusiveness level. The system informs the user about the messages *proactively* in a notorious manner requiring a high level of attention (*completely-awareness*) if the user is alone in the room and is not busy (e.g., ambient sound can be used). However, if the user is not in the room and is not busy, or is busy but the messages are important, the system informs the user about these messages in a subtle manner (e.g., using his mobile device). Otherwise, if the user is busy and the messages are not urgent, the task is carried out in an *invisible manner* without explicitly notifying to the user, only sending the messages in background to the user mobile device.

Once the context conditions of the obtrusiveness levels selected are defined, we map the obtrusiveness levels to the appropriate media object that support the underlying system. This is illustrated in the following subsection.

2.3 Mapping Obtrusiveness to Media Objects

Once identified the service's tasks and their obtrusiveness levels, designers indicate the media objects that are required for supporting a task in the different obtrusiveness levels in which the task can be performed. We consider a media object as a physical object that stimulates any human sense by using ambient media (e.g., mobile vibration, TV display, lights, etc.) [16]. Each media object is controlled by means of a driver that provides the atomic operations that describe the media object functionality. Designers can model these drivers in a technology-independent way by using the method proposed in [25], which allows automatically generating Java/OSGi drivers from a design model. In this model, the available drivers, their operations and its parameters are defined at a high level of abstraction.

Our approach interacts with the media objects by means of these drivers. In this way, different tasks could use the same media object with a different purpose only by passing it different parameters or using a different operation. For example, the *ambient sound* media object could be used either by the "wake-up call" task (starting the radio) or by the "communicate hotel messages" task (reproducing the new messages).

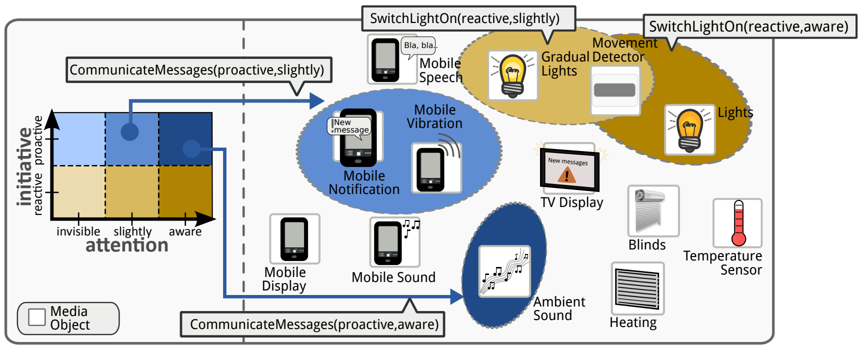


Fig. 6 Implication of the media objects with the attentional demand

Figure 6 shows the part of the media objects that supports some tasks from the "entering the room" service explained in Section 2.2. Media objects are represented by a square. For example, the *ambient sound* of the room is used for the "communicate hotel messages" task when the task is performed proactively at the highest level of attention. This mapping between the *ambient sound* and the obtrusiveness level for the "communicate hotel messages" task is expressed as *CommunicateMessages(proactive,aware)* where the first coordinate refers to the initiative and the second one to the attention axis (see Figure 6). However, the *mobile notification* and the *mobile vibration* are used for the same task when it is performed at the slightly level of attention. Another example is the "switch light on" task. The *gradual lights* that regulate light's intensity are used for this task when it is performed at the slightly level of attention. Conversely, the *regular lights* are used when the task is performed at the highest level of attention (i.e., *completely-awareness*). Figure 6 illustrates all these mappings for the tasks.

In order to specify which media objects support a certain task for a given obtrusiveness level, the *Superimposition operator* ($\odot$) is defined. The Superimposition operator takes a task and an obtrusiveness level and returns the set of media objects, their operations and their parameters required for the task (from one to many). Some examples of the relationship between the obtrusiveness level for the *communicate hotel messages* task and its mapping with the media objects (see Fig. 6) are as follows:

$$\odot CommunicateMessages(proactive, slightly) =$$
$$\{mobile\_notification.notify\_message(message), mobile\_vibration.soft\_vibration\}$$
$$\odot CommunicateMessages(proactive, aware) =$$
$$\{ambient\_sound.reproduce\_message(message)\}$$

Designers must define which system media objects are associated with each obtrusiveness level for each executable task. In order to avoid complexity, this mapping can be defined in an intensional manner (not all the obtrusiveness combinations must be explicitly defined by the designers).

