Wireless Pers Commun
DOI 10.1007/s11277-011-0387-3

A User-Centric Approach for Personalized Service Provisioning in Pervasive Environments

Anis Yazidi · Ole-Christoffer Granmo · B. John Oommen · Martin Gerdes · Frank Reichert

Abstract The vision of pervasive environments is being realized more than ever with the proliferation of services and computing resources located in our surrounding environments. Identifying those services that deserve the attention of the user is becoming an increasingly-challenging task. In this paper, we present an adaptive multi-criteria decision making mechanism for recommending relevant services to the mobile user. In this context, "*Relevance*" is determined based on a user-centric approach that combines both the reputation of the service, the user's current context, the user's profile, as well as a record of the history of recommendations. Our decision making mechanism is adaptive in the sense that it is able to cope with users' contexts that are changing and drifts in the users' interests, while it

The first author gratefully acknowledges the financial support of the *Ericsson Research*, Aachen, Germany, and the third author is grateful for the partial support provided by NSERC, the Natural Sciences and Engineering Research Council of Canada.

A. Yazidi (✉) · O.-C. Granmo
Department of ICT, University of Agder, Grimstad, Norway
e-mail: anis.yazidi@uia.no

O.-C. Granmo
e-mail: ole.granmo@uia.no

B. J. Oommen
School of Computer Science, Carleton University,
Ottawa, ON K1S 5B6, Canada
e-mail: oommen@scs.carleton.ca

B. J. Oommen · F. Reichert
University of Agder, Grimstad, Norway

F. Reichert
e-mail: frank.reichert@uia.no

M. Gerdes
Ericsson Research, Aachen, Germany
e-mail: martin.gerdes@ericsson.com

Published online: 23 August 2011

simultaneously can track the reputations of services, and suppress repetitive notifications based on the history of the recommendations. The paper also includes some brief but comprehensive results concerning the task of tracking service reputations by analyzing and comprehending Word-of-Mouth communications, as well as by suppressing repetitive notifications. We believe that our architecture presents a significant contribution towards realizing intelligent and personalized service provisioning in pervasive environments.

Keywords Pervasive computing · Unobtrusive applications · Service recommendation

1 Introduction

The environment in which we live in today is truly "pervasive". The proliferation of services and computing resources, indeed, makes the very dream of computing in such a pervasive environment realizable. However, this task has numerous real-life hurdles. Most prominent among these is the task of identifying those services that deserve the attention of the user. Ironically, as the services and tools become more pervasive, this task, in itself, is becoming increasingly-challenging due to the fact that:

1. The increasing number of services can overwhelm the attention of even the most educated user. It is, rather, plausible that an arbitrary user is *not even aware* of the services at his disposal.
2. The changes of a user's preferences and needs over time, renders the task of predicting his current services/interests extremely difficult.
3. The result of the interaction of the user with any specific service is usually uncertain. It surely depends on the performance of the latter. As low performance services can provoke his dissatisfaction, it is mandatory that an expedient system must be capable of identifying reputable (and disreputable) services [30,32].

The complexity of understanding what services could be interesting and important enough to justify disturbing the user, is the main challenge of our research. To respond to this challenge, we argue that service recommendation should rely on a multi-criteria decision maker that combines different aspects (dimensions) of the system/environment in order to decide, on behalf of the user, whether a service is relevant or not. "Relevance", we propose, should be determined based on a user-centric approach that collectively combines the reputation of the service, the user's current context, the user's profile, as well as a record of the history of recommendations. This is precisely what we have attempted to achieve in our endeavor, and we thus believe that our architecture presents a significant contribution towards realizing intelligent personalized service provisioning in pervasive environments.

A number of research studies have been interested in providing context-aware service recommendations for mobile users. Context awareness permits the system to reason about the user's current tasks, and to infer his needs in a personalized fashion. The shortcoming of these studies is that they merely rely on the user's context, combined with a *statically*-defined user's profile, in order to personalize the service recommendation. Examples of these techniques are various nomadic context-aware applications such as tourist guide applications [11] and mobile marketing and advertising applications [13].

In this paper, we argue that service recommendation is much more than simply utilizing the user's context, and we therefore present a comprehensive multi-criteria approach that goes beyond mere context awareness. Our approach resorts to Artificial Intelligence (AI) techniques to improve the quality of services recommendation over time by leveraging learning capabilities.

To clarify things, we shall present an instantiation of our architecture to a real-life, day-to-day scenario involving a proactive location-based application which provides an ensemble of services. In the scenario, the goal is to build a personalized and context-aware decision maker that delivers narrowly-targeted notifications to the user about relevant services in his environment. Nevertheless, even though this instantiation is specific, the proposed architecture is generic and can be applied to recommend a wide range of services.

Before we proceed we would like to mention that it is impossible to *comprehensively* describe the design and implementation of the entire system in a single paper. The system which we propose contains numerous modules which deal with inter-user communications, the ranking of services, inferring the dependability of other users within a social network, communication from the system to the user, discovering and recording reputations etc. Each of these modules, in itself, is a contribution in its own right. The pertinent results describing some of these modules have already been published, and the results concerning the other modules are currently being compiled. Thus, we emphasize that while this paper contains the design and implementation details of the overall architecture, we will briefly describe the functionality of some of the component modules, and omit the details which are found in the associated citations. The reader should note that a more detailed description of all of these components and the overall system will be found in the doctoral thesis of the first author [29].

The remainder of this paper is organized as follows. In Sect. 2, we describe some reported studies which are closely related to our approach. Then, in Sect. 3, we present the details of the architecture by explaining the functionality of each of the components and their mutual interactions. Section 4 reports the results of simulations conducted. These results demonstrate the efficiency of our design in reducing the unobtrusiveness that might be caused by traditional service recommendation systems. Concluding remarks and future lines of research are outlined in the conclusion.

2 Related Work

The rich availability of services in pervasive environments has the effect of over-burdening the system's service selection task. According to the vision of pervasive computing promoted by Mark Weiser, the intention of incorporating more advanced technology should be that it provides the user the possibility of operating in a *calm* frame of mind [26]. "Increasingly, the bottleneck in computing is not its disk capacity, processor speed, or communication bandwidth, but rather the limited resource of human attention" [7]. Filtering out irrelevant information has been a focal concern in a number of studies. The main issue has been to reduce the cognitive load on the user when it comes to selecting services. It is well known that "pushing" (or downloading) notifications messages to users can cause interruptions and distractions. Users who receive irrelevant notifications may become dissatisfied with their recommendation service. According to the I-centric paradigm proposed by Wireless World Research Forum (WWRF), the service provision should be tailored to the actual needs of the user [3]. The I-centric vision promotes personalization, ambient awareness and adaptability as the core requirements of future services.

A number of studies have been performed to realize this user-centric vision. A pioneering recent work was performed by Hossain et al. [9]. In this work, the authors proposed a gain-based media selection mechanism. In this regard, the gains obtained by ambient media services were estimated by combining the media's reputation, the user's context and the user's profile. As a result of such a modeling process, the service selection problem was

formulated as a gain maximization problem. Thereafter, a combination of a dynamic and a greedy approach was used to solve the problem. There are some fundamental differences between the study of [9] and the approach that we have proposed in this paper. From an architectural point of view, our work is based on a Publish/Subscribe paradigm in order to realize matchmaking between available services and the user's preferences.[1] Moreover, the authors of [9] did not present mechanisms to compute the reputation of the media services, thus, in effect, assuming that it is merely static. We argue that this assumption is not always valid, and that it is of paramount importance that the system tracks the variations in the reputation of the services since they, almost certainly, change over time.

A pertinent study that falls in the same class of our current work is the *Dynamos* project [21]. The *Dynamos* approach is an example of a context-aware mobile application that can be used for recommending relevant services to the user. In [21], the authors designed a hybrid recommender system for notifying users about relevant services in a context-aware manner. The model is based on a peer-to-peer social functionality model, where the users can generate contextual notes and ratings, and attach them to services, or to the environments. They are also permitted to share these with their peers. The attached notes to the environment are delivered to other users whenever they are in the spatial vicinity of the entities associated with the notes. A main difference between their work and what we propose is the way by which preferences are described. Their work assumed that the user was expected to explicitly describe his preferences by manually entering them. In this sense, the profile is defined by the user by explicitly specifying the types of activities and associating multiple interests to them. Such an approach can be considered to be a more "primitive" approach—it is not viable in pervasive environments where preferences change over time. Moreover, the issue of suppressing repetitive notifications was not addressed in [21].

A comprehensive study for personalized service provision has been performed by Naudet et al. from Bell Labs [17]. In [17], Naudet et al. designed an application for filtering the TV content provided to users' mobiles based on their learned profiles. The application is based on the use of ontologies to capture content descriptions as well as the users' interests. The latter interests are, in turn, mined using a dedicated profiling engine presented in [1], which leveraged Machine Learning (ML) techniques for user profiling.

