AwareNotifications: Multi-Device Semantic Notification Handling with User-Defined Preferences

Journal of Ambient Intelligence and Smart Environments 0 (0) 1–17
IOS Press

AwareNotifications: Multi-Device Semantic Notification Handling with User-Defined Preferences

Fulvio Corno [a], Luigi De Russis [a] and Alberto Monge Roffarello [a,*]
[a] *Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi, 24 Torino, 10129 Italy*
E-mails: fulvio.corno@polito.it, luigi.derussis@polito.it, alberto.monge@polito.it

Abstract. With the increase of connected devices and online services, the number of notifications received by each person is growing. Although notifications are useful to inform users about important information such as new messages and events, the continuous interruptions, the notification duplication, and the rigid delivery can be sources of discomfort. To overcome these issues, we present AwareNotifications, an intelligent system based on user-defined preferences to manage multi-device notifications. AwareNotifications is powered by Semantic Web technologies. By directly exploiting user preferences in the semantic reasoning process, the system is able to identify suitable device(s), modality, and moment(s) to deliver the incoming user notifications. We evaluated AwareNotifications in a user study with 15 participants, in which we compared our system with the "traditional" notification delivery system. The study confirms the perceived effectiveness of AwareNotifications, and provides insights to further improve the system.

Keywords: Internet of Things, Notifications, Notification Delivery, Semantic Web, Reasoning

1. Introduction

Nowadays, mobile devices like smartphones and tablets are becoming increasingly common. Using the Internet, they allow users to stay connected anytime and anywhere. Moreover, the number of smart objects increases every day, ranging from wearable devices to smart appliances, like thermostats and fridges. All these devices contribute to create the Internet of Things (IoT) world and bring us closer to the realization of Smart Environments. In this context, notifications are widely adopted. Most mobile applications use notifications to communicate with users. Applications like Facebook or Twitter can generate a great number of notifications, e.g., they can notify the user about an incoming message or a friend's new post. IoT devices provide their custom mobile applications, which are mainly used to show device-generated information as notifications. For example, smart thermostats notify the user when the temperature drops below a given threshold, while a smart fridge alerts the householders if the door is left open.

Consequently, the number of notifications received by a user is growing, and their user experience is put to a hard test [1]. Although notifications are useful to inform users about important information such as new messages and events [2], the continuous interruptions can be a source of discomfort. In fact, while receiving more messages and social network updates make users feel more connected with others, an increasing number of notifications is associated with an increase in negative emotions [3]. Furthermore, it is demonstrated that not all notifications have the same importance [4, 5]. In particular, users mostly dismiss notifications that are perceived as not useful [5, 6]. Past studies [7–11] have shown that the user interruptibility depends on numerous factors, such as current activity, so-

*Corresponding author. E-mail: alberto.monge@polito.it.

cial engagement, notifications modality, user location, notification sender, etc. However, the "traditional" notification system delivers notifications independently of user preferences and contexts. In addition, the disruptive effects of notifications are amplified by the increasing number of devices, as stated in [12]. In this multi-device scenario, it is common to receive duplicate notifications on several devices, or to receive them in an inappropriate moment, or on an unsuitable device. Think of a new Facebook private message: it is sent to all user devices with an installed Facebook application and to all logged-on browser windows, not to speak about desktop notifications, supported by some browsers. Moreover, think of a user who is driving her car; notifications sent to her smartphone could disturb her driving activity.

From the literature, three issues emerge in the handling of "traditional" notifications: a) unconditional delivery, b) limited customization, and c) mono-device management, The contribution of this paper is a new approach for allowing users to customize notification delivery by taking into account their preferences and context, thus being able to tackle the previous three issues. To validate the approach, we propose AwareNotifications, an intelligent system based on user-defined preferences to manage multi-device notifications delivery. The system is designed starting from a careful analysis of existing works about notifications. It is composed by a series of clients (user devices and IoT devices) and a server. Clients are in charge of collecting the user preferences and the context information. The AwareNotifications server collects the information deriving from the clients, and delivers incoming notifications by semantically reasoning about preferences and context. We assessed the perceived effectiveness of AwareNotifications by comparing it with the "traditional" notification delivery system in a user study with 15 participants. In particular, we were interested in evaluating whether a multi-device system that takes into account both user preferences and context *improves notification delivery* in an effective way. In the study, users simulated the reception of notifications with and without AwareNotifications on multiple devices, evaluating any received message with suitable questionnaires. Results show that AwareNotifications improves the user experience with notifications in terms of notification acceptability, accessibility of the notification content, and notification delivery. Interviews with the participants highlighted the pros and cons of the evaluated solutions ("traditional" notification delivery system vs. AwareNotifications), and confirmed the perceived effectiveness of the approach.

The remainder of this paper is organized as follows: Section 2 reviews the related literature about the issues with notifications handling. Section 3 describes the guidelines and the motivations that guided our work. Section 4 present the semantic model we designed to be exploited by the AwareNotifications system, whose architecture is then described in Section 5. Section 6 reports the implementation of the first version of AwareNotifications, while Section 7 presents the user study we conducted to evaluate the effectiveness of the proposed approach, with the related results and discussion. Eventually, Section 8 concludes the paper and presents future works.

2. Related Work

Our work relates to the management of user notifications. We analyzed the literature about the main problems caused by the increase of notifications (Section 2.1), and about proposed smart notification systems (Section 2.2).

2.1. The Problem of Overwhelming Notifications

The problem of overwhelming notifications has been analyzed in numerous recent studies. Although notifications are useful to inform users about important information such as new messages and events [2], delivering notifications runs the risk of interrupting the user's ongoing task, affecting users' performances, annoyance, and anxiety [7, 13, 14].

In particular, the work of Adamczyk et al. [7] measures the effects of interrupting a user at different moments in terms of current task performance, emotional state, and social attribution, thus demonstrating that different interruption moments have different impacts on user emotional state and positive social attribution. The authors found that the participants to their study felt higher workloads if notifications were delivered while they were in the middle of a work task, such as correcting or writing text, or conducting a web search.

In a similar way, in a study with 11 information workers, Czerwinski et al. [15] conclude that people find difficult to return to a previous task after having been interrupted by a notification. With a controlled experiment, Bailey and Iqbal [8] measure users workload during several interactive tasks, and demonstrate that the moment at which a notification is delivered af-

fects the user interruption cost. Furthermore, Mark et al. [16] report that interrupted tasks a) require more effort from the users and b) are more stressful and frustrating than normal tasks.

More recently, with a large-scale assessment of mobile notifications, Pielot et al. [3] report the disadvantages brought by the increasing number of notifications. They found that mobile phone users have to deal with a large volume of notifications every day, mostly from instant messaging services and e-mails. They demonstrate that the increasing number of notifications is correlated with negative emotions, such as stress and feeling overwhelmed.

Finally, it is demonstrated that not all notifications are of the same importance to the user. The study of Mashhadi et al. [4] reveals that users desire to have more fine-grained control over the notification management, specifying what is important to them. In a large-scale analysis of mobile notifications, Sahami et al. [5] derive an holistic picture of notifications, reporting a number of guidelines to effectively use them. Also in this case, the authors reports that not all notifications are important. In particular, they demonstrate that the importance of a notification depends on the application source.