3 Executing Unobtrusive Personalized Services at Runtime

The models proposed in the previous section allow unobtrusive personalized services to be described regardless the particular technology used for the implementation. To achieve the execution of such services, these models are prepared to be interpreted at runtime. Thus, all the effort invested at design time is reused at runtime providing new opportunities for adaptation capabilities without increasing development costs.

With this purpose, we have designed and developed a software infrastructure[1] that interprets the models at runtime to automate the personalized ambient services adjusting their obtrusiveness level as needed. Note that, in this way, the services are only represented in the models, therefore, the models are the primary means to understand, interact with, and modify the services.

Figure 7 shows an overall view of the components involved in this infrastructure. These components are: the *runtime models*, which provide all the

[1] A video of the software infrastructure in execution is provided in http://www.pros.upv.es/routines

needed knowledge for carrying out the unobtrusive personalized services; an *Automation Engine*, which is in charge of managing the context changes and executing the services in an unobtrusive way by interpreting the models at runtime; and the *Managed System*, which provides the media object drivers for performing the services' tasks in a considerate manner.

This software infrastructure is designed to provide a loosely-coupled and model-based solution by decoupling technology-independent models, unobtrusive automation process, and ambient media objects. In this way, we can define how the different tasks involved in a service are adapted to user's context in terms of obtrusiveness (by means of an automation engine) using the appropriate media objects (managed system) by means of executable models (runtime models). The software infrastructure applies the following process for executing a designed unobtrusive personalized service:

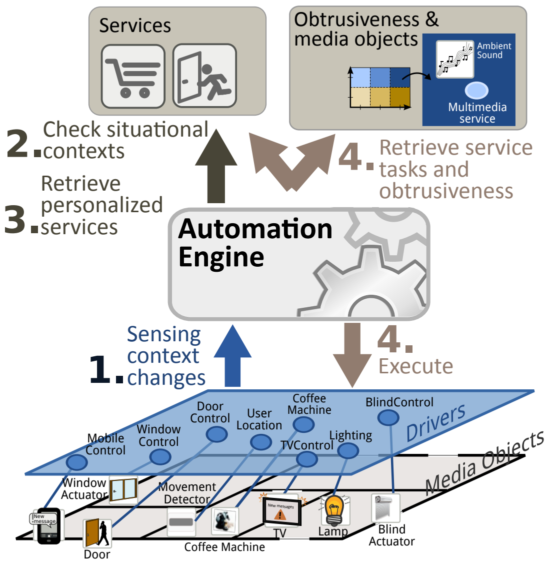


Fig. 7 Runtime infrastructure

1. *Sensing context*: The media objects deployed in the smart environment sense context changes. These media objects are also capable of identify each user. For instance, when Bob is approaching to his room, his location and identification are sensed by RFID and presence detector objects.
2. *Retrieving personalized services*: Once a context change is detected, the services defined for the identified user according to her/his persona are retrieved. For instance, since Bob location has changed, the services described for his persona (*the business traveller*), such as the *Entering the room* service described in Section 2.2, are retrieved.

3. *Checking situational contexts*: The situational context of the retrieved services (e.g., *UserLocation=RoomDoor*) are checked to see if any of them is satisfied and the corresponding service has to be executed.
4. *Executing personalized services*: If the situational context of a service is satisfied, the service is carried out in a considerate manner. For instance, when Bob location is in front of his room door, the situational context of the *Entering the room* service is satisfied and the service is carried out. To achieve this, the following steps are executed:

   – *Retrieving executable tasks and obtrusiveness:* The different executable tasks of the service are progressively retrieved according to their refinements, the temporal relationships specified among its tasks and the up-to-date context information. After retrieving an executable task, if the precondition is satisfied (considering it is true if the task has not a precondition), the obtrusiveness space described for the task is retrieved.
   – *Adjusting the obtrusiveness level:* The context conditions specified in the obtrusiveness space are checked to determine in which obtrusiveness level the task must be executed. For instance, for the *communicate hotel messages* task, when the user is not busy and is alone, the task must be carried out in the *proactive-aware* obtrusiveness level (see Figure 5). Specifically, the *reproduce_message* operation with a *message* as a parameter is used for this media object.
   – *Using the appropriate media objects:* The media objects that must be used to perform the task for that obtrusiveness level are searched for. Given a task and an obtrusiveness level, the mapping defined between media objects and obtrusiveness levels returns the media objects to be used, their operations and their corresponding parameters. For instance, for the *communicate hotel messages* task the *ambient sound* is used in the *proactive-aware* obtrusiveness level (see Fig. 6).

