A Hybrid Model of Context-aware Service Provisioning Implemented on Smart Phones

Oriana Riva
Helsinki Institute for Information Technology
P.O. Box 9800, FIN-02015 TKK, Finland
Email: oriana.riva@hiit.fi

Santtu Toivonen
VTT Technical Research Centre of Finland
P.O. Box 1000, FIN-02044 VTT, Finland
Email: santtu.toivonen@vtt.fi

***Abstract*— Mobile users of future ubiquitous environments require novel means for locating relevant services available in their daily surroundings, where relevance has a user-specific definition. In this paper, we propose the hybrid service provisioning model which complements the traditional model of interaction service provider to consumer with peer-to-peer functionalities. In addition to being notified proactively and in a context-aware manner about services available in the surroundings, users can generate several types of contextual messages, attach them to services or to the environment, and share them with other peers. We have designed and implemented a platform that supports this hybrid service provisioning model. The current application prototype runs on commercial smart phones. To demonstrate the feasibility and technical deployability of our approach, we conducted field trials in which the research subject was a community of recreational boaters.**

I. Introduction

The emergence of small and relatively powerful mobile devices and the introduction of various wireless connectivity solutions such as 3G, Wi-Fi, and Bluetooth have had a tremendous impact on society by technologically enabling the deployment of ubiquitous services and applications. However, given the huge amount of information potentially available in ubiquitous environments, and the limitations anyway present in mobile devices such as PDAs and smart phones (e.g., small-sized screens, limited processing power), the ways of searching content and discovering services are also evolving. The challenge is to locate anytime, anywhere "relevant" content which is available in our daily environments, where relevance has a user-specific definition (e.g., cost, location, accessibility, etc.). In this regard, according to the I-centric model proposed by the WWRF (Wireless World Research Forum), personalization, ambient awareness, and adaptability are the three key-properties that set functional requirements on future service platforms [1]. For instance, a city guide assistant should dynamically retrieve maps and information about shops, restaurants, and monuments located in the user's vicinity and depending on the user's current task and preferences.

Research efforts have been undertaken to develop systems that accomplish this user-centric model in mobile environments. Mobility-aware recommenders such as the PILGRIM system [2] use the user's location to filter web links of interest. Context-based applications such as tourism information services [3] or remembrance agents [4] proactively retrieve content of interest based on the user's current task. Although these systems target different needs, they are mostly based on a common design: the system collects different types of (contextual) information characterizing the user, and uses this to filter and/or rank relevant content items. However, this approach does not always reflect what our daily life practice is. Let us consider this example[1]

> *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?" Frequently, people call or e-mail to tell me that they've eaten at a restaurant, or read a magazine article that they know I'll love. Businesses thrive or whither based on this informal method of marketing. But is there a way to harness and direct the power of word-of-mouth advertising?*

This extract highlights two motivations behind our work. Many information retrieval, information filtering, and recommendation systems are based on the construction of preference models; preferences can be built by directly asking the user to fill out a form, by collecting feedback every time she uses the system, or by observing her behavior over time. We argue that in many cases, systems supporting context-based content provisioning cannot assume to have complete access to user's personal and contextual information. Reasons for this could be user's reluctance due to privacy, security, and trust concerns, or mere laziness. Conversely, we observe that information about the user that the system cannot acquire might be known by other users such as friends, colleagues or people in the proximity.

Our second motivation derives from the observation that the success of phenomena such as Amazon have demonstrated the importance for customers to browse reviews and ratings expressed by previous clients. Typically, service content provided by professional content providers (meaning commercial service providers or public administration bodies) is officially expressed, impersonal, and utility-oriented. Nevertheless, the

[1] from the article "Dynamos: Figuring Friends into the Services Equation" appeared in the The Feature (http://www.thefeaturearchives.com/101445.html), Feb 2005.

information space easily becomes static and out of date as updating information implies a cost and is time consuming. Rather than being a monolithic and static concept, we consider the service space as a dynamic and malleable region of interaction and experience, in which occupants or visitors are able to capture their live experience and create a record to other users for later access and review [5], [6].

Stemming from these observations, we propose a hybrid model of context-aware service provisioning. It is our intention to complement the common user-centric model of content provisioning with peer-to-peer social functionalities[2], which are largely successful in Internet communities and on which we commonly rely in our daily life. Moreover, it is our objective to concretely apply such a model to daily scenarios involving mobile users equipped with commercial devices such as smart phones.

In this paper, we present how the hybrid service provisioning model has been applied in the DYNAMOS project[3]. Our platform enables mobile users *i)* to be proactively provided with a subset of relevant services available in the territory; *ii)* to generate several types of contextual notes, attach them to the environment, and eventually share them with other users; and *iii)* to annotate official service descriptions with personal observations, comments, ratings to be shared with others. To evaluate the feasibility and usefulness of our model, we considered a sailing scenario. We implemented an application prototype running on smart phones and specifically deployed to target the needs of recreational boaters. We then conducted field trails in which sailboaters utilized our application prototype during a sailing regatta.

The remainder of the paper is organized as follows. Section II describes the hybrid service provisioning model. In Section III, we explain how the model has been applied in the DYNAMOS project. Section IV describes our use case and results from the field trials we conducted. Section V presents related work. The paper concludes in Section VI.