The work reported in [28] presented a system that recommends vendors' web pages by measuring the similarity between the user's profile and the vendor's web page when the user is in the vicinity of the vendor (seller). The user's profile is constructed through mining the history of his web log. Another example of a mobility-aware application is *PLIGRIM* system that makes use of the user's location to recommend relevant web links [4]. In the same vein, the *SMMART* framework dynamically locates products that match the shopping preferences of the mobile user [13]. Another example is the *Mycampus* [18] product, which is a system developed and implemented by Carnegie Mellon University. *Mycampus* offers several different types of services including: context-aware recommender services, context-aware message filtering services, context-aware reminder services, applications collaboration applications, and community applications. For example, apart from the system being able to recommend nearby services such as restaurants or movies, it can remind users about things they need to purchase when they are close to a store. It can also send messages to the user when he is not busy.

[1] Adopting a "Push" based approach does not limit the applicability of our approach. In fact, the paradigm is still valid and can function in a "Pull" based manner, as in the *Dynamos* project [21].

The motivations behind our work are the following:

1. First of all, most of the reported context-aware recommendation systems do not consider the reputation of the services when issuing recommendations. In order to ensure that our hybrid recommender systems is unobtrusive, we need to locate reputable services. The success of reputation systems (such as *Ebay*) suggests that there are significant latent benefits in the convergence of these ideas in pervasive environments.
2. Secondly, in order to ensure minimal user distraction, the system should be able to track the changes in a user's interests, over time. In fact, static approaches, where the user manually defines his interest's domains, are usually not expedient as the user's needs and interests change over time. Therefore, appropriate ML techniques are needed for adapting to changing interests by *inconspicuously* monitoring service usage.
3. Thirdly, repetitively reissuing the same notification regarding the same service is usually regarded as a nuisance to the user's attention. In [30], we addressed the issue of suppressing repetitive notifications in a social mobile application. With regards to recommender systems, to the best of our knowledge, the question of suppressing repetitive notifications has not been addressed before in the literature.

Stemming from these observations, we construct a hybrid recommender system that minimizes the distraction to a user's attention while, simultaneously, maximizing the hit ratio of the service notifications. In accordance with the multiple dimensions that affect the decision making process, we have also defined a set of enabler components. The synergy between these enabler components is ensured through a Publish/Subscribe architecture.

All of these issues will be crystallized in the next section where we describe the architecture of our proposed system.

3 Architecture

The main goal of our our multi-criteria decision maker is to pro-actively notify the user about relevant services. In this section, we present the different components which articulates the architecture of our system.

3.1 Context-Dependent Service Category Activation Rules

Within our framework, the "context" includes any information that can be used to characterize the situation of a mobile user requesting a service. It could include numerous pieces of information including the user's location (where), the time of presence (when), his current activity, his "mood" etc. In our work, the description of the concept of the "context" follows the works of [5,23]. Indeed, Schmidt et al. [23] define context as: "knowledge about the user's and IT device's state, including surroundings, situation, and to a less extent, location". Dey, on the other hand, [5] defines context as: "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".

Inferring high-level context, such as the user's current activity, from low-level context data (for example, from sensor data) has been an active field of research. In this paper, we assume that such context information is available, and thus the question of inferring high-level context from raw contextual data is beyond the scope of our study. In this vein, it is worth noting that there many pieces of middleware available in the marketplace, such as *Contory*, that can

achieve the latter task. Without loss of generality, we believe that they can be reused in our architecture as a sperate component for reasoning about low-level context data [20]. For a review of pieces of context-aware middleware, we refer the reader to [12].

We should emphasize that in general, a user's interests are *context-dependent*. For example, recommendations about restaurants might be of interest to a certain user during weekends, when he is both close to the restaurant in question and when he is not busy. Also, data related to tourist attractions can be of use in the context of tourism. Such contexts can be inferred from the fact that the user is on holidays (for example, from the user's calendar) and traveling. Therefore, a viable approach is to provide the user with the ability to specify that certain kinds of services (those of interest) are active in a particular context. This is the approach that we have adopted in the current study. The idea is relatively novel and has been recently applied in the *Dynamos* framework [21]. In [21], a user is permitted to specify several types of activities and their associated status, and to associate multiple interests to each of them.

The essence of this idea has also been utilized in the *Mobilife* project, where the user has different context-dependent sub-profiles [25]. In the *Mobilife* project [25], a user's profile can be separated into different sections that include personalized information that concerns different applications. The appropriate sub-profiles are then activated as a result of changes to the context. By modeling these issues in this manner, the profile and context information are treated as meta-data that describe the user's specific instance.

With regard to specifics, in this paper, we will adopt a *two-level filtering approach* in order to support efficient matching between the available services and the user's profile. The first level of filtering is based on the user's context and is called *Context-Dependent Service Category Based Filtering*. The concepts here are akin to those found in [21], where for *each service category* (for example, restaurants, shopping, tourist attractions etc.) the user specifies the context attributes needed to make this category valid. Note that this sort of filtering is static, and can be implemented using fixed rules or stereotypes. Consequently, since the rules are static, they can be entered by the user or can be given by a template, while the names of the interests can be predefined based on a service taxonomy. From this perspective, this filtering is coarse, since we retain the *service category* such as restaurants, but do not consider refining the service recommendation by considering sub-categories of restaurants, such as Italian Pizza restaurants, Japanese restaurants, etc. In order to realize a more diversified service category matching, we integrate a wider range of pieces of contextual information, and not only location. The context attributes are mainly:

- Where: The location of the user.
- When: The time context.
- What: The activity of the user.
- What Mood: The mood of the user.

We define a function F, that statically maps a set of context attributes to a service Category as:

$$F : C_{\text{location}} \times C_{\text{time}} \times C_{\text{activity}} \times C_{\text{mood}} \mapsto \text{Service Categories}$$

An example of a Context-Dependent Service Categories Activation Rule, based on the inferred context attributes is:

F(location= *, time= weekend evening, user activity= walking, mood= *) = Restaurants

Some of the context attributes may be considered as optional, and be given a wild-card instantiation.

3.2 Learning Preferences Manager

In the previous subsection, we explained our approach to filter the available services based on their categories using Context-Dependent Service Category Activation Rules. Obviously, the category-based filtering will reduce the number of eventual services that might be of interest to the user. Nevertheless, such a filtering is coarse and needs further refinement. Therefore, we propose to carry out a second level service filtering which performs an even closer match. In this sense, the second level filtering re-filters the services *via* a finer granularity, based on the learned interests in the sub-categories. In fact, it is important that we want to model not only a *general user's interests* such as restaurants, shops, movies etc., but also the *sub-categories* of these interests that are relevant to a given user. In [31], we had presented a novel, personalized *Learning Preferences Manager* that is able to adapt to changes brought about by variations in the distribution of the user's interests, using the principles of weak estimation. This module is a fundamental component of our architecture. In this subsection, we briefly give some insights about our Learning Preferences Manager, the details of which are found in [31], and omitted here in the interest of brevity and to avoid repetition.[2]

Profile Representation: A essential element of the Learning Preference Manager is the *Profile Representation*. In [31], we adopt the Profile Representation Model advocated by [9,10]. It is important to remark that in the these publications, the latter Profile Representation Model was mainly devised for representing the user's preferences in content media. Nevertheless, the model can be easily applied to encompass a wider set of interests. The model reported in [9,10] is similar to that of [34] in the sense that it is based on <feature, weight> pairs, except that in [9,10], the authors have invoked a normalized score for the data-items. We shall first present briefly the Profile Representation Model reported in [9,10].

The user's affinity of interests in a service type, such as movies, or restaurants, is represented by a set of *attributes*. For example, for a repository of services of type movie, the set of possible attributes could be {movie genre, director name, etc.}. An attribute, in turn, possesses a set of *data-items*. For example, if the movie attribute "genre" has two data-items, namely "action" and "comedy", a vector associated with the attribute (comedy affininity = 0.7, action affininity = 0.3) reflects that the user likes comedy movies more than action movies, with a relative weighting of 0.7–0.3. The update of the weights of the data-items for a particular attribute is done in an incremental manner.