Figure 8 shows how the *entering the room* tasks are planned and executed in a hot sunny day, when the user is alone, and he is not busy. Red tasks represent tasks not executed due to their context conditions are not satisfied. The green ones represent tasks that are executed by using the appropriate media objects according to the user attentional demands and context. Specifically, the *open the door* task is executed in a completely aware manner using an electronic lock and making a beep sound when it is opened, the room is illuminated by raising the blinds because the outside brightness is high, the temperature is adjusted by switching on the air conditioner (the *close windows* task is not executed due to windows was already closed), and finally the *hotel messages* are communicated to the user in a notorious manner (completely-aware level of attention) using the ambient sound due to the user is alone and he is not engaged in an important task.

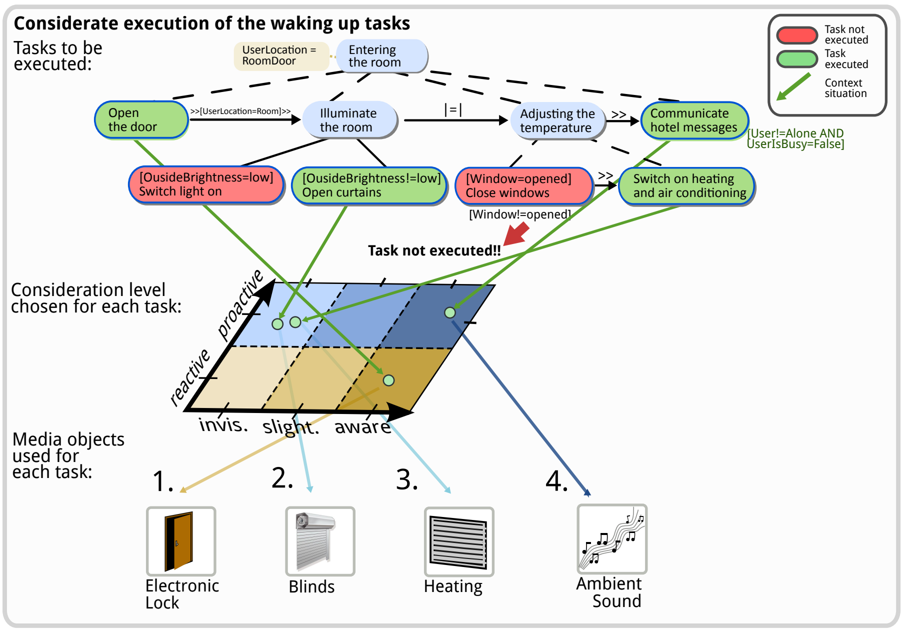

Considerate execution of the waking up tasks
Tasks to be executed:
UserLocation = RoomDoor
Entering the room
Open the door
>>[UserLocation=Room]>>
Illuminate the room
|=|
Adjusting the temperature
>>
Communicate hotel messages
[User!=Alone AND UserIsBusy=False]
[OusideBrightness=low] Switch light on
[OusideBrightness!=low] Open curtains
[Window=opened] Close windows
>>
Switch on heating and air conditioning
[Window!=opened]
Task not executed!!
Task not executed
Task executed
Context situation
Consideration level chosen for each task:
reactive
proactive
invis.
slight.
aware
Media objects used for each task:
1.
2.
3.
4.
Electronic Lock
Blinds
Heating
Ambient Sound

Fig. 8 Considerate execution of the *entering the room* service

3.1 Implementation Details

Our infrastructure is implemented in Java/OSGi[2] technology and must be run in an OSGi server together with the media object drivers. Using OSGi, the engine can listen the changes produced in the media objects to detect context changes. Also, the engine can search for the media object drivers associated to the task in the OSGi server by using its service registry and the Java Reflection capabilities. Context is represented by means of OWL ontologies[3], while services and obtrusiveness spaces are represented in the XML Metadata Interchange standard (XMI)[4]. To query the models involved in our approach at runtime, the engine uses the Eclipse Modelling Framework[5] (EMF) that allows a system to work with any XMI model by querying its structure dynamically at runtime. More detail about these operations can be found in [5].

For the media objects provided by a mobile device, we have developed a web service based on Android[6] and the Restlet Framework[7]. This service listens to the mobile client requests and reply to them.