II. Hybrid Service Provisioning

The main goal of the hybrid service provisioning model is to offer mobile users a means to *i)* constantly receive information regarding services of interest in the present situation, and *ii)* share information about services in a small/medium -sized community of users (e.g., among members of a sports club, friends, or colleagues). Essentially, as Fig. 1 depicts, the hybrid service provisioning model combines the traditional service provider to consumer model (B2C) with the consumer to consumer (C2C) model. In our approach, service providers offer users official content regarding available services (service-generated content), and do this in a context-aware manner. Additionally, users can generate unofficial content such as observations and personal opinions (user-generated content),

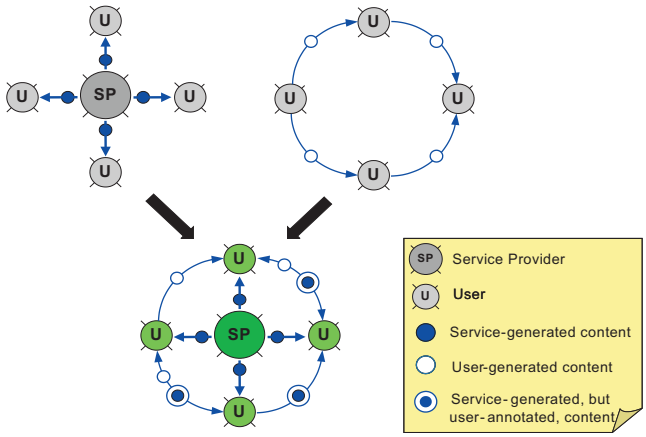

Fig. 1. Hybrid service provisioning model

attach it to service descriptions (service-generated, but user-annotated, content), and eventually share it with one or more users within a certain community. For instance, after visiting a sports center a user can create a comment stating "*Today 50% discount on trekking equipment*" and send it to a friend along with the pointer to the shop.

To describe context-based service provisioning models, metadata can be used to characterize capabilities and requirements of users, devices, and services. Among the several possible types of metadata, profile and policies have been largely exploited in distributed systems and also in supporting context management (e.g., [7], [8]). In our case, *profile* and *context* information are metadata describing the user instance. Profile represents the nearly static characterization of the user, whereas context is the dynamic characterization. In other words, users make some aspects of their internal representation explicit by expressing their preferences, interests, and control requirements in profiles, and by disclosing their location, activity, and mood in context information. Likewise, we consider *service descriptions* as metadata elements that characterize the service instance. These are usually supplied by official service providers.

As the upper part of Fig. 2 shows, a traditional service provisioning system decides whether to establish a "link" between a user `U1` and a service `S1` (that is, it provides the user `U1` with the service `S1`) upon matching metadata elements that are system-accessible. Metadata are either explicitly submitted by users and service providers or implicitly learnt by the system based on previous observations and inference algorithms. In addition to links entirely recognized by the system, our hybrid service provisioning model (bottom part of Fig. 2) allows other users to participate (implicitly or explicitly) in establishing this kind of links between services and a certain user. In this case, the matching involves a larger number of metadata submitted by several users. We distinguish two cases.

Case 1): In the case `U1-U2-S1`, `U1` receives a recommendation about `S1` from `U2`. `U2` could be a friend, colleague or even somebody unknown, but who knows something about `U1` (e.g., because he is in the proximity of `U1`). The potential value

[2]hereinafter we refer to peer-to-peer more as a model of social communication and interaction between persons than as the architectural design of P2P networks

[3]Dynamic Composition and Sharing of Context-Aware Mobile Services. URL: http://virtual.vtt.fi/virtual/proj2/dynamos/

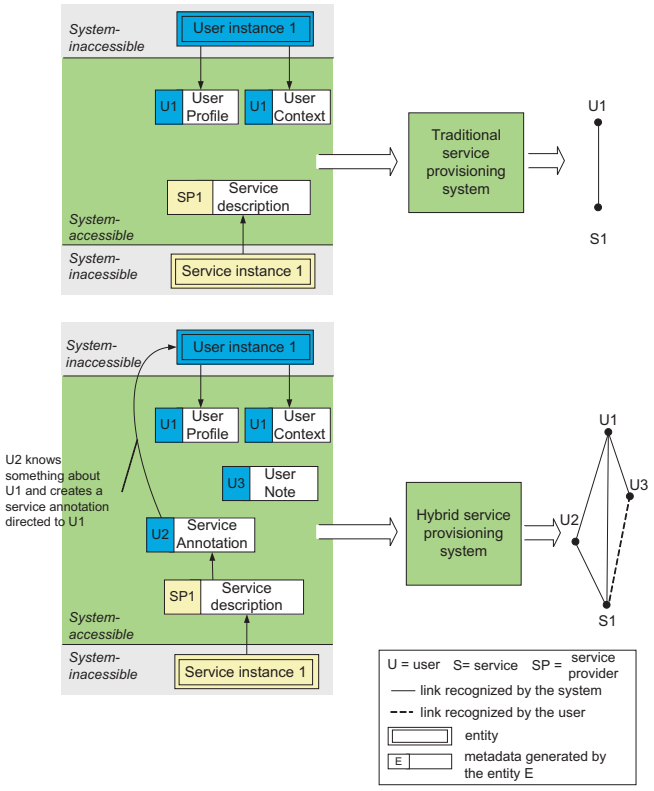

Fig. 2. Comparison of a traditional service provisioning model and our hybrid model