In our work, we start to address these issues about the problem of overwhelming notifications, to design a system able to optimally manage user's notifications through user-defined preferences.

2.2. The Vision of a Smart Notification System

The literature analysis clearly shows the need of a smart notification system. Many works propose strategies to improve the user experience with notifications, like reducing interruptions, deferring notifications until the right time, and communicating (un)availability of users [3]. Rosenthal et al. [17] contribute with a method to learn when to mute a phone call or a notification to avoid embarrassing interruptions, demonstrating the need for personalized behaviors. To avoid abrupt full-screen notifications (e.g., an incoming phone call notification), Böhmer et al. [18] propose a smaller partial-screen layout for incoming messages.

However, the vision of a smart notification system does not seem completely reached by any previous work. Most of the studies found in the literature focus on user interruptibility, only. Given the complexity of creating systems to smartly interrupt users, the focus of these works is on understanding how to compute accurate costs of interruption. Some systems were developed to computationally reason about appropriate moments for interrupting users engaged in tasks (e.g., [19–22]). A statistical model that predicts the responsiveness of the users to instant messaging notifications has been created and tested by Avrahami and Scott [23]. Such a model uses data coming from notifications and desktop events to predict at what time the user will reply to an incoming notification. Ho et al. [24] developed a context aware mobile computing device that automatically detects activity transitions using wireless accelerometers. With the evaluation of the system, they support the strategy of using activity transitions as a trigger for non time-critical interruptions to potentially reduce feelings of information overload.

Other more recent studies focus on the usage of smartphones, analyzing factors such as notification acceptability and delivery instant. Pejovic and Musolesi [25] propose the design of an intelligent interruption mechanism to show notifications to the user at the right moment. The mechanism is based on a machine learning classifier for smartphones that exploits context information derived from the same device. Poppinga et al. [26] analyze the dependency of user interruptibility and many factors, such as user activity, time of the day, user location, etc. The study of Fisher et al. [9] states that suitable moments to deliver notifications arise at the endings of episodes of mobile interaction, like phone calls or text messages. Pielot et al. [27] consider the data deriving from smartphones usage to analyze user attention. In particular, they detect when a user is bored with a user-independent machine learning model.

Okoshi et al. [28] propose a middleware, named Attelia, to detects good breakpoints to deliver notifications. Attelia uses only smartphone data, and it has been evaluated with an in-the-wild study. Differently from other works, which consider mobile notifications only, Vastenburg et al. [29] conducted a user study in a living-room laboratory to analyze the acceptability of notifications in a home environment. They used a single display to show notifications in different moments and modalities. Another study focuses on the presentation of notifications to the user: Kern and Schiele [10] assert that the notification modality should be chosen according to the user interruptibility. Only a few studies do consider more than one device in the notification delivery process. In particular, Okoshi et al. [1] extend their previous work (Attelia) to Attelia II, and

detect breakpoints to deliver notifications when users use more than one mobile or wearable device.

However, at the best of our knowledge, in the literature there is no system that effectively manage all the notifications of a user, taking into account multiple devices, context, and user preferences at the same time. We built our work combining some guidelines defined after the analysis of the previous systems with innovative features (i.e., multi-device environment, user-defined preferences, and semantic knowledge representation).

3. Design Guidelines and Motivation

We present a simple scenario to explain our approach of an intelligent notification system in a multi-device and smart environment that takes into account user preferences.

> Mary, an architect, is in her office. In this place, there are four devices connected to the Internet: three Mary's personal devices (a smartphone, a tablet, and a laptop), and a smart TV, which Mary uses during her breaks or for meetings. Currently, Mary is working on her laptop. At the same time, a friend of Mary sends her a chat message. When Mary is working on the computer, she prefers to avoid distractions. Furthermore, she considers a chat message coming from a friend not so important. When Mary turns off the laptop, she switches the smart TV on, having a break.

By considering the preferences of Mary, a smart notification system should not deliver the chat notification when Mary is working, but it should maintain the notification pending until Mary starts her break. Upon detecting this change in devices status and current activity, a smart system should send the pending notification to the TV. Since Mary is probably looking at the screen, the notification should be sent with a visual representation.

AwareNotifications is a system able to manage situations such as those reported in the scenario. In the first part of our work, we defined 9 design guidelines (GLs) informed by the literature to design AwareNotifications, an intelligent system based on user-defined preferences for managing multi-device notifications delivery. These guidelines are reported in Table 1 and explained in the following sections.

3.1. Involved Technologies

Most of the previous studies about notifications analyzes large data sets of notification data to train machine learning algorithms. We chose to explore another way of manage notifications by exploiting the reasoning capabilities of the Semantic Web, in combination with a mechanism of user-defined preferences (GL1). These choices are motivated by literature findings. Some previous studies (e.g., [28, 29]), in fact, show the need of personalized behaviors in the management of notifications. In [30], the authors demonstrate that the interaction between users and intelligent algorithms is feasible, showing the potential of rich human-computer collaborations. In the same study, an experiment shows that a rule-based explanation paradigm of a machine learning algorithm is the most understandable for the participants. Finally, two previous works confirm that allowing the user to help an algorithm could make a crucial difference in terms of accuracy and understandability [31, 32].

3.2. Delivery Behavior

We take into account an important aspect that is too often neglected in previous studies, i.e., the actual notifications delivery. Additionally, few studies consider more than one device in the notification context (e.g., Attelia II [1]) and almost none of them includes IoT devices such as smart TVs [33], smart thermostats, etc. For this reason, AwareNotifications is designed to operate on a multi-device and smart environment (GL2). In agreement with the literature, we found that the modality (GL3) and the instant to show the notification to the user (GL4) are two factors that most influence the notification acceptability [9–11, 26, 29]. Consequently, in AwareNotifications, we adopted a rich notion of notification delivery, by considering three factors related to the notification delivery process:

- the target device;
- the notification modality (i.e., visual and/or audio);
- the delivery moment:
  - instantly;
  - between two activities, i.e., in a natural breakpoint.

In particular, in the target device selection, we preferred currently used devices (GL5). This choice is supported by some previous studies, which show that a good moment to present a notification to the user is when mobile devices are in use [4, 26, 27]. For what

Table 1
Design guidelines

Category	Guideline	Description
Involved technologies	GL1	AwareNotifications should allow users to define their preferences to customize the delivery process at any time.
Delivery behavior	GL2	AwareNotifications is designed for a multi-device environment. It should be able to dispatch notifications to all the different user devices.
	GL3	AwareNotifications should show notifications in different modalities, according to contextual information and user preferences.
	GL4	AwareNotifications should be able to deliver notifications to the user in specific moments, according to contextual information and user preferences.
	GL5	When a notification has to be delivered, AwareNotifications should preferably target a device that is currently used by the user.
	GL6	To choose the right instant to deliver notifications, AwareNotifications should consider natural breakpoints between user activities.
	GL7	When user does not specify preferences for certain situation, AwareNotifications should consider default rules, to resemble the behavior of the "traditional" notification system (i.e., deliver notifications instantly).
Notification classification	GL8	AwareNotifications should allow users to define a priority for each type of notifications.
	GL9	AwareNotifications should allow users to define a priority for the senders of their notifications.

concerns the delivery moment, it is demonstrated that the acceptability of a notification increases during a natural breakpoint [34], i.e., the boundary between two adjacent units of a user's activity (GL6). Postponing the notification delivery when the user finishes the activity in which she is currently involved, in fact, can lower the impact on users cognitive load caused by the interruption [28]. Finally, when there are no explicit preferences, the notification delivery should resemble the "traditional" mechanism, i.e., immediate delivery, visual representation, etc. (GL7).