[2] http://www.osgi.org/
[3] http://www.w3.org/TR/owl-features/
[4] http://www.omg.org/spec/XMI/
[5] www.eclipse.org/modeling
[6] http://www.android.com/
[7] http://www.restlet.org/

4 Validation of the Proposal

In order to validate our proposal, we have applied a case study based evaluation by following the research methodology practices provided by Runeson and Host [23]. These practices describe how to conduct and report case studies and recommend to design and plan the case studies before performing them.

By using these practices, we have designed and developed a smart hotel whose overall purpose is to make guests stay comfortable and save energy consumption. The case study illustrates how personalized services are executed by interacting with users when needed adjusting the obtrusiveness level. The approach takes into account user's needs and the needed context information such as user preferences, location, message urgency, etc. All these factors have an effect in the execution of the personalized services' tasks and the obtrusiveness level to properly interact with users. Next, we present the design of the case study and the evaluation results.

4.1 Design of the Case Study

The case study gives support to several scenarios of a guest in a smart hotel. In these scenarios, several personalized services take place adapting their obtrusiveness according to context by using different media objects. The running example used along the paper forms part of them. As an example, one of the scenarios is described as follows: Alice has booked a room in the Royal Station Hotel of NewCastle in England where she is attending Pervasive 2012. When she first enters in the hotel at 12:30 p.m., the hotel detects her mobile phone to integrate its functionality with the hotel services. In that moment, Alice receives in her mobile phone a message informing her that her room is the 302 room and showing her the hotel map with indications to arrive to that room. The room is fitted out according to the preferences that Alice indicated when she registered online from the internet. When Alice is approaching to the 302 room, the door detects her presence and is automatically opened. As soon as she enters inside, the curtains are opened and the heating, which was at the save-energy mode, increments the temperature to 21 °C. The ambient sound of the room welcomes her and, since it is lunch time and she indicated that she wanted to have lunch in the hotel, she is asked whether she prefers the room service takes her the menu or to go down to the restaurant. She is tired because of the trip, so she chooses the first option, receiving the message that her meal will be ready in 10 minutes. After having lunch watching the TV, Alice lays down for resting a little; therefore, the curtains are closed and the volume of the TV is turned down. One hour later, she wakes up and the curtains are slowly opened again. Then, she gets ready and leaves the room for sightseeing the city. Once the door is closed, the TV is switched off and the heating is changed to save-energy mode.

In the case study, the personalized service tasks are presented at different obtrusiveness levels according to the user context. In this way, users could evaluate the execution of the personalized services and its obtrusiveness. For example, in this scenario, the messages provided to Alice are offered *proactively* in a *completely aware* manner when Alice is not busy and she is alone. However, the same task is performed in other scenario in which Alice enters the room with someone. In this scenario, these messages were communicated to Alice more privately, by sending her a mobile sms with sound notification if it is an important message (*proactively* and *slightly aware*), or without notification if it is not (*proactively* and *invisible*).

With this case study we want to validate that our design strategy allows unobtrusive and personalized services to be described. In particular, we wanted to evaluate whether the personalized services were automated adequately and whether guests were satisfied with the result of the ambient media environment. To achieve this, we first developed a prototype version for the described services by following the design steps. We then conducted an experiment[8] in which users performed several scenarios such as the one described above, adopting the Alice's role. In the experiment, we simulated the different activities of the scenarios. The experimental setup included a scale environment with KNX devices[9] to represent the media objects of the Smart Hotel, a PC running Equinox and a HTC Magic mobile device running Android Operating System. A total of 15 subjects between 23 and 54 years old participated in the experiment (7 female and 8 male). Most of them had a strong background in computer science.

To collect and analyse the results of the two evaluated dimensions (*user satisfaction* and *personalized services automation*), we used an adapted IBM Post-Study Usability questionnaire [15]. Although it is focused on measuring user satisfaction with system usability, we also included other questions regarding the execution of the personalized services and its obtrusiveness level. We applied a Likert scale (from 1 to 5 points) to evaluate the items of the questionnaire.

4.2 Evaluation Results

In this section, we present the evaluation results of the case study. Overall, the experiment showed that by following our method, user personalized services can be executed adjusting its obtrusiveness level according to context.