Profile Acquisition Through Relevance Feedback: In the quest to learn the user's dynamic profile, the Learning Preferences Manager is guided by so-called *Relevance Feedback* (RF) [16]. In this paper, we rely on the *Service Usage History* maintained by the authors of [9,10] as the main source of the RF. In fact, a common approach towards constructing a user's profile is through non-intrusively monitoring the history of the usage of his services. A Service Usage History (also known as the *Interaction History*), contains the history of the services used by the user over time. For example, when the user has used a certain service at a certain time instant, the Learning Preferences Manager refines and revises the user's profile based on the current instance of the usage history, which, in turn, is automatically and unobtrusively observed in the background. To obtain an index to measure this, the sum of the scores of a

[2] The paper [31] can be sent to interested readers and to the Referees, if required.

data-item for a given attribute is made to be equal to unity. To now quantify this, we have recommended the use of a Weak Estimator (devised by Oommen et al. [19]) so as to update the score of the data-item based on the usage history. Whenever, a user selects a service, the metadata describing the service is used to update the score of data-item. Thus, for example, if a user currently views a "action" movie, the scheme would increase the weight associated with the data-item, "action".

Profile Adaptation: The core function of a personalized Learning Preferences Manager is to update the user's profile in a dynamic and incremental way. This is done so that the can closely follow closely the real-time evolution of the user's interests. From this perspective, the user's interests are not constant over time, and therefore it is imperative that the system takes the profile's drift into account. In this sense, the most recent observations are more reliable, when one attempts to represent the user's *current* interests, than older ones. From a more general perspective, the task of learning the drifts in the user's interests correspond to learning evolving concepts [27].

There are several studies that have dealt with the task of learning a user's interests. These include the use of a time window [15], aging examples [14], a Gradual Forgetting function [24] etc. However, of all these, a *Sliding Window* approach is the most popular one. It consists of learning the description of the user's interests from the most recent observations, and then of discarding the observations that fall outside the window. A substantial shortcoming of the "time window" approach is the choice of the window size. In [15], the authors adopted a fixed-size time window in order to learn a user's scheduling preferences. They empirically determined that a window size of 180 was a proper choice for their particular scheduling application.

In the field of Statistical Pattern Recognition, a promising approach for estimating target distributions that change over time was recently introduced in [19]. Using these concepts, in [31], we devised a Preferences Learning Manager which takes advantage of the latter updating scheme, so as to accurately estimate the user's interest domains in non-stationary environments. The Stochastic Weak Estimator was proven to have a fast adaptation rate compared to legacy approaches such as the Sliding Window approach. Apart from the updating mechanism, philosophically, our strategy, reported in [31], is related to the approach presented in [9,10] where the authors utilized the history to update the affinity of the user's interests. We believe that this will facilitate the ease of the retrieval of personalized information, and help alleviate the user's cognitive load, needed to locate relevant information.

A Converged-Profiling Engine: In the course of their activities, users leave behind them a trail of information that can be used to model their interests. These sources of information are heterogenous: some of this information is left by the user after browsing the Internet, other pieces remain after the use of the TV or movie selection, and even through his online shopping at sites like *Amazon* or *Ebay*, for example. Therefore, we argue that the profiling engine need no longer be application dependent. Rather, it can and should support numerous applications. An interesting approach to achieve this is to exploit all these pieces of information and to integrate all these heterogonous service usages to procure a holistic picture of the user's profile. Hence, the profiling engine aggregates the usage traces coming from all these different platforms, and builds an end-user profile based on a common model. Such a comprehensive engine was proposed in [1]. We are currently considering such an enhancement for future research.

3.3 Service Reputation Manager

In this section, we introduce the Service Reputation Manager [30,32], which is a cornerstone component of our architecture. Reputation is a particularly important criterion for filtering services.

With the abundance of services available in a pervasive environment, identifying those of high quality is a crucial task. When services are pervasive, in order to maximize the usefulness of the services accessed, the user needs to build his opinion about these services in the absence of direct experience, and as a consequence, must rely on the experiences of his acquaintances. In fact, through leveraging the power of Word-of-Mouth communications, our hybrid recommender system permits us to identify reliable services possibly deserving the user's attention. This is well in-line with the view of the end-user. Indeed, in a user study carried in the *Dynamos* project [21], a user who was interviewed stated: "I don't pay much attention to movie reviewers or restaurant critics. I've been disappointed by them too many times. Instead, I ask a friend I can trust, 'Have you seen any good movies lately?'." This remark reveals much of the latent potential of a *Service Reputation Manager*, and underlines the importance of the user's social relationships in the recommendation process. The essential idea is to draw the attention of the user only to services that are recommended by his social network.

In the absence of direct experience, a user can rank the quality of the services in question by optimizing the power of Word-of-Mouth communications. By allowing users to share their feedback in the form of ratings, it is possible for users to expediently obtain knowledge about the nature, quality and drawbacks of specific services by considering the experiences of other users. Traditional reputation systems, usually compute the reputation of a service as the average of all provided ratings. This corresponds, for instance, with the percentage of positive ratings in the *eBay* feedback form [22]. Such a simplistic approach of just blindly aggregating users' experiences may mislead the reputation system if some of the user's acquaintances are misinformed/deceptive users. Misinformed/deceptive users attempt to collectively subvert the system by providing either unfair positive ratings about a service, or by unfairly submitting negative ratings. Since an alternate way to interpret unfair ratings is to consider the unreliable referrals as coming from people with different tastes, such "deceptive" agents may even submit their inaccurate ratings innocently—due to differences in tastes. Our system can easily become intrusive and ultimately become unusable if the "trust component" (or equivalently, the Service Reputation Manager) does not deal with unfair ratings of this sort. The risk of attacks from malicious users is a crucial issue that we have incorporated in our system, which is especially pertinent in a competitive marketplace.

It is reasonable to assume that the acquaintances of the user can be divided into two classes: trustworthy acquaintances that provide accurate ratings, and unreliable acquaintances that provide unfair ratings. It follows that a good reputation manager component would seek to classify the acquaintances in one of these two classes so as to counter the detrimental effect of unfair ratings. In [30,32], some of the authors of this present paper developed a Service Reputation Manager which is based on a concept analogous to collaborative filtering in order to separate between these two classes. The premise of the scheme in that paper was to separate the users' types by observing how they rate the same services. The latter scheme was designed in such a way that these users would be in the same group by maximizing the "within-group" similarities and minimizing the "between-group" similarities.

The latter problem was formulated as the Agent-Type Partitioning Problem, or *ATTP* in brief, and an algorithm to solve it was presented in [30,32]. The aim of the solution to the *ATTP* is to incrementally partition the users as being true/fair or deceptive, concurrently

with their experiences being communicated. Thus, our scheme can be divided into two interleaving phases: a Partitioning Phase and a Service Selection Phase. The inputs to the Agent Partitioning phase are the reports communicated by the other agents, while the input to the Service Selection phase is the current partitioning of the agent. The reports serve as an input to the *OMA-Based-Partitioning*, given formally in Algorithm 1 taken from [30,32].[3] Decisions about whether the service is reliable or not are the output of the overall procedure. The decision is based on intelligently combining the feedback of deceptive and fair users. As the learning proceeds, we can exclusively rely on the feedback from the fair users. Details of the algorithm and the notations used in Algorithm 1 can be found in [30,32].

Algorithm 1 Procedure OMA_Based_Partitioning

Input: $U = \{u_1, \ldots, u_N\}$ is the set of agents to be partitioned.
Output: Partitioning agents into two sub-partitions.
Method:

x_{il}: Result of the interaction between u_i and S_l
y_{il}: Reported experience
Update the vector W_l^D
for $k \leftarrow 1$ to $D-1$ **do**
  $j \leftarrow$ Index of the agent associated to record $W_l^D[k]$
  if $(y_{il} = y_{jl})$ **then**
    if (u_i and u_j are in same group) **then**
      Reward_Agreeing_Nodes(i, j)
    else
      Penalize_Agreeing_Nodes(i, j)
    end if
  else
    if (u_i and u_j are in same group) **then**
      Penalize_Disagreeing_Nodes(i, j)
    else
      Reward_Disagreeing_Nodes(i, j)
    end if
  end if
end for
End Procedure OMA_Based_Partitioning

3.4 Notification Novelty Checker

The last phase of our decision maker is a module whose task is to identify if triggering a service notification will be perceived by the user as being "repetitive" information. In [33],[4] we have argued that the user's activities can be modeled to follow some "noisy" periodic pattern, and so we can, in turn, affect the services notifications to be periodic as well. This argument will be true unless we suppress repetitive information. Consequently, any notification about a service that provides redundant information to the user can be regarded as being an unnecessary distraction. Consider the case when we represent an event as a notification attached to a particular service. If we can discern that an event is repeating (even though this repetition is non-periodic), we should attempt to suppress it. To be more specific,

[3] In the algorithm, $W_l^D(t)$ denotes: {Last D records prior to instant t relative to service S_l}. The details of the procedures Reward_Agreeing_Nodes, Penalize_Agreeing_Nodes, Penalize_Disagreeing_Nodes and Reward_Disagreeing_Nodes are found in [30,32]. They are not included here to avoid repetition, but can be included if requested by the Referees.