3.3. Notification Classification

The literature analysis clearly shows that not all notifications have the same importance for the user [4, 5]. We identified two different types of priority for notifications. As reported in [5], the importance of a notification is correlated with the application source that generates the message. We generalized this concept by defining a priority depending on notification type, i.e., instant message, e-mail, calendar event, etc. (GL8). Sahami et al. [5] demonstrate that notifications from instant messaging apps are more important than other categories of notifications because they involve other people. Furthermore, Mehrotra et al. [11] demonstrate that the type of the sender has a significant impact on the notification acceptability. They show that a notification from a colleague is more disruptive than a notification from a family member. Thus, we identified another type of priority, correlated with the sender of a notification (GL9). These factors are the most relevant for notification classification, according to the literature, and they are much more important than other information, like notification content or the mood of the notification recipient.

4. The Semantic System Model

To reach the goal of an intelligent notification system in a multi-device and smart environment, AwareNotifications must integrate and classify information coming from heterogeneous devices and online services. For this reason, we developed the AwareNotifications ontology, and we enhanced it with a semantic reasoning mechanism. The ontology is used as the system model and it is available at http://elite.polito.it/ontologies/awarenotifications.owl.

The model offers all the advantages of the Semantic Web framework, e.g., data integration, on-the-fly composition and interoperation of Web services, etc. [35] and, with a semantic representation, it can easily answer queries such as "which devices are currently

available to display notifications?" or "what is the user current activity?", thus assisting the system in the smart notification delivery. Furthermore, semantic reasoning is able to infer logical consequences from a set of asserted facts or axioms, and it allows the discovery of hidden information. Such characteristics are extremely useful in those cases where all the information is not available. We used the semantic reasoning, for example, to classify devices on the basis of their current status, e.g., their volume level, their screen settings, etc.

The ontological model is composed of two types of concepts: a) *context* and b) *user preferences*.

Table 2 and Table 3 describes the main *context* and *user preference* concepts, respectively, while Figure 1 shows how such concepts are linked together.

For *context*, the AwareNotifications ontology uses information about notification types, notification senders, notification recipients, locations, user activities, and devices. Such contextual information pertains to three categories and they are provided by the clients, which extrapolate them from different sources:

1. Notification types, senders, and recipients are dynamic information that are taken from each received notification.
2. User locations and activities, instead, may be gathered from sensors or by using reasoning capabilities available in the IoT environment.
3. Devices and their features are static information that change slowly or not at all.

AwareNotifications does not use information about notification content, for privacy reasons and since the importance of a notification is mainly correlated with other factors, as reported in the literature [4, 5, 11].

Within the *user preferences* concept, instead, a user can define: a) the obtrusiveness level, i.e., if she is currently available to receive notifications, b) an accessibility level, i.e., how she is able to interact with her devices during an activity of a certain type, c) notification priority, i.e., the priority for all types of notifications, and d) social priority, the priority of notification sender categories.

If the user does not specify any preference, the system uses the default settings, to mimic a traditional notification system. These default settings are:

- for the notification modality, use visual representation if possible;
- deliver notifications instantly:
  - default notification priority is medium;
  - consider all notification senders as equally important.

As reported in Figure 1, user preferences and context concepts are joined. Each activity, for example, is associated with one or more accessibility level, and each notification has its priorities.

To understand the advantages of using an ontology and a semantic reasoning mechanism as the system model, we report in Figure 2 a portion of the tree that characterize the *an:NotificationsSystem* class, used for describing potential target devices for notifications. In real time, the reasoner classifies a device in different categories, such as *InputSystem*, *OutputSystem*, *HearingUsableNotificationSystem*, *SightUsableNotificationSystem*, *HandsUsableNotificationSystem*, etc, on the basis of the device features.

A *HearingUsableNotificationSystem* is a device that can be used as a target for audio notifications. In the ontology, it is defined as a device that is turned on, it owns a speaker, and its operating mode is not silent.

A *SightUsableNotificationSystem* is a device that can be used as a target for visual notifications. In the ontology, it is defined as a device that is turned on, it owns a display, and the display is switched on.

A *HandsUsableNotificationSystem*, instead, is a device that needs a user interaction to show the content of a notification. An example could be a smartphone in silent mode in the user's pocket.

5. System Architecture

The architecture of AwareNotifications is shown in Figure 3. The system is composed of a series of clients (user and IoT devices) and the AwareNotifications server. Clients are in charge of collecting contextual information (defined in Section 4) and user preferences, while the server implements the smart notification delivery.

In the remainder of this section, we first introduce the client applications and the AwareNotifications server. Then, we present the dispatching algorithm we designed to implement the smart notification delivery, along with a working example.

5.1. The AwareNotifications Clients

In AwareNotifications, client applications are a fundamental part of the system. They are responsible for collecting and updating contextual information, and

Table 2

The context model. In the Ontology Class column, we report the classes used in the ontology to model the concepts (*an* is the adopted ontology prefix).

Context sub-concept	Description	Ontology Class
Notification type	A hierarchy of notification types. Notifications are classified in user notifications (e.g., instant messaging, email, calendar events, etc.) or IoT notifications (smart appliance messages, alarms, etc.).	*an:Notification*
Notification sender	It is classified into one of these categories: friends, family members, work colleagues, and acquaintances.	*an:Person*
Notification recipient	With this information, the system can determine the user context (their location, their devices, etc.).	*an:Person*
User location	It is a hierarchy of locations, ranging from houses to single rooms, to external locations. It is useful to determine where the user is, and what devices are nearby her.	*an:Location*
User activity	A hierarchy of activity types. The top level elements of the hierarchy are physical activity, free time activity, traveling activity, work activity, and communication activity. For example, a free time activity can be specialized in reading, cooking, using the PC, etc.	*an:Activity*
Device	It is classified through a hierarchy of device types, differentiating them between user devices (e.g., smartphones, tablet, smartwatch, etc.) and IoT devices (e.g., smart appliances, connected lamps, etc.).	*an:NotificationSystem*
Device feature	A hierarchy of features that allow to determine what a device can do. For example, a device with a speaker could be used to reproduce audio.	*an:Feature*