Figure 9 shows a summarized table of the obtained results. Regarding *user satisfaction*, most users (85%) were generally satisfied with the system and how to use it. Most of them also answered that the system helped them to complete the tasks and, therefore, they could perform the tasks of the scenarios effectively and quickly. Although most users (83%) thought that it was easy to learn to use the system, they complained that they did not know how to recover the mistakes because it was not clear. Some of them also said that they would be more comfortable with the system if it allows them to change the way

[8] Screenshots in http://www.pros.upv.es/routines

[9] http://www.knx.org

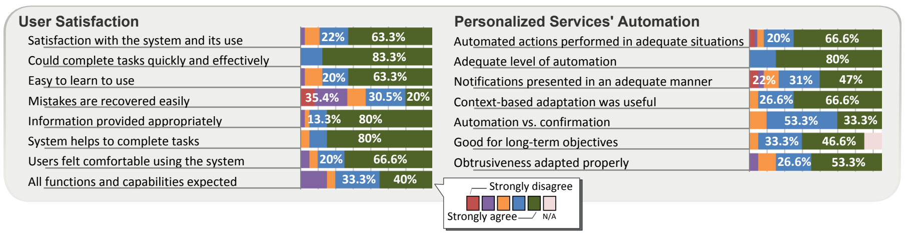


Fig. 9 Summarized results of the evaluation

some service tasks were performed. In the results, the information provided by the ambient services was considered appropriate by the 93% of the subjects. Regarding the functions and capabilities provided by the system, some users said that they expected the system let them to personalize the services and also that it should learn about the service tasks they do not like (e.g., two users said that they preferred the TV volume was not turned down because they liked sleeping watching the news).

Regarding the *personalized services' automation*, 86% of subjects strongly agreed or agreed in that automated actions were performed in appropriate situations interacting with them only when needed. They said that this helped them to perform the personalized service tasks described in the scenario. In addition, most users agreed that the level of automation was adequate and the system generally asked them for confirmation when it was essential. However, some users complained because they would like to personalize the media objects used for some tasks (22%); for instance, some of them said that they would prefer having received the messages in a subtle manner (e.g., through the television or their mobile phone) because the ambient sound frightened them a little since they did not expect to receive messages when they entered to the room. In spite of this, most users considered the context-based adaptation was useful and the obtrusiveness adaptation for personalized service tasks was adequate. Furthermore, users considered the system good for long-term objectives because it could help them to save natural resources. In addition, most of them commented that they would like to have this system for automating their home ambient services.

The performed evaluation was useful because allowed us to detect some drawbacks that will be improved in future work such as personalization and self-learning. However, although the initial results of the experiment are quite good, additional experimentation would be required to analyse the user acceptance of the system in real environments where the context sensing is not perfect and users compete with other distractions.

5 Related Work

This work supports the design and automation of unobtrusive personalized services that interact with users by using the appropriate ambient media objects at each situation.

Several approaches have proposed the use of models at runtime for software adaptation [4]. Bencomo et al. [1] propose the use of architectural models to support the generation and execution of adaptive systems. The authors represent the adaptation policies under the form of a state-transition system in which the states correspond to the system configurations and the transitions correspond to the adaptations between these configurations. Morin et al. [18] propose a combination between model-driven and aspect-oriented techniques to support dynamic runtime reconfiguration. They dynamically compose aspects to produce a set of configuration models and then use these models to generate the scripts needed to adapt a running system from one runtime configuration to another. Blumendorf et al. [2] focus on the development of adaptive user interfaces and their adaptation at runtime combining multiple models. These models are self-contained to ensure executability. This approach allows developers to monitor, maintain, manipulate and extend interactive applications at runtime and thus manage the continuously changing requirements of user interface development. However, none of these approaches have focused on supporting personalized services taking their obtrusiveness level as a primary concern and, therefore, none of them take into account the intrusiveness of the system behaviour. We address a different issue that is more related to human limitations (e.g., user attention) than device technical limitations (e.g., screen size).

Towards creating systems that adapt their level of intrusiveness to the context of use, works are mainly focused on minimizing unnecessary interruptions to the user [22]. Moreover, the amount of studies concentrating on the design of unobtrusive interactions is still limited. Approaches in the area of Considerate Computing [8] are mainly focused on inferring attention in order to predict acceptability. Horvitz et al. [12] demonstrated the potential use of Bayesian networks for computing the cost and value of interruptions. Hinckley and Horvitz [10] modelled interruptibility by considering the user's likelihood of response and the previous and current activity. Ho and Intille [11] suggested that proactive messages delivered when the user is transitioning between two activities may be received more positively. Vastenburg et al. [29] conducted a user study of acceptability of notifications to find out what factors influence the acceptability of them. Although these initiatives recognize the need to adapt the interaction, efforts have been put on minimizing unnecessary interruptions, overlooking automation and adaptation aspects that we have taken into account. Moreover, we propose to represent behaviour aspects in the interaction design level in order to define how the different interactions are adapted when a particular situation is produced.