[4] The paper [33] can be sent to interested readers and to the Referees, if required.

let us suppose that the user visits the same location regularly. For example, let us suppose that the user visits the shop malls in the city center every weekend. One can then imagine the "nuisance" generated by a *Push* system that delivers the same service notifications on a regular basis as soon as the user approaches these malls. Prior to our work in [33], to the best of our knowledge, the problem of suppressing repetitive notifications in the context of services provisioning and recommendation systems, was both uninvestigated and unsolved.[5] However, in [33], we presented a friendly reminder application in which we demonstrated how one can harvest the benefit of event-sharing, without distracting the application user with redundant notifications.

Arguing along the same vein, in this paper, we propose that we can incorporate here the same approach that we have used for suppressing repetitions in the friendly reminder application of [33]. In [33], we introduced a new scheme for discovering and tracking noisy spatio-temporal event patterns, with the purpose of suppressing re-occurring patterns, while discerning novel events. Our scheme is based on maintaining a collection of hypotheses, each one conjecturing a specific spatio-temporal event pattern. A dedicated Learning Automaton (LA)—the *Spatio-Temporal Pattern LA* (STPLA)—is associated with each hypothesis. Whenever a user receives a service notification related to a given service, a STPLA is instantiated, and this is attached to the latter service notification in order to learn the periodicity of the context in which the service is available to the user. By processing events as they unfold, we attempt to infer the correctness of each hypothesis through a real-time guided so-called random Walk/Jump process.

To clarify issues, suppose that a notification is issued regarding a relevant service, s. If the STPLA attached to service s learns that the service is available to the user on a weekly basis, the subsequent alerts are suppressed. In other words, if the attached STPLA discovers that the user visits the location of service s on a weekly basis for example, then the alert attached to service s is suppressed because it will be, rather, perceived as a distraction. In this sense, a service notification can be useful if it arises spontaneously and unexpectedly, without being part of or being associated with a periodic pattern due to the user's activities or mobility patterns. However, if the user's mobility pattern changes and he stops his weekly visit to the the vicinity of service s, the learned pattern will be unlearned after a sufficient period of time. Therefore, there is an increased likelihood of the alert being triggered if the user revisits the same location again after that period of interruption.

In brief, the learning is divided into three parts, as illustrated in Fig. 1. Each of these sub-modules is described below.

Pattern Evaluation: In the Pattern Evaluation part, the goal of the LA is to discover the presence of the spatio-temporal event pattern associated with the maintained hypothesis. In this phase, the state transitions illustrated in the figure are such that any deviance from the hypothesized pattern, modeled as a Penalty (P), causes a jump back to State '1'. Conversely, only a systematic presence of the pattern hypothesized, modeled as a pure sequence of Rewards (R), will allow the LA to pass into the Pattern Acceptance part.

Pattern Acceptance: In the Pattern Acceptance part, consisting of state $N_1 + 1$, we affirm that the hypothesized pattern has been confirmed with a fairly high probability.

Pattern Tracking: In the Pattern Tracking part, consisting of states $\{N_1 + 2, \ldots, N_1 + N_2 + 1\}$, the goal is to detect when the discovered pattern disappears, without getting distracted by omission errors. Thus, this part is the "opposite" of the Pattern Evaluation part in the sense that a pure sequence of Penalties is required to "throw" the LA back into the Pattern

[5] Prior to our work in [33], the state-of-the-art was a simplistic approach proposed by the authors of [8], where a notification was quite simply suppressed if it had been triggered in the past.

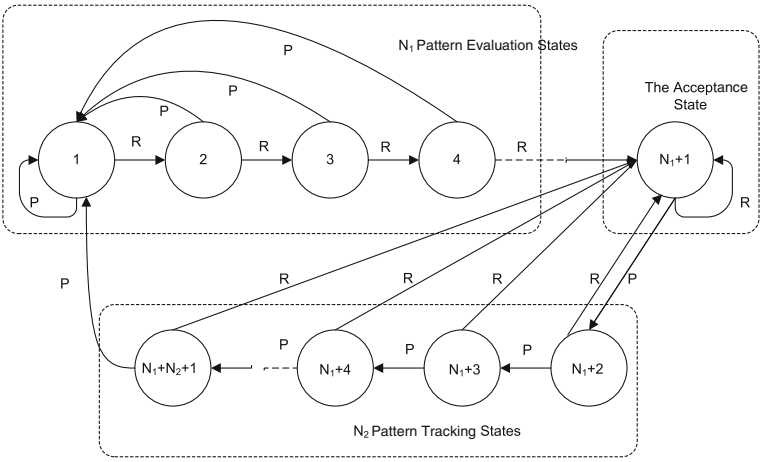

Fig. 1 The schematic state transition map of the spatio-temporal pattern LA

Evaluation part again, while a single Reward reconfirms the pattern, forcing the LA to return to the Pattern Acceptance part of the state space.

Our STPLA presents a simple learning-based mechanism for suppressing repetitive notifications. Nevertheless, we believe that our approach presents a first step in the right direction due to the scarcity of the related literature. Understanding what seems "repetitive" and "unobtrusive" to the user from a human and social perspective requires a more profound and complicated analysis involving human psychology and cognition. Thus, we believe that a more advanced learning scheme must ideally also consider issues from the perspective of human psychology and cognition, which is beyond the scope of the present paper.

Another remark concerns the choice of the internal memory parameters of the STPLA, i.e., N_1 and N_2. We advocate that N_1 and N_2 be set according to the specific user's preferences. Thus, for example, some users may desire to receive a few notifications before they are suppressed. In such cases, we can choose to increase N_1. On the other hand, others may want to increase N_2 so as to suppress notifications for a longer period of time.

3.5 Service Notification Based on a Publish/Subcribe Paradigm

Now that the individual modules have been explained, we state that the overall architecture of or system would be as described pictorially in in Fig. 2.

Our requirement of offering highly pertinent information through a push-based approach to the user can be well supported through the Publish/Subscribe paradigm [2]. The system can be deployed using a Publish/Subscribe System, which puts all the pieces of this puzzle together. A Publish/Subcribe model consists of information providers, who publish events to the system, and of information consumers, who subscribe to events of interest within the system. A Publish/Subscribe architecture ensures the timely notification of events to the interested subscribers. Note too that the use of a Publish/Subcrsibe server will enhance privacy, since no user-sensitive private information need be transmitted to the service providers. It is worth mentioning that the case where the services are not published in the registry of the Publish/Subscribe System (therefore the user can not subscribe to them) is a research problem in itself which we are currently investigating.

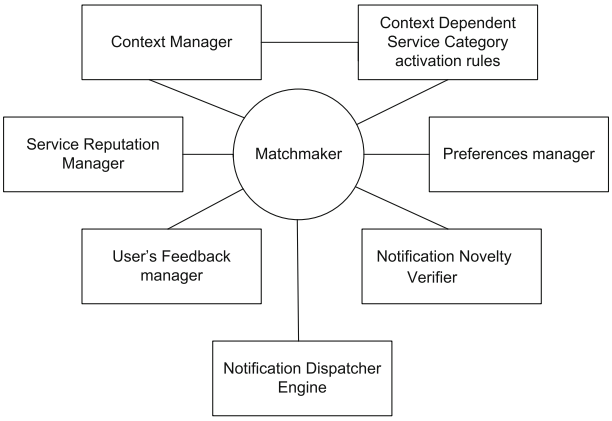

Fig. 2 The high-level architecture of our system

In our solution, we deploy a user-centric approach in order to decide wether to trigger a notification regarding a service. First of all, a user must subscribe to receiving notifications. The subscription, in turn, contains a static and a dynamic part. The static part is defined by the user, and relies on the Context Based Service Category Filtering explained in Sect. 3.1. The dynamic part is affected by the continuously changing user's interests, the versatility of the reputation of the services, and the novelty of the notifications. The "Matchmaker" achieves its task by utilizing different types of matching phenomena, such as category-based matching, location-based matching etc. A notification is issued whenever the service matches the user's subscription. In other words, this occurs whenever the following conditions should be met:

- A spatial filter reports that the service is in the user's vicinity. We agree that in the case of location-based services, the knowledge of the user's context is the most differentiating information within this context.
- The Service Reputation module returns the truth value of whether the service is reputable, as per the approach defined in [30,32], and briefly explained in Sect. 3.3.
- The service description matches the user's profile according to the above-mentioned two-level filtering approach. To identify service items of interest, the matching process consists of two steps. First, for each service, its associated category is matched with the set of active service categories. These service categories, generally, specify the business branches of the service (e.g., Restaurants, Shops). After applying the context-dependent category activation rules, only the services belonging to the active categories are maintained. Moreover, the service sub-category should match the second filter characterized by a finer granularity, namely the Learning Preferences Manager.
- The Novelty Detection module reports that the eventual service notification would not be repetitive by checking wether the notification is a part of a spatio-temporal pattern.