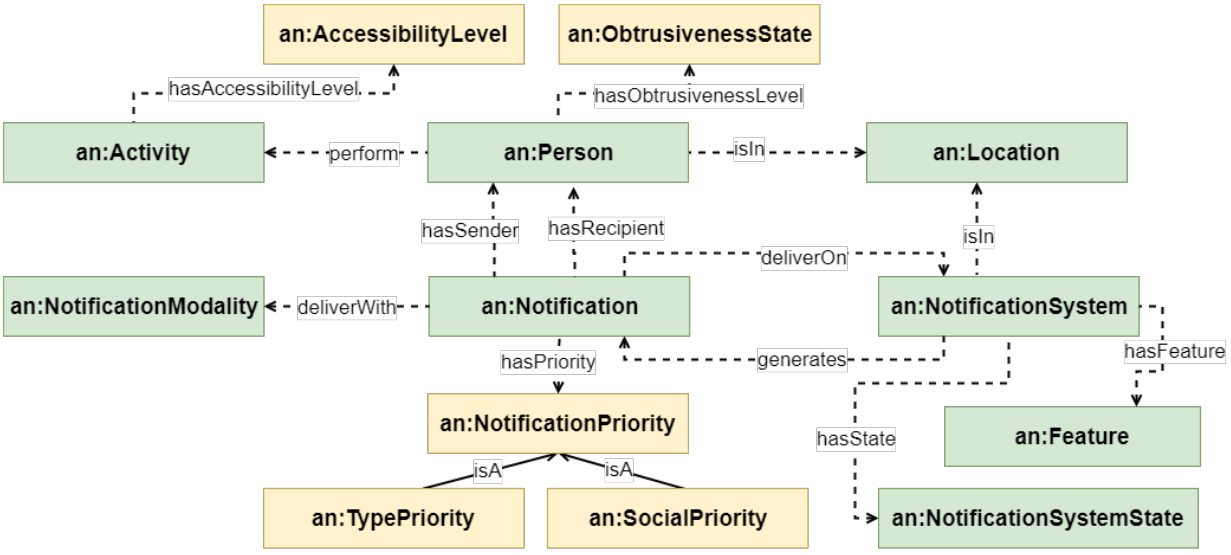


Fig. 1. The class diagram of the AwareNotifications ontology, where *an* is the ontology prefix. The classes reported in the yellow rectangles model the context concepts, while green rectangles represent user preference concepts. Classes are linked together by object properties (represented by dashed lines).

Table 3

The user definable preferences. In the Ontology Class column, we report the classes used in the ontology to model the concepts (*an* is the adopted ontology prefix).

Preference	Value	Description	Ontology Class
Accessibility level	Sight	During an activity with this accessibility level, the user is available to read notifications on nearby displays, if available.	*an:AccessibilityLevel*
	Hearing	During an activity with this accessibility level, the user is available to receive audio notifications, if there are devices able to reproduce audio.	
	Hands	During an activity with this accessibility level, the user is available to directly interact with his devices to receive notifications (e.g., take the smartphone from the pocket).	
Obtrusiveness level	Available	The user is currently available to receive notification. This is the default.	*an:ObtrusivenessLevel*
	Not available	The user is currently not available to receive notification.	
Notification priority	High priority	An urgent notification, to be delivered instantly on all turned-on devices and in the most intrusive way, without considering any other preferences.	*an:TypePriority*
	Medium priority	It is delivered instantly, but only if there are suitable devices, according to the context and the other preferences. Otherwise, the notification remains pending, waiting for a context change.	
	Low priority	It is delivered in a natural breakpoint, only.	
Social priority	Important sender	A notification from an important sender category is delivered instantly, but only if there are suitable devices, according to the context and the other preferences. Otherwise, the notification remains pending, waiting for a context change.	*an:SocialPriority*
	Not important sender	A notification of a not important sender category is delivered during a natural breakpoint, only.	

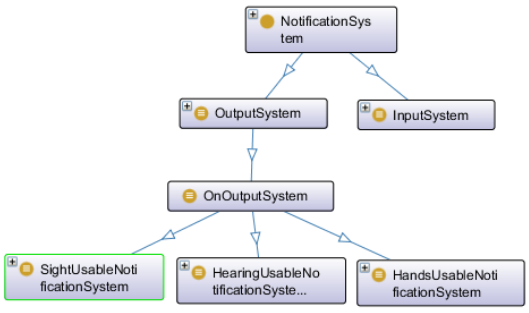


Fig. 2. A portion of the devices model in the AwareNotification ontology

to handle the final notification representation. For this purpose, each client can register three listeners, one for collecting incoming notifications, and two dedicated to retrieve contextual information:

Notification listener With the Notification listener, each "raw" notification, i.e., coming from external sources, is intercepted before appearing to the user and it is sent to the AwareNotifications server to be smartly redistributed. Furthermore, the Notification listener waits for the smartly redistributed notifications, i.e., coming from the AwareNotifications server, and display (or reproduce) them to the user according to the decided notification modalities.

Sensor listener A sensor listener is used to collect external contextual information (e.g., lighting level, noise level, user movements, etc.) and sends them

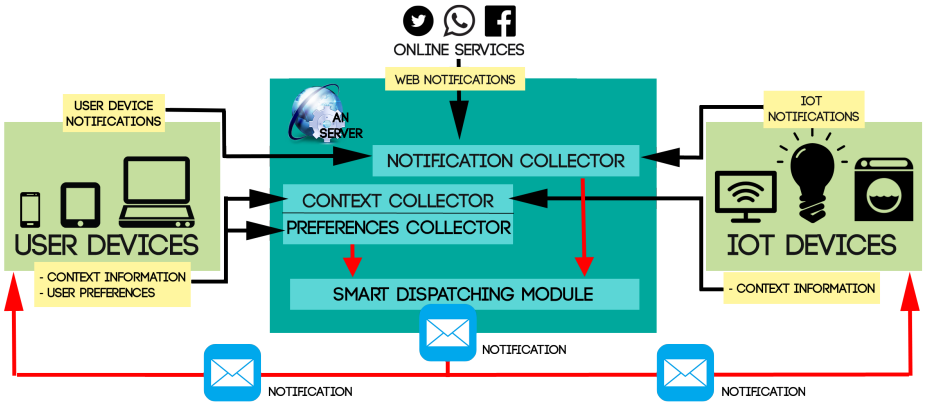


Fig. 3. System Architecture

to the AwareNotifications server. Such an information is used by the server to fill the *context* concepts of the semantic system model.

Status listener A status listener is used to communicate the status of each device in real-time to the AwareNotifications server, e.g., if the device is turned on, which volume settings is set, etc. Also this information is used by the server to fill the *context* concepts of the semantic system model. With this data, in particular, the server is able to understand whether a device is currently available for receiving notifications or not.

In addition to implementing such listeners, mobile clients (i.e., smartphones and tablets) are designed to allow users to define their own preferences (Table 3). Different mechanisms and graphical representations can be used for the definition of user preferences. One of the most promising approach, widely used for end-user personalization [36–38], is the trigger-action programming paradigm, i.e., "*if* something happens, *then* perform an action". We adapted this paradigm to define preferences associated with a context concept, i.e., accessibility level, notification priority, and social priority (Table 3). In this way, users can define rules such as "*if* I am currently working on the PC, *then* my accessibility level is Hearing."

Figure 4 shows how users can insert such rule-based preferences in our prototype implementation, described in Section 6. From an initial menu (Figure 4b), users can select the *if* clause, e.g., the activity type. Then, from another menu (Figure 4c), users define the associated *then* clause, e.g., the desired accessibility level. Users can edit such preferences or add new ones at any time, without affecting the run time operation of the AwareNotifications server.

5.2. The AwareNotifications Server

The server collects information from the clients through three collector modules. Through the *preference collector*, the server collects user preferences, while the *context collector* is used to collect contextual information coming from clients, e.g., device status changes. Furthermore, with the *notification collector*, the server intercepts all the notifications coming from the clients that need to be smartly delivered.