In addition, research on smart objects and the Internet of Things [7] has been growing on the focus of technology- and business-focused research on

RFID, smart objects, and smart products [28]. Work on smart objects has focused on technical aspects (hardware platforms, software infrastructure, etc.) and application scenarios [26]. Mattern [17] defined smart objects as everyday artifacts capable of communicating with people and other smart objects, and also capable of discovering where they are and which other objects are in the vicinity. Norbert et al. [27] distinguished between two types of smart objects: system-oriented that importunate smartness and people-oriented that empower smartness. Recently, Kortuem et al. [14] have explored the smart-object design space for identifying canonical smart-object types. Also, human-interface aspects of smart-object technology are just beginning to receive attention [19]. Conversely, in this work, we provide smart objects with consideration capabilities in order to achieve a seamless interaction with the environment. In addition, since models are technology-independent, the media objects that a task must use for interacting with the user can be easily changed by updating the models accordingly.

6 Conclusions and Further Work

In this work, we have presented and evaluated a strategy for describing and automating unobtrusive personalized services in an ambient media environment. The provided models describe how services must interact with each persona in an unobtrusive way. Using these abstract models, the system is designed by using concepts that are much less bound to the underlying implementation technology and are much closer to the problem domain [21]. This makes the models easier to specify, understand, and maintain than code.

Our approach also provides a software infrastructure that interprets these models at runtime to execute the needed services in such a way that they demand the adequate user attention in each situation. Thus, the models become into the primary means to understand, interact with, and modify the personalized services and their obtrusiveness level. This considerably facilitates the evolution of the system at runtime: as soon as the models are changed, the evolutions are applied by the software infrastructure [5]. In addition, by handle obtrusiveness as a separate concern, a given ambient service can be presented to the user in a complete different manner just by changing the obtrusiveness specification, without altering the service description. Moreover, since the models that form the basis for adaptations are available at design time, they can be extensively exploited for validation and verification. Also, runtime probabilistic model checking [6] can be applied to verify system requirements at runtime.

However, as the evaluation results shown, the proposed method has also some drawbacks that should be improved in further work. The main one is that, although the execution of the services takes into account the preferences indicated by guests, they should be able to personalize both the ambient services and their obtrusiveness level. For this reason, we plan to develop an end-user tool that facilitates to perform this personalization. Using this tool, guests not only will be able to configure what is already designed, but also they could add some personal automations that could not be taken into account before. Also, with this tool, users could change the media objects to be used for each task. It is worth noting that, since our approach allows evolution to be performed by only changing the models, this tool will not have to change any line of code to apply users personalization. Further work will be also dedicated to provide the software infrastructure with self-learning capability for detecting if users execute tasks for compensating any automated task. To achieve this, the media objects should provide information about their operations that perform contradictory tasks.

Acknowledgements This work has been developed with the support of MICINN under the project EVERYWARE TIN2010-18011 and co-financed with ERDF, in the grants program FPU, and also with the support of the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria.