We propose that an Event Server, analogous to the one described in [6], allow the different modules to notify the core of the system if relevant changes occur. Based on the result of the above-mentioned Matchmaker, the Notification Dispatcher Engine will be triggered to deliver notifications to the user. As stated in [6], Metadata listeners are essential to the proper functioning of our Publish/Subscribe paradigm. In fact, if a change occurs that might eventually affect the result of the matchmaking procedure, the system would re-invoke the

Matchmaker. For example, if a notification reports changes to the context of active users, the system mandates that the Matchmaker be re-invoked so as to verify whether new matched services have to be notified.

In this direction, we need the following Metadata Listener:

- We need a Metadata Listener attached to the Context Manager module. Whenever the Context Manager reports a change in a context attribute that might affect the Context-Dependent Service Category Activation rules, or a change in the user's location, this Listener invokes the Matchmaker.
- We need a Metadata Listener attached to the the Service Reputation module. Whenever a service reputation undergoes a change as a result of a new submitted rating, the Listener invokes the Matchmaker.
- We need a Metadata Listener attached to the Preferences Manager module. Thus, whenever an update in the weights of the interests of the services is performed due to a *Relevance Feedback*, this Listener invokes the Matchmaker.
- We need a Metadata Listener attached to the Context-Dependent Service Category Activation Rules. Whenever the Context-Dependent Service Category Activation Rules are manually changed, there is, again, a need to re-evaluate the service match.

3.6 Improvement to Current Practice: Collecting Advertisements for Later Use

In this section, we conclude these design considerations, by providing an enhancement to the architecture so as to be able to provide offers for future contexts. In fact, in the previous section, we argued that the notification should be trigged instantaneously as a result of matchmaking procedures. However, suppose that a service category is not valid in the user's current context. In that case, all the surrounding services belonging to that service category will be ignored by the Matchmaker, and the user will not be aware of them in his current context. However, in order to provide more flexibility, we argue that some of the advertisements concerning services should still be collected and stored in a Registry for later use, even though they do not match the user's current context.

4 Experimental Results

As the reader can easily comprehend, it is impossible to completely test the entire system that we have proposed since it involves numerous subjective and real-life components. However, to demonstrate the proof of these concepts, in this section, we present results of simulations that we have conducted, that puts into a nutshell all the components of the proposed architecture. To do this, we have adopted a Discrete Event Simulation methodology. The performance metric to assess our architecture is the hit ratio, denoted $\alpha(t_n)$, and defined at any given time instance, t_n, as the ratio of the relevant notifications delivered to the user at time t_n. We define a relevant (or equivalently, non-distractive) notification as one where:

- The service matches the user's profile
- The notification is not repetitive
- The user's interaction with the service leads to the user's satisfaction.

By virtue of the above, we consequently regard a distractive notification as one that is either repetitive, or if the interaction with the service does not lead to the user's satisfaction

due to its low performance value [30,32], or if the recommended service does not match the current user's interests.

In the same vein, we define the "Distraction" ratio (denoted $\beta(t_n)$), at any given time instance t_n, as the ratio of distractive notifications delivered to the user at time t_n. Clearly $\alpha(t_n) + \beta(t_n) = 1$.

Unguided Recommendation System: To demonstrate the power of our architecture, we compare our approach to an *Unguided Recommendation System*, that delivers to the user notifications regarding services that match only his contextual preferences based on the Context Based Service Category static filtering. In other words, we suppose that the *Unguided Recommendation System* performs only coarse contextual filtering. For example, in the case of the notification of location-based services, the *Unguided Recommendation System* sends restaurant suggestions every time the user is close to a restaurant without learning his profile, without suppressing repetitive alerts and without checking the reputation of the service. In our simulation, we considered delivering only a single notification per location.[6]

Modeling the Spatio-Temporal Pattern: We assume that the user's mobility follows a given noisy periodic spatio-temporal pattern. As explained previously in Sect. 3.4, the location and time primitives are combined from their cross-product spaces to produce spatio-temporal patterns. Let us suppose that the mobile user in question, u, visits a given location R according to a weekly spatio-temporal pattern characterized by an omission noise $q = 0.1$. We suppose that a pool of services S is available in the visited area, for eventual access by the user.

Two Time Scales: At this juncture, it is important to remind the reader that, for the sake of clarity, we use two time granularities (or two time scales) for different events in our Discrete Event Simulation model. In fact, at the granularity of a week, namely at time instances t_n (n denotes the week index), the user visits the location R, and therefore, it is likely that service notifications can take place. On the other hand, at the lower time scale (or equivalently, at the finer time granularity) of a day, we assume that other possible events can take place, such as the generation of Relevance Feedback that serves as input to the Learning Preferences Manager, or the submission of a service rating by user in $\mathcal{U}$ that serves as input to the Service Reputation Manager.

Comparison Approach: In order to assess the efficiency of our approach, we compare, through simulation results, the hit ratio of our design with the hit ratio of the Unguided Recommendation System, and show that we can reduce the perceived distraction and unobtrusive application behavior by resorting to AI-based learning mechanisms. However, it is worth noting that our design can still lead to distractive notifications if the "learning process" fails to cope with the continuously-changing user's interests, or the trust component does not filter low performance services due to the presence of deceptive agents that subvert the Service Reputation Manager. In order to present the entire picture clearly, we specify how the components of our architecture are integrated together in the artificially-created Discrete Event Simulation environment.

[6] It is possible to adopt a top-N recommendation approach in order to not overwhelm the user with a long list of services and thus limit the size of the list.

Modeling Experience Sharing: We further assume that the mobile user u possesses a set of acquaintances $\mathcal{U}$ that communicate their experiences regarding the performance of the available pool of services, S. We assume that at discrete time instances, the acquaintances in $\mathcal{U}$ communicate their ratings to u. In the absence of direct experience from the user, the feedback provided by the acquaintances serves as input to the Service Reputation Manager, referred to in Sect. 3.3. Intuitively, the ability of the Service Reputation Manager to identify high reputation services depends on the quantity of feedback submitted by the user's acquaintances. In fact, if the feedback is scarce, the Service Reputation Manager would probably fail to identify and recommend high performance services to the user.

Modeling Preferences: For the sake of clarity and simplicity, we assume that the user preferences fall into two categories C_1 and C_2. We further assume that the underlying distribution of the weights of the preferences that reflect the affinity of user's interest in each of the preferences categories C_1 and C_2 follows a binomial distribution [31]. Therefore, the problem of estimating the user's interests in this particular case is modeled as the estimation of the parameters for binomial random variables. The Relevance Feedback concerning the preferences categories C_1 and C_2 is generated according to the true underlying value of s_1 and s_2. The intention of the Learning Preferences Manager is to estimate S, i.e., s_i for $i = 1, 2$. We achieve this by maintaining a running estimate $P(n) = [p_1(n), p_2(n)]^T$ of S, where $p_i(n)$ is the estimate of s_i at time granularity 'n', where n denotes the day index. Note that we assume that the Relevance Feedback is available at the finer time granularity of a day.

If $s_i > s_j$, we say that category C_i represents the user's preferred interest category, and thus assume that only services that belong to category C_i are of interest to the user. All the services belonging to category C_j will not be of interest to the user, and notifying him about these services will result in a distraction. Consequently, the matchmaking of the preferences will rely on the same simple mechanism, and recommend the services whose estimated category weight is larger between the two categories. The reader should observe that this simple rule is similar to decision rules in classifiers, where the decision maker has to decide on a hypothesis on the state of nature between two exclusive hypotheses. However, a more sophisticated preferences matchmaking approach, analogous to the one in [9,10] that is based on assessing a linear combination of the weights, can be easily adopted in combination with our Learning Preferences Manager [31].

An important parameter that must be specified is the rate at which the Relevance Feedback occurs. We suppose that at the finer time granularity of a day, a Relevance Feedback is generated according the underlying distribution, S. Therefore, the estimated weights of C_1 and C_2 are tracked and updated at the granularity of a day.

Classifying the Services: We further model the performances of services as either being High Performance or Low Performance as reported in [30,32]. We also assume that the services either belong to C_1 or C_2. Therefore, we will have a combination of 4 exclusive classes of services in the current experiments:

- 25 High performance services that belong to C_1
- 25 High performance services that belong to C_2
- 25 Low performance services that belong to C_1
- 25 Low performance services that belong to C_2.