When a notification arrives to the *notification collector*, it is passed to the *dispatching module*, which is in charge of implementing the smart notification delivery. First, the *dispatcher module* exploits the semantic model, i.e., the AwareNotifications ontology, to reason about the contextual information and the inserted user preferences, thus taking into account the design guidelines (see Table 1). Then, for each notification, it determines:

- the target device(s);
- the modalities to show it to the user;
- the delivery moment, either immediately or at a suitable natural breakpoint.

According to the chosen device(s), modalities, and delivery moment, the module eventually dispatch the notification. The full algorithm used by the *dispatching module* is better detailed in the remainder of this section.

5.3. The Dispatching Algorithm

The dispatching algorithm implemented by the *dispatching module* can be summarized into three main steps.

5.3.1. Priority Check

When a notification arrives, the AwareNotifications server saves and classifies it by type and sender category through semantic reasoning. Then, the algorithm retrieves the user-defined preferences about notification priority and social priority, and performs the priority check:

- if the received message is a high priority notification, the system delivers it immediately, on all turned-on user devices, and in the most intrusive way;
- if the received message is a medium priority notification, or if the sender of the message is an important contact, the server proceeds with the next step of the algorithm, i.e., the conditional delivery (Section 5.3.2);
- if the received message is a low priority notification and if the sender of the message is a not important contact, the server keeps the notification pending until the next natural breakpoint detection, the third step of the algorithm (Section 5.3.3).

5.3.2. Conditional Delivery

In the conditional delivery phase, the algorithm will choose one or more target devices, and the notification modality. By analyzing the inserted accessibility and obtrusiveness level preferences, the system tries to select one or more target devices considering these sets of devices (in order of priority), retrieved from the ontology:

1. currently used device(s);
2. turned-on and nearby device(s);
3. turned-on device(s).

Target device(s) and, consequently, the notification modality, are chosen by taking into account the inserted accessibility level preferences (or applying the default settings), and the features of the involved devices. If there are no available devices, the notification is kept pending until the next natural breakpoint detection.

5.3.3. Natural Breakpoint Detection

When the system detects a natural breakpoint (i.e., when the user finishes her current activity), the system retrieves all pending notifications, and it invokes the conditional delivery for each of them. Natural breakpoints are detected thanks to the contextual information coming from the client. Through the *sensor* and *status* listeners, in fact, clients inform the server about device status and external changes, that are analyzed and stored in the AwareNotifications ontology. In this way, the server can detect simple transition between user activities, e.g., when the user unlocks his smartphone and opens the browser, or when the user turns the PC off after having checked the e-mails. Furthermore, client sensor data can be also used to recognize more complex activities, e.g., *Cooking* or *Taking a shower*, as already demonstrated in the literature, e.g., [39].

5.4. Working Example

To conclude and summarize this section, we report a working example of the system.

John is an athletic person, and he likes to go running. During his sport activities, he always brings with him his smartphone to listen to music. John also loves TV series. He owns a smart TV that he uses daily. John starts to use AwareNotifications, and he inserts the following preferences:

- *if* my activity is of type “Running”, *then* my accessibility level is “Hearing”;
- *if* my activity is of type “Watching the TV”, *then* my accessibility level is “Sight” and “Hearing”;
- *if* notification is of type “Instant Messaging”, *then* the notification priority is “Low”;
- *if* notification sender is a “Friend”, *then* the sender is “Not important”;
- *if* notification sender is a “Family Member”, *then* the sender is “Important”.

Now, suppose that John is running. If a relative of John contacts him through a chat message, AwareNotifications immediately delivers the message, by reproducing an audio notification on John’s smartphone. On the contrary, if the chat message comes from a friend of John, the system does not propagate the message, and the notification is kept pending. When John returns home and turns the TV on, the system shows the pending notification on the smart TV, with a visual representation.

6. Implementation

To validate our approach of a smart notification system, we implemented a prototype of AwareNotifications. For the purpose of this first validation, we simplified the real time management of the context on the clients application. In particular, in this first version of AwareNotifications, complex user activities, e.g., *Cooking* and *Taking a shower*, are manually provided to the web server, which uses them in the reasoning

process together with the other contextual information. All other information is collected in real time. This simplification has no impact on the evaluation reported in the next sections.

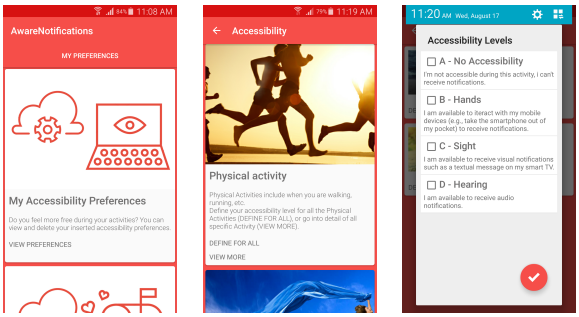
The web service has been implemented as a JavaEE web application, using the Spring framework and following the RESTful software architectural style. The ontological model exploited by the *dispatching module* for the smart delivery process has been defined with Protégé [40], an open-source OWL ontology editor and framework. Furthermore, the module uses the HermiT [41] library as a reasoner. HermiT is the first publicly-available OWL reasoner based on a novel "hypertableau" calculus which provides much more efficient reasoning than any previously-known algorithm. Given an OWL file, HermiT can determine whether or not the ontology is consistent, identify hidden relationships between classes, etc. To interact with the model, the web server uses OWL API [42], a high level Application Programming Interface (API) for working with OWL ontologies.

In the implementation, clients collects contextual information and user preferences, and send them to the collector modules of the AwareNotifications server (Figure 3) through HTTP post requests. Preferences are firstly translated into SWRL rules [43] by the *preference collector* module, and then saved in the AwareNotifications ontology. In this way, the *dispatching module* directly uses the preferences in the reasoning process. We decided to use SWRL because it allows us to easily add logic into the semantic model, to extend the expressivity of OWL with simple rules. To understand how the semantic reasoning process works, suppose that the ontology contains the following SWRL rule, here rephrased for the sake of readability:

> "*if* I am watching the TV, *then* my accessibility level is Hearing",

When the user turns the TV on, the reasoning process analyzes the devices stored in the AwareNotifications ontology, looking for devices that are turned on, that own a speaker, that are not in silent mode, and near the user. The output of the reasoner process is therefore a set of *HearingUsableNotificationSystem* devices currently available for reproducing notifications to the user.

We implemented an application for four client types: smartphone, tablet, PC, and smart TV. All the clients are in charge of showing or playing messages and listen for new notifications with a background service. For the PC, we implemented a Google Chrome extension, while for the Smart TV we used the Kodi[1] media center. For smartphones and tablets, we implemented an Android application. Figure 4 shows three screenshots of the preference-composition part of the mobile application.



(a) *The main preference menu* (b) *Activity type selection* (c) *Accessibility level values*

Fig. 4. The definition of an accessibility preference in the Android app

In AwareNotifications, notifications are managed and delivered through Firebase Cloud Messaging (FCM)[2], a Google cross-platform messaging solution for delivering messages at no cost.

7. Evaluation

We conducted a user study to compare AwareNotifications with the "traditional" management of notifications that users experience daily (i.e., unconditional immediate notification delivery). We were interested in investigating the perceived effectiveness of the novel notification delivery approach and whether AwareNotifications improves the notification delivery, in terms of notification acceptability and ease of access to the notification content.