References

1. Bencomo, N., Grace, P., Flores-Cortés, C.A., Hughes, D., Blair, G.S.: Genie: supporting the model driven development of reflective, component-based adaptive systems. In: ICSE, pp. 811–814 (2008)
2. Blumendorf, M., Lehmann, G., Albayrak, S.: Bridging models and systems at runtime to build adaptive user interfaces. In: Proc. of EICS 2010, pp. 9–18. ACM (2010)
3. Brown, D.M.: Communicating Design: Developing Web Site Documentation for Design and Planning (2nd Edition). New Riders Press (2010)
4. Calinescu, R.: When the requirements for adaptation and high integrity meet. In: Proceedings of the 8th workshop on Assurances for self-adaptive systems, ASAS '11, pp. 1–4. ACM, New York, NY, USA (2011)
5. E.Serral, P.Valderas, V.Pelechano: Supporting runtime system evolution to adapt to user behaviour. In: Proc. of CAiSE'10, pp. 378–392 (2010)
6. Filieri, A., Ghezzi, C., Tamburrelli, G.: Run-time efficient probabilistic model checking. In: Proceedings of the 33rd International Conference on Software Engineering, ICSE '11, pp. 341–350. ACM, New York, NY, USA (2011)
7. Gershenfeld, N., Krikorian, R., Cohen, D.: The Internet of Things. Scientific American **291**(4), 46–51 (2004)
8. Gibbs, W.W.: Considerate computing. Scientific American **292**(1), 54–61 (2005)
9. Gulliksen, J., Goransson, B., Boivie, I., Blomkvist, S., Persson, J., Cajander, A.: Key principles for user-centred systems design. Behaviour & Information Technology **22**, 397–409 (2003)
10. Hinckley, K., Horvitz, E.: Toward more sensitive mobile phones. In: Proc. of the UIST '01, pp. 191–192 (2001)
11. Ho, J., Intille, S.S.: Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In: Proc. of CHI '05, pp. 909–918. ACM (2005)
12. Horvitz, E., Kadie, C., Paek, T., Hovel, D.: Models of attention in computing and communication: from principles to applications. Commun. ACM **46**, 52–59 (2003)
13. Ju, W., Leifer, L.: The design of implicit interactions: Making interactive systems less obnoxious. Design Issues **24**(3), 72–84 (2008)
14. Kortuem, G., Kawsar, F., Fitton, D., Sundramoorthy, V.: Smart objects as building blocks for the internet of things. Internet Computing, IEEE **14**(1), 44 –51 (2010)
15. Lewis, J.R.: Ibm computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. Int. J. Hum.-Comput. Interact. **7**(1), 57–78 (1995)
16. Lugmayr, A., Risse, T., Stockleben, B., Laurila, K., Kaario, J.: Semantic ambient media an introduction. Multimedia Tools and Applications pp. 337–359 (2009)

17. Mattern, F.: From smart devices to smart everyday objects. In: Proc. Smart Objects Conf. (SOC 03), pp. 15–16. Springer-Verlag (2003)
18. Morin, B., Barais, O., Jezequel, J.M., Fleurey, F., Solberg, A.: Models@ run.time to support dynamic adaptation. Computer **42**(10), 44 –51 (2009)
19. Nelson, L., Churchill, E.F.: User experience of physical-digital object systems: Implications for representation and infrastructure. paper presented at smart object systems workshop, in cojunction with ubicomp 2005 (2005)
20. Paternò, F.: Concurtasktrees: An engineered approach to model-based design of interactive systems. In: L.E. Associates (ed.) The Handbook of Analysis for Human-Computer Interaction, pp. 483–500 (2002)
21. Paternò, F.: From model-based to natural development. HCI International pp. 592–596 (2003)
22. Ramchurn, S.D., Deitch, B., Thompson, M.K., Roure, D.C.D., Jennings, N.R., Luck, M.: Minimising intrusiveness in pervasive computing environments using multi-agent negotiation. MobiQuitous '04 pp. 364–372 (2004)
23. Runeson, P., Hst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Soft. Eng. **14**(2), 131–164 (2009)
24. Schmidt, A.: Implicit human computer interaction through context. Personal Technologies **4**(2-3), 191–199 (2000)
25. Serral, E., Valderas, P., Pelechano, V.: Towards the model driven development of context-aware pervasive systems. PMC **6(2)**, 254–280 (2010)
26. Siegemund, F.: A context-aware communication platform for smart objects. In: In Proc of the Int Conf on Pervasive Computing, pp. 69–86. Springer-Verlag (2004)
27. Streitz, N.A., Rocker, C., Prante, T., Alphen, D.v., Stenzel, R., Magerkurth, C.: Designing smart artifacts for smart environments. Computer **38**(3), 41–49 (2005). DOI 10.1109/MC.2005.92. URL http://dx.doi.org/10.1109/MC.2005.92
28. Thiesse, F., Kohler, M.: An analysis of usage-based pricing policies for smart products. Electron. Market. **18**(3), 232–241 (2008). DOI 10.1080/10196780802265751. URL http://dx.doi.org/10.1080/10196780802265751
29. Vastenburg, M.H., Keyson, D.V., de Ridder, H.: Considerate home notification systems: a field study of acceptability of notifications in the home. Personal and Ubiquitous Computing **12**(8), 555–566 (2008)