If, for example, C_1 represents the current preferred interest category, the Recommendation System will recommend services to the user that are both of high performance, and that belong to category C_1. In the simulation settings, we assume that the user possesses

40 acquaintances in his social network—twenty of which are deceptive and the remaining 20 are trustworthy [30,32]. Furthermore, the trustworthy user's acquaintances are characterized with $p = 0.8$, while the deceptive ones have $p = 0.2$. In all the experiments, we configure the STPLA with $N_1 = 5$ and $N_2 = 5$. The high performance services have an performance probability of 0.8, while the low performance services have are characterized by the performance probability of 0.2 [30,32].

As alluded to previously, we suppose that the user's mobility follows a weekly periodic noisy pattern, and thus we conducted the simulations for a period of 40 week instances.

We report now the results obtained by testing our proposed architecture in a variety of settings.[7]

4.1 The Case of Static Preferences

In this experiment, we compared the distraction ratio as well as the hit ratio obtained by our approach with the respective ratios obtained by utilizing an Unguided Recommendation System. The results were obtained from an ensemble of 100 simulations, and we report $\alpha(t_n)$, where n denotes the week index. In Fig. 3a, we report the hit ratio, and in Fig. 3b, we report the distraction ratio. The preferences were assumed static, and thus, in other words, we employed the same underlying distribution for the weights of the preferences. We also assumed that the current preferred services category was C_1. We supposed that at a finer time granularity, namely, at a daily basis, each of the acquaintances submitted a rating for a randomly chosen service among the pool of available services. We observe from Fig. 3b that the distraction ratio asymptotically approaches the value 0.2 and that the hit ratio approaches the value 0.8. These values can be explained by the fact that our architecture tends to recommend only the 25 High performance services that belong to C_1 as time advances.

Furthermore, we remark from Fig. 3a and its counterpart Fig. 3b that the performances achieved by utilizing our proposed architecture improves almost uniformly over time, and that it outperforms the Unguided Recommendation System.

The results for another set of experiments are shown in Fig. 4a and b. In these experiments, we changed the settings by assuming that the high performance services had an performance probability of 0.7, while the low performance services were characterized by the performance probability of 0.3. Whereas in Fig. 4a, we report the hit ratio, in Fig. 4b, we report the distraction ratio. As in the case of the previous figures, the convergence of the graphs to their optimal levels is clear from these figures too.

4.2 The Case of Changing Preferences

In the next set of experiments, we assumed that at a finer time granularity, for example, at a daily basis, each of the acquaintances submitted a rating for a randomly-chosen service among the pool of available services. Clearly, as in the previous experiment, the feedback was provided at a lower time scale when compared the accesses and notifications of the service.

At week 20, we performed an environment "switch" by artificially modifying the distribution of the preferences, so that $s_2 > s_1$, and consequently C_2 became the current user's preferred service category instead of C_1. In Fig. 5, we report the distraction ratio obtained via utilizing our architecture for an internal parameter $\lambda = 0.8$ of the Weak Estimator. The results

[7] We have done experiments for numerous settings and scenarios. In the interest of brevity, we merely report a few representative (and typical) experimental results, so that the power of our proposed methodology can be justified.

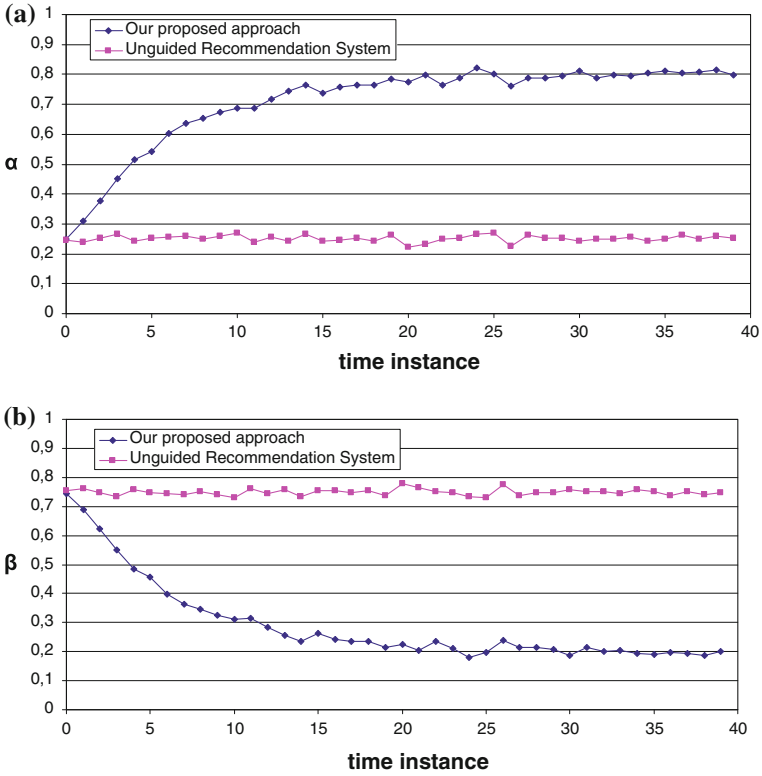

Fig. 3 **a** The evolution of the hit ratio in the case of our proposed approach and the Unguided Recommendation System, when the performance probabilities of the high and low performance services are 0.8 and 0.2 respectively. **b** The evolution of the distraction ratio in the case of our proposed approach and the Unguided Recommendation System for the same settings

were again obtained from an ensemble of 100 simulations, in which we report $\alpha(t_n)$, where n denotes the week index. The distraction ratio increases to unity after the switch at time instance 20 because all the notifications concerned the category of services that the user is not interested in, namely C_1. The Learning Preferences Manager spends some time instances before it unlearns and therefore reduces the estimated weight of C_1 so that C_2 becomes the current interest.

Figure 6 displays the plot of the corresponding distraction ratio obtained by our architecture for the parameter $\lambda = 0.9$ used to update the Weak Estimators. We observe that in this case, more time instances are required for unlearning than in the previous case when $\lambda = 0.8$. This observation is typical, because, in fact, the adaptivity of the estimator, and consequently of our Learning Preferences Manager, is dependent on the internal parameter λ, which controls the rate of convergence but not the final terminal value.

4.3 Effect of Varying the Experiences Submission Rate

In this experiment, we studied the effect of varying the rate at which the acquaintances submitted their experiences to the Service Reputation Manager. In Fig. 7, we depict the distraction ratio for two cases, namely the ones in which the reports were submitted at a "normal

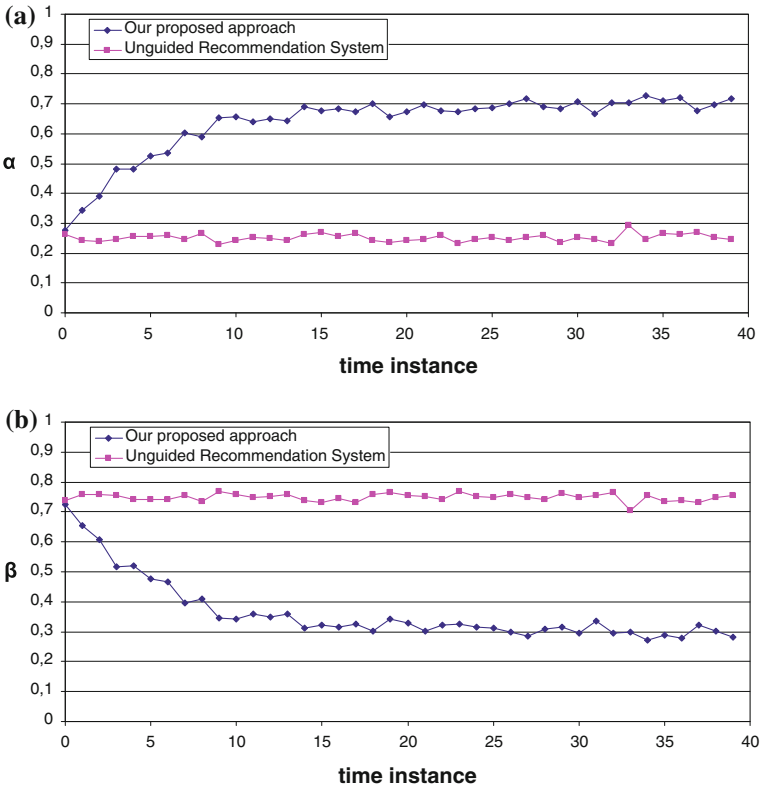

Fig. 4 **a** The evolution of the hit ratio in the case of our proposed approach and the Unguided Recommendation System, when the performance probabilities of the high and low performance services are 0.7 and 0.3 respectively. **b** The evolution of the distraction ratio in the case of our proposed approach and the Unguided Recommendation System for the same settings