7.1. Study Design

We performed a controlled in-lab study following a within-subject design. The test was a trial composed of five different scenarios. A scenario described a user, her current activities, and her devices. Also, the sce-

[1] https://kodi.tv/download/ (last visited on June 11, 2017)
[2] https://firebase.google.com/docs/cloud-messaging/ (last visited on June 11, 2017)

nario included a specific notification that someone or any application sent to the user in a specific instant. All the participants simulated all the five scenarios twice (with and without using AwareNotifications), by evaluating the reception of the described notification with specific questionnaires. The context of each scenario (current activities, locations, etc.) as well as the user preferences were predefined in the system. To easily change the evaluated notification system, we created a specific setting in the AwareNotifications clients to disable the interception of the "traditional" notifications. The scenarios, the questionnaires, and the conversations between participants and the moderator interventions, were in Italian. We translated the research material in English for the purpose of this paper. The five scenarios were:

1) Anna is cooking, and her hands are currently busy. Behind her, on the table, there are two personal devices, her smartphone and a tablet. Anna has forgotten her smartphone in silent mode since her morning meeting, while the volume of the tablet is active. Anna owns a smart anti-theft system that sends notifications in case of suspicious situations. Anna has installed the anti-theft app on her smartphone. When Anna is cooking, the anti-theft system detects an anomaly on the upper floor, and it generates an alarm notification.

2) Mark is driving his car. Mark has set the destination on the on-board computer, always connected to the Internet. Mark owns a personal smartphone, that is currently in his pocket. The smartphone is in stand-by, in silent mode, with the vibration modality active. During the journey, Mark receives a WhatsApp message from his mother.

3) Mary, an architect, is in her office. In this place, there are four devices connected to the Internet: three Mary's personal devices (a smartphone, a tablet, and a laptop), and a smart TV, which Mary uses during her breaks or for meetings. Currently, Mary is working on her laptop. At the same time, a friend of Mary sends her a chat message. When Mary turns off the laptop, she switches the smart TV on, having a break.

4) Paul is a computer engineer and he is currently working on his PC. Paul left his smartphone and his tablet on the table near his desk. Both the devices are turned on, with the volume on. During his working activity, his co-worker tries to contact him through a Telegram message. After a while, Paul takes a break by playing with his tablet.

5) Jack just finished a meeting. He is coming home on foot. His smarphone is in his pocket, with the volume turned on. Suddenly, Jack's secretary sends him an e-mail with the minutes of the latest meeting. At home, Jack turn his PC on to watch a movie.

7.1.1. Participants

We recruited 15 participants (five female) with a mean age of 25.4 years (SD = 3.29, range: 19-33) from a mailing list of students and personnel of our university. Participants had a mix of technical and non-technical backgrounds. With an initial questionnaire, we asked the participants to indicate their profession, the owned devices, and the estimated number of notifications received during a normal day. 11 participants were undergraduate students, 2 were researchers, and 2 were programmers. Most of the participants owned a smartphone, a tablet, and a laptop. Some participants owned a PC and a smart TV, while only few participants owned a smartwatch.

Table 4 shows the estimate number of notifications received daily by the participants. The number is quite high for most of the users: only one of them claimed to receive less than 10 notification per day.

Table 4
Number of notifications per-day received by participants of the study.

Notifications per day	**Participants**
< 10	1
10 - 50	8
50 - 100	5
> 100	1

7.1.2. Setup

To simulate the scenarios, we prepared an office with four devices connected to the Internet: a PC, a smart TV, an Android tablet (Asus Transformer Pad TF300T), and an Android smartphone (One Plus 3). All of them were equipped with the AwareNotifications client application. We hosted the AwareNotifications web server on another PC.

7.1.3. Procedure

We gave the participants a privacy module, since all sessions were video recorded, and an initial questionnaire for collecting demographic information. Then,

we explained the nature and objectives of the test. Participants started to simulate the scenarios with both notification systems, i.e., AwareNotifications and the traditional one. To avoid bias, we generically called them System A and System B. Participants were asked to impersonate the user described in each scenario by replicating the described activities (e.g., to work on a laptop, to watch TV) by using the devices in the room. The order of the scenarios and the order of the experimented notification systems were counterbalanced. In AwareNotifications, user preferences were pre-inserted according to the specific scenario. During the simulation, we sent the notification described in the scenario to the user. The notification delivery (target device(s), notifications modalities, and delivering moment) depended on the active notification system. After each simulation, the participants were asked to fill out a simulation-questionnaire of 3 questions to evaluate the notification delivering process, described in the next paragraph (*Measures*). Each user filled out 10 simulation-questionnaires in total. At the end of the test, we performed an interview with the participants, asking them the perceived advantages and disadvantages of the two systems they experimented. We gave the participants a final questionnaire with a question about the usefulness of AwareNotifications. The experiment took about 45 minutes and participants received a gadget for their participation.

7.1.4. Measures

The independent variable of the study was the notifications system used to simulate the scenarios (AwareNotifications and the traditional management of notifications).

We evaluated three dependent variables, corresponding to the three questions on the questionnarie, related to the notification delivery:

- **notification acceptability:** considering the target device(s), the notification modality, and the delivery instant, how acceptable is the notification?
- **accessibility of the notification content:** considering the target device(s), the notification modality, and the delivering instant, how much effort is required to access the notification content?
- **notification delivery:** is the notification delivery satisfactory?

We collected quantitative measures of these variables with the three questions on the simulation-questionnaires. Questions were based on a Likert-scale from 1 (Very bad) to 5 (Excellent). Before communicating to the participants the nature of the evaluated systems, we asked them to find out pros and cons of both System A and System B. Finally, after this qualitative debriefing, we collected a quantitative measure of the perceived usefulness of AwareNotifications through a final questionnaire composed of a single question based on a Likert-scale from 1 (Not useful at all) to 5 (Extremely useful).

7.2. Quantitative Results

To evaluate the three dependent variables, we conducted a one-way repeated measures ANOVA with Mauchly's sphericity test satisfied by considering the notification system as the independent variable, and we performed a post-hoc analysis with Bonferroni correction.

- **Notification acceptability.** There was a significant effect of the used system ($F(1, 14) = 81.06$, $p < .05$) on the acceptability of a notification. Therefore, the acceptability of a notification was significantly different for scenarios simulated with different notification systems. Table 5 and Figure 5a show the means for the main effect of the used system on the notification acceptability. The notification acceptability was higher with AwareNotifications than with the "traditional" management of notifications (4.35 ± 0.14 vs 2.20 ± 0.15, respectively). Post-hoc tests using the Bonferroni correction revealed that this difference was statistically significant ($p < .05$).
- **Accessibility of the notification content.** There was a significant effect of the used system ($F(1, 14) = 98.36$, $p < .05$) on the accessibility of the notification content. Therefore, the accessibility of a notification was significantly different for scenarios simulated with different notification systems. Table 6 and Figure 5b show the means for the main effect of the used system on the notification accessibility. The content of a notification was more accessible with AwareNotifications than with the "traditional" management of notifications (4.45 ± 0.13 vs 2.43 ± 0.22, respectively). Post-hoc tests using the Bonferroni correction revealed that this difference was statistically significant ($p < .05$).
- **Notification delivery.** There was a significant effect of the used system ($F(1, 14) = 143.01$, $p < .05$) on the perceived satisfaction of the notification delivery. Therefore, ratings of users on the delivery of a notification significantly dif-

fered for scenarios simulated with different notification systems. Table 7 and Figure 5c show the means for the main effect of the used system on the notification delivery satisfaction. The notification delivery was perceived more satisfying with AwareNotifications than with the "traditional" management of notifications (4.27 ± 0.12 vs 2.08 ± 0.15, respectively). Post-hoc tests using the Bonferroni correction revealed that this difference was statistically significant ($p < .05$).

Table 5
The estimated marginal means, standard errors, lower bounds (LB), and upper bounds (UB), for the system effect on the notification acceptability (95% confidence interval).

System	Mean	Std. Err.	LB	UB
Traditional	2.200	0.153	1.872	2.528
AwareNotifications	4.350	0.138	4.055	4.645

Table 6
The estimated marginal means, standard errors, lower bounds (LB), and upper bounds (UB), for the system effect on the notification accessibility (95% confidence interval)

System	Mean	Std. Err.	LB	UB
Traditional	2.433	0.216	1.970	2.897
AwareNotifications	4.450	0.134	4.162	4.738

Table 7
The estimated marginal means, standard errors, lower bounds (LB), and upper bounds (UB), for the system effect on the notification delivery satisfaction (95% confidence interval)

System	Mean	Std. Err.	LB	UB
Traditional	2.083	0.154	1.754	2.413
AwareNotifications	4.267	0.121	4.008	4.525

Finally, by analyzing the answers to the final questionnaires, we found that AwareNotifications was perceived more than useful, with a mean of 4.27 out of 5 (SD = 0.59).

7.3. Qualitative Results

We analyzed and summarized pros and cons of the two evaluated systems, as found by the participants.

For what concerns the traditional notification system, the only advantage that emerges from the participant answers is that the unconditional immediate notification delivery could be useful in some cases. However, most of the participants immediately recognized that the evaluated system was similar to what they experience daily, finding many disadvantages. In particular, they said that the system annoys the user during her activities, because notifications can be sent at inopportune moments. Furthermore, the participants noticed some missing features with respect to the other evaluated system. They said that not all notifications have the same importance, but without the priority they could not manage this possibility. Another disadvantage found for the "traditional" management of notifications is that a user has to remember to properly set up her devices, otherwise notifications may be lost.

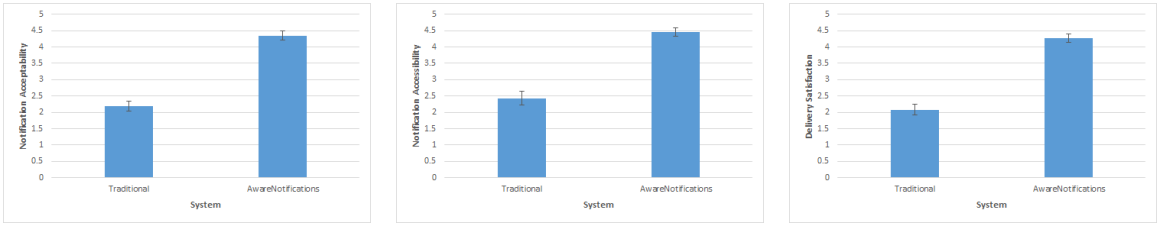
On the contrary, participants found many advantages of AwareNotifications. In many cases, these advantages naturally solved the disadvantages found for the "traditional" notification system. For example, some participants said that with AwareNotifications a user can not miss notifications, because the system intelligently notifies the user when necessary, without annoying her. Thus, the system is not disruptive, and it reduces the waste of time, because less effort is required to receive notifications. Furthermore, by giving priorities and using more than one message representation, AwareNotifications allows a user to have full control on her notifications. The possibility to manage more than one device was also appreciated.

7.4. Discussion

Thanks to the user study, we demonstrated the perceived effectiveness of AwareNotifications, comparing our system with the "traditional" management of notifications. We found that AwareNoticationsimproves the notification delivery in general (target device(s), notification modalities, and delivery instants) and the notification acceptability, the ease of access to the notification content, in particular.

Most of the participants found many disadvantages in the current management of notifications. Many of these disadvantages support what is present in the literature about the problem of the overwhelming notifications, clearly motivating the need of a smart notification system. For example, the participants said that the traditional notification system was annoying (as in [3]) and they highlighted the lack of notification priorities [4].

On the contrary, the reported advantages of AwareNotifications reveal the perceived effectiveness of the system. These benefits (e.g., multi-device environment,



(a) *Notification acceptability* (b) *Notification accessibility* (c) *Notification delivery satisfaction*

Fig. 5. Estimated marginal means for Study I.

notification priority, smart delivery) have the potential to solve many of the problems that users experience nowadays.

8. Conclusion and Future Works

This paper presents a new approach for intelligent management of multi-device notification delivery that takes into account user-defined preferences and context. To validate the approach, we designed the AwareNotifications system, grounded in well defined guidelines informed from the literature. Differently from previous works, which mainly use machine learning algorithms, we chose to explore a Semantic Web approach, by allowing users to explicitly define their own preferences. We implemented and evaluated AwareNotifications, validating it in a user study with 15 participants. Results show that AwareNotifications is perceived as useful by users, and it improves the user experience with notification delivery in terms of notification acceptability and ease of access to notifications.

Future works will include the extension of the context awareness part of AwareNotifications, e.g., by integrating previous work in this field (e.g., [44–47]). In this way, we will be able to further study our approach in-the-wild. Furthermore, we are aware that the preference definition may be not an easy task for some end-users. Do users understand the defined preferences? Are there any other better mechanisms to be exploited in client applications for the preference composition? We are exploring such questions as future works.

References

[1] Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K. Dey, and Hideyuki Tokuda. Reducing users' perceived mental effort due to interruptive notifications in multi-device mobile environments. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 475–486, New York, NY, USA, 2015. ACM.

[2] Shamsi T. Iqbal and Eric Horvitz. Notifications and awareness: A field study of alert usage and preferences. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, CSCW '10, pages 27–30, New York, NY, USA, 2010. ACM.

[3] Martin Pielot, Karen Church, and Rodrigo de Oliveira. An in-situ study of mobile phone notifications. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '14, pages 233–242, New York, NY, USA, 2014. ACM.

[4] Afra Mashhadi, Akhil Mathur, and Fahim Kawsar. The myth of subtle notifications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, UbiComp '14 Adjunct, pages 111–114, New York, NY, USA, 2014. ACM.

[5] Alireza Sahami Shirazi, Niels Henze, Tilman Dingler, Martin Pielot, Dominik Weber, and Albrecht Schmidt. Large-scale assessment of mobile notifications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 3055–3064, New York, NY, USA, 2014. ACM.

[6] Joel E. Fischer, Nick Yee, Victoria Bellotti, Nathan Good, Steve Benford, and Chris Greenhalgh. Effects of content and time of delivery on receptivity to mobile interruptions. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '10, pages 103–112, New York, NY, USA, 2010. ACM.

[7] Piotr D. Adamczyk and Brian P. Bailey. If not now, when?: The effects of interruption at different moments within task execution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 271–278, New York, NY, USA, 2004. ACM.