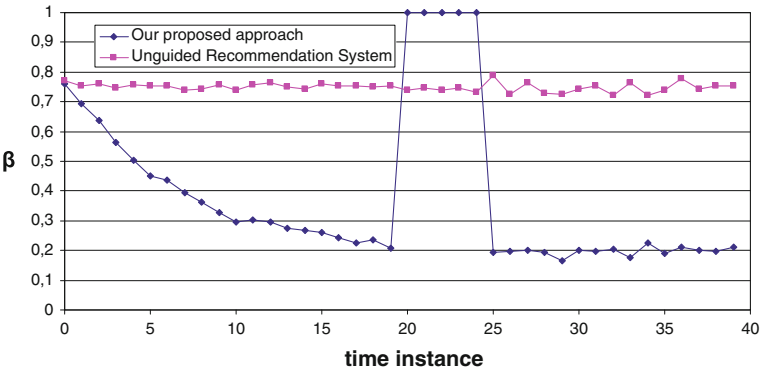

Fig. 5 The distraction ratio under varying preferences when $\lambda = 0.8$

submission" rate and the reports were submitted at a "lower rate". In this context, by "normal submission rate" we mean the case where, at the finer time granularity of a daily basis, each of the acquaintances submitted a rating for a randomly-chosen service among the pool of

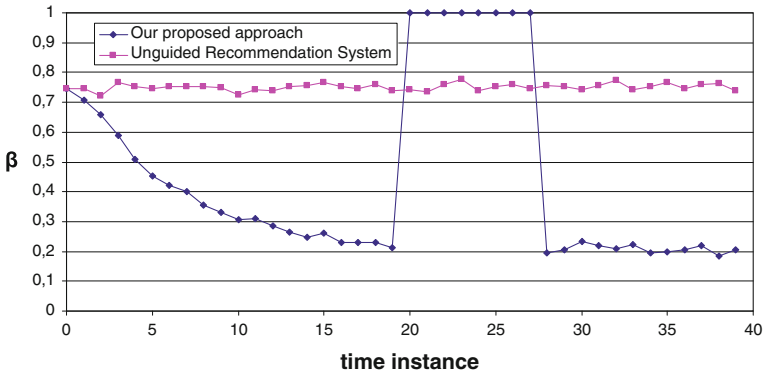

Fig. 6 The distraction ratio under varying preferences where $\lambda = 0.9$

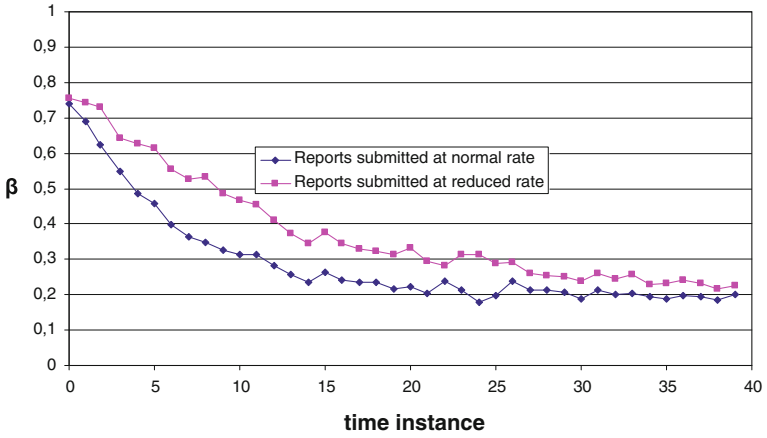

Fig. 7 The plot of the distraction ratio by varying the submission rate of the reports. In this case, the probability of an acquaintance of the user submitting a rating is 0.5

available services. Similarly, for a "lower submission rate" we mean the case where, at the finer time granularity of a daily basis, each of the acquaintances, submitted a rating with probability 0.5, for a randomly-chosen service among the pool of available services. Therefore, this so-called "lower submission rate" is, on the average, half of the "normal submission rate". By reducing the rate of submitting reports, our Recommendation System is more prone to recommending low performance services due to the more scarce feedback received from the acquaintances. In other words, in the presence of scarce feedback, the Service Reputation Manager requires more time to learn to differentiate between high performance services and low performance services. As in the above, the results were obtained from an ensemble of 100 simulations. In Fig. 7 we report the distraction ratio $\alpha(t_n)$, where n denotes the week's index.

Figure 8 displays the analogous results of Fig. 7 in which the probability of an acquaintance of the user submitting a rating is 0.25. As in the previous case, the results converge. But interestingly enough, the convergence to the optimal distraction ratio is slower—by virtue of the scarcer feedback. This result is clearly intuitively satisfying!

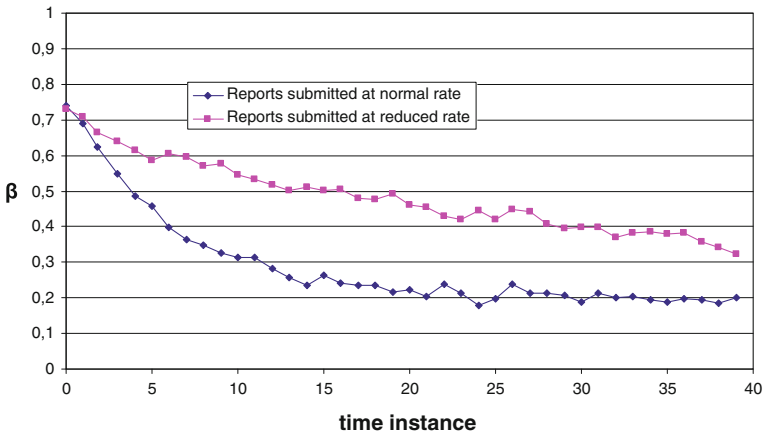

Fig. 8 The plot of the distraction ratio by varying the submission rate of the reports. In this case, the probability of an acquaintance of the user submitting a rating is 0.25

5 Conclusion

In this paper we have considered the problem of computing in pervasive environments, and in particular, in identifying those services that deserve the attention of the user. We have presented an adaptive multi-criteria decision making mechanism for recommending relevant services to the mobile user, where "*Relevance*" is determined based on a user-centric approach that combines both the reputation of the service, the user's current context, the user's profile, as well as a record of the history of recommendations. We have proposed the architecture of a system that builds a personalized and context-aware application that delivers narrowly targeted information to the user, while being unobtrusive. The design avoids flooding the user with irrelevant information. The architecture is based on a "Push"-based paradigm, where the user is spared from continuously seeking for interesting services. However, it can be easily adapted to a "Pull"-based approach, by incorporating the same principles of filtering. We have also conducted simulations and reported results that suggest that our architecture can significantly reduce unobtrusiveness. To gain more insights into the acceptance of the system by an end user, in the future, we propose that the system be deployed into a real-life application domain, which also incorporates a user study.

References

1. Aghasaryan, A., Betgé-Brezetz, S., Senot, C., & Toms, Y. (2008). A profiling engine for converged service delivery platforms. *Bell Labs Technical Journal, 13*(2), 93–103.
2. Aguilera, M. K., Strom, R. E., Sturman, D. C., Astley, M., & Chandra, T. D. (1999). Matching events in a content-based subscription system. In *PODC '99: Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing ACM, New York, NY, USA* (pp. 53–61). doi:10.1145/301308.301326.
3. Arbanowski, S., Ballon, P., David, K., Droegehorn, O., Eertink, H., Kellerer, W., et al. (2004). I-centric communications: Personalization, ambient awareness, and adaptability for future mobile services. *IEEE Communications Magazine*, 42(9), 63–69.
4. Brunato, M., & Battiti, R. (2003). Pilgrim: A location broker and mobility-aware recommendation system. In *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications* (p. 265). Washington, DC, USA: IEEE Computer Society.

5. Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing, 5*(1), 4–7. doi:10.1007/s007790170019.
6. Eugster, P. T., Felber, P. A., Guerraoui, R., & Kermarrec, A. M. (2003). The many faces of publish/subscribe. *ACM Computing Surveys, 35*, 114–131.
7. Garlan, D., Siewiorek, D., Smailagic, A., & Steenkiste, P. (2002). Project aura: Toward distraction-free pervasive computing. *IEEE Pervasive Computing, 1*(2), 22–31. doi:10.1109/MPRV.2002.1012334.
8. Hinze, A., & Voisard, A. (2003). Location- and time-based information delivery in tourism. In *Proceedings of 8th International Symposium in Spatial and Temporal Databases (SSTD)* (pp. 489–507). Springer.
9. Hossain, M. A., Atrey, P. K., & El Saddik, A. (2008). Gain-based selection of ambient media services in pervasive environments. *Mobile Networks and Applications, 13*(6), 599–613.
10. Hossain, M. A., Parra, J., Atrey, P. K., & El Saddik, A. (2009). A framework for human-centered provisioning of ambient media services. *Multimedia Tools and Applications, 44*, 407–431.
11. Kaasinen, E. (2003). User needs for location-aware mobile services. *Personal Ubiquitous Computing, 7*(1), 70–79. doi:10.1007/s00779-002-0214-7.
12. Khedo, K. K. (2006). Context-aware systems for mobile and ubiquitous networks. In *ICNICONSMCL '06: Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies* (p. 123). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ICNICONSMCL.2006.68.
13. Kurkovsky, S., & Harihar, K. (2006). Using ubiquitous computing in interactive mobile marketing. *Personal Ubiquitous Computing, 10*(4), 227–240. doi:10.1007/s00779-005-0044-5.
14. Maloof, M. A., & Michalski, R. S. (2000). Selecting examples for partial memory learning. *Machine Learning, 41*, 27–52.
15. Mitchell, T. M., Caruana, R., Freitag, D., McDermott, J., & Zabowski, D. (1994). Experience with a learning personal assistant. *Communications of the ACM, 37*(7), 80–91. doi:10.1145/176789.176798.
16. Montaner, M., Lopez, B., & de la Rosa, J. L. (2003). A taxonomy of recommender agents on the internet. *Artificial Intelligence Review, 19*, 285–330.
17. Naudet, Y., Aghasaryanb, A., Mignon, S., Toms, Y., & Senot, C. (2010). Ontology-based profiling and recommendations for mobile tv. In M. Wallace, I. Anagnostopoulos, P. Mylonas, & M. Bielikova (Eds.), *Semantics in adaptive and personalized services, studies in computational intelligence, Vol 279* (pp. 23–48). Berlin/Heidelberg: Springer.
18. Norman, S., Fabien, G., & Kwon, O. B. (2006). Ambient intelligence and pervasive computing, chap. Ambient Intelligence: The MyCampus Experience. ArTech House.
19. Oommen, B. J., & Rueda, L. (2006). Stochastic learning-based weak estimation of multinomial random variables and its applications to pattern recognition in non-stationary environments. *Pattern Recognition, 39*(3), 328–341. doi:10.1016/j.patcog.2005.09.007.
20. Riva, O. (2006). Contory: A middleware for the provisioning of context information on smart phones. In *Middleware '06: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, (pp. 219–239). New York, Inc., New York, NY, USA: Springer.
21. Riva, O., & Toivonen, S. (2007). The dynamos approach to support context-aware service provisioning in mobile environments. *Journal of Systems and Software, 80*(12), 1956–1972. doi:10.1016/j.jss.2007.03.009.
22. Schlosser, A., Voss, M., & BrÄuckner, L. (2005). On the simulation of global reputation systems. *Journal of Artificial Societies and Social Simulation 9(1)*, 4.
23. Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Laerhoven, K.V., & Velde, W. V. D. (1999). Advanced interaction in context. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing* (pp. 89–101). London, UK: Springer.
24. Schwab, I., Kobsa, A., & Koychev, I. (2001). Learning user interests through positive examples using content analysis and collaborative filtering. In *30 2001. Internal Memo, GMD.*
25. Sutterer, M., Droegehorn, O., & David, K. (2007). User profile management on service platforms for ubiquitous computing environments. In *VTC Spring* (pp. 287–291).
26. Weiser, M. (1991). The computer for the twenty-first century. *Scientific American, 265*(3), 94–104.
27. Widmer, G. (1997). Tracking context changes through meta-learning. *Machine Learning, 27*(3), 259–286. doi:10.1023/A:1007365809034.
28. Yang, W. S., Cheng, H. C., & Dia, J. B. (2008). A location-aware recommender system for mobile shopping environments. *Expert Systems with Applications, 34*(1), 437–445. doi:10.1016/j.eswa.2006.09.033.
29. Yazidi, A. (2011). Intelligent learning automata-based strategies applied to personalized service provisioning in pervasive environments. Ph.D. thesis, Department of ICT, University of Agder, Grimstad, Norway.

30. Yazidi, A., Granmo, O. C., Lin, M., Wen, X., Oommen, B. J., Gerdes, M., et al. (2010). Learning automaton based on-line discovery and tracking of spatio-temporal event patterns. In B. T. Zhang & M. Orgun (Eds.), *PRICAI 2010: Trends in artificial intelligence, Lecture notes in computer science,* Vol. 6230 (pp 327–338). Berlin/Heidelberg: Springer.
31. Yazidi, A., Granmo, O. C., & Oommen, B. J. An adaptive approach to learning the preferences of users in a social network using weak estimators. Submitted for publication.
32. Yazidi, A., Granmo, O. C., & Oommen, B. J. Service selection in stochastic environments: A learning-automaton based solution. To Appear in Applied Intelligence.
33. Yazidi, A., Granmo, O. C., & Oommen, B. J. (2010). A learning automata based solution to service selection in stochastic environments. In *Proceedings of the Twenty Second International Conference on Industrial, Engineering, and Other Applications of Applied Intelligent Systems (IEA-AIE 2010), Lecture Notes in Artificial Intelligence* (pp. 209–218).
34. Yu, Z., Zhou, X., Zhang, D., Chin, C. Y., Wang, X., & Men, J. (2006). Supporting context-aware media recommendations for smart phones. *IEEE Pervasive Computing, 5*, 68–75. doi:10.1109/MPRV.2006.61.

Author Biographies

Anis Yazidi received his Masters degree from University of Agder, Norway. Prior to that, he worked with Devoteam Telecom, Norway, before commencing his research at Agder Mobility Lab, University of Agder, Norway in 2008. His areas of interest include Network architectures, Mobile and Pervasive computing, Artificial Intelligence and Learning Automata.

Ole-Christoffer Granmo was born in Porsgrunn, Norway. He obtained his MSc in 1999 and the PhD degree in 2004, both from the University of Oslo, Norway. He is currently a Professor in the Department of ICT, University of Agder, Norway. His research interests include Intelligent Systems, Stochastic Modelling and Inference, Machine Learning, Pattern Recognition, Learning Automata, Distributed Computing, and Surveillance and Monitoring. He is the author of more than 55 refereed journal and conference publications.

B. John Oommen was born in Coonoor, India on September 9, 1953. He obtained his BTech degree from the Indian Institute of Technology, Madras, India in 1975. He obtained his ME from the Indian Institute of Science in Bangalore, India in 1977. He then went on for his MS and PhD which he obtained from Purdue University, in West Lafayette, Indiana in 1979 and 1982 respectively. He joined the School of Computer Science at Carleton University in Ottawa, Canada, in the 1981–1982 academic year. He is still at Carleton and holds the rank of a Full Professor. Since July 2006, he has been awarded the honorary rank of Chancellor's Professor, which is a lifetime award from Carleton University. His research interests include Automata Learning, Adaptive Data Structures, Statistical and Syntactic Pattern Recognition, Stochastic Algorithms and Partitioning Algorithms. He is the author of more than 355 refereed journal and conference publications, and is a Fellow of the IEEE and a Fellow of the IAPR. Dr. Oommen has been on the Editorial Board of the IEEE Transactions on Systems, Man and Cybernetics, and Pattern Recognition.

Martin Gerdes has studied Electrical Engineering at the Technical University (RWTH) in Aachen/Germany until 1998, and became a researcher at the Ericsson Eurolab in Herzogenrath/Germany afterwards. His research experience spans from performance analysis of IP based protocols over cellular communication networks, over mobile payment solutions, to architectures for pervasive communication environments like residential and personal networks. Currently he works as senior researcher on service network solutions, with focus on end-to-end architectures for automotive and other M2M environments utilizing cellular broadband communication infrastructures.

Frank Reichert has been working for over 25 years on technology and strategies for fixed and wireless communication systems both on national and international level. Since July 2005 he is with UiA (University of Agder), undertaking research in wireless communication networks and services. Currently he is the Dean of the Faculty of Engineering & Science. Cooperations include projects with Ericsson on wireless residential communications, and the Norwegian Center for Wireless Innovation with six other institutions. From 1995 he was with Ericsson Sweden, guiding investigations on, e.g., Future Service Layer Architectures, Wireless Internet technologies, and 3G applications and terminals. Assignments included creating and managing European Research projects (e.g. EUREKA Subproject PRO-COM, ACTS OnTheMove). Frank established Ericsson Cyberlab Singapore in 1999, focusing on user centric, ethnographic application and terminal design, as well as rapid prototyping of new HW/SW products exhibited at fairs like CeBIT 2001 and COMDEX (e.g. Ericsson Cordless WebScreen, Delphipad, Nanorouter). He was board member for Singapore CWS (Centre for Wireless Communications). He has been working as an expert, evaluator and auditor for the European Commission in industrial R&D frameworks such as RACE, ACTS, 5FP, 6FP, and Celtic Calls 1–3. Dr. Reichert holds a PhD degree in Electrical Engineering from Aachen University of Technology, Germany. He has published about 60 papers at international conferences on mobile technologies.