[8] Brian P. Bailey and Shamsi T. Iqbal. Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management. *ACM Transactions on Computer-Human Interaction*, 14(4):21:1–21:28, January 2008.

[9] Joel E. Fischer, Chris Greenhalgh, and Steve Benford. Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 181–190, New York, NY, USA, 2011. ACM.

[10] Nicky Kern and Bernt Schiele. Context-aware notification for wearable computing. In *Proceedings of the 7th IEEE Interna-*

tional Symposium on Wearable Computers, ISWC '03, pages 223–230, Washington, DC, USA, 2003. IEEE Computer Society.

[11] Abhinav Mehrotra, Veljko Pejovic, Jo Vermeulen, Robert Hendley, and Mirco Musolesi. My phone and me: Understanding people's receptivity to mobile notifications. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1021–1032, New York, NY, USA, 2016. ACM.

[12] Dominik Weber, Alexandra Voit, Philipp Kratzer, and Niels Henze. In-situ investigation of notifications in multi-device environments. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, pages 1259–1264, New York, NY, USA, 2016. ACM.

[13] Mary Czerwinski, Ed Cutrell, and Eric Horvitz. Instant messaging and interruption: Influence of task type on performance. In *OZCHI 2000 Conference Proceedings*, pages 356–361. ACM, December 2000.

[14] Brian P. Bailey and Joseph A. Konstan. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior*, 22(4):685 – 708, 2006.

[15] Mary Czerwinski, Eric Horvitz, and Susan Wilhite. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 175–182, New York, NY, USA, 2004. ACM.

[16] Gloria Mark, Daniela Gudith, and Ulrich Klocke. The cost of interrupted work: More speed and stress. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 107–110, New York, NY, USA, 2008. ACM.

[17] Stephanie Rosenthal, Anind K. Dey, and Manuela Veloso. *Using Decision-theoretic Experience Sampling to Build Personalized Mobile Phone Interruption Models*, pages 170–187. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[18] Matthias Böhmer, Christian Lander, Sven Gehring, Duncan P. Brumby, and Antonio Krüger. Interrupted by a phone call: Exploring designs for lowering the impact of call notifications for smartphone users. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 3045–3054, New York, NY, USA, 2014. ACM.

[19] Brian P. Bailey, Piotr D. Adamczyk, Tony Y. Chang, and Neil A. Chilson. A framework for specifying and monitoring user tasks. *Computers in Human Behavior*, 22(4):709 – 732, 2006.

[20] James Fogarty, Andrew J. Ko, Htet Htet Aung, Elspeth Golden, Karen P. Tang, and Scott E. Hudson. Examining task engagement in sensor-based statistical models of human interruptibility. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 331–340, New York, NY, USA, 2005. ACM.

[21] James Fogarty, Scott E. Hudson, Christopher G. Atkeson, Daniel Avrahami, Jodi Forlizzi, Sara Kiesler, Johnny C. Lee, and Jie Yang. Predicting human interruptibility with sensors. *ACM Transactions on Computer-Human Interaction*, 12(1):119–146, March 2005.

[22] Eric Horvitz and Johnson Apacible. Learning and reasoning about interruption. In *Proceedings of the 5th International Conference on Multimodal Interfaces*, ICMI '03, pages 20–27, New York, NY, USA, 2003. ACM.

[23] Daniel Avrahami and Scott E. Hudson. Responsiveness in instant messaging: Predictive models supporting inter-personal communication. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 731–740, New York, NY, USA, 2006. ACM.

[24] Joyce Ho and Stephen S. Intille. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 909–918, New York, NY, USA, 2005. ACM.

[25] Veljko Pejovic and Mirco Musolesi. Interruptme: Designing intelligent prompting mechanisms for pervasive applications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 897–908, New York, NY, USA, 2014. ACM.

[26] Benjamin Poppinga, Wilko Heuten, and Susanne Boll. Sensor-based identification of opportune moments for triggering notifications. *IEEE Pervasive Computing*, 13(1):22–29, January 2014.

[27] Martin Pielot, Tilman Dingler, Jose San Pedro, and Nuria Oliver. When attention is not scarce - detecting boredom from mobile phone usage. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 825–836, New York, NY, USA, 2015. ACM.

[28] Tadashi Okoshi, Hiroki Nozaki, Jin Nakazawa, Hideyuki Tokuda, Julian Ramos, and Anind K. Dey. Towards attention-aware adaptive notification on smart phones. *Pervasive and Mobile Computing*, 26:17 – 34, 2016.

[29] Martijn H. Vastenburg, David V. Keyson, and Huib de Ridder. Considerate home notification systems: A user study of acceptability of notifications in a living-room laboratory. *International Journal of Human-Computer Studies*, 67(9):814–826, September 2009.

[30] Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human-Computer Studies*, 67(8):639–662, August 2009.

[31] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 159–166, New York, NY, USA, 1999. ACM.

[32] Abhinav Mehrotra, Robert Hendley, and Mirco Musolesi. Prefminer: Mining user's preferences for intelligent mobile notification management. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM UbiComp'16, 2016.

[33] Dominik Weber, Sven Mayer, Alexandra Voit, Rodrigo Ventura Fierro, and Niels Henze. Design guidelines for notifications on smart tvs. In *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video*, TVX '16, pages 13–24, New York, NY, USA, 2016. ACM.

[34] Darren Newtson and Gretchen Engquist. The perceptual organization of ongoing behavior. *Journal of Experimental Social Psychology*, 12(5):436 – 450, 1976.

[35] Tim Berners-Lee, J Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):28–37, 2001.

[36] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 803–812, New York, NY, USA, 2014. ACM.

[37] Barbara Rita Barricelli and Stefano Valtolina. *End-User Development: 5th International Symposium, IS-EUD 2015, Madrid, Spain, May 26-29, 2015. Proceedings*, chapter Designing for End-User Development in the Internet of Things, pages 9–24. Springer International Publishing, Cham, Germany, 2015.

[38] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. Trigger-action programming in the wild: An analysis of 200,000 ifttt recipes. In *Proceedings of the 34rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '16, pages 3227–3231, New York, NY, USA, 2016. ACM.

[39] Gierad Laput, Yang Zhang, and Chris Harrison. Synthetic sensors: Towards general-purpose sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 3986–3999, New York, NY, USA, 2017. ACM.

[40] University of Stanford. A free, open-source ontology editor and framework for building intelligent systems, 2016. http://protege.stanford.edu/.

[41] Univesity of Oxford. Hermit owl reasoner, 2016. http://www.hermit-reasoner.com/.

[42] Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web*, 2(1):11–21, January 2011.

[43] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. Swrl: A semantic web rule language combining owl and ruleml. W3c member submission, World Wide Web Consortium, 2004.

[44] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical report, Dartmouth College, Hanover, NH, USA, 2000.

[45] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. Context-aware systems. *Expert Systems with Applications*, 36(4):8509–8522, May 2009.

[46] Charith Perera, Arkady B. Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context Aware Computing for The Internet of Things: A survey. *IEEE Communications Surveys Tutorials*, 16(1):414–454, 2014.

[47] Veljko Pejovic and Mirco Musolesi. Anticipatory mobile computing: A survey of the state of the art and research challenges. *ACM Computing Surveys*, 47(3):47:1–47:29, April 2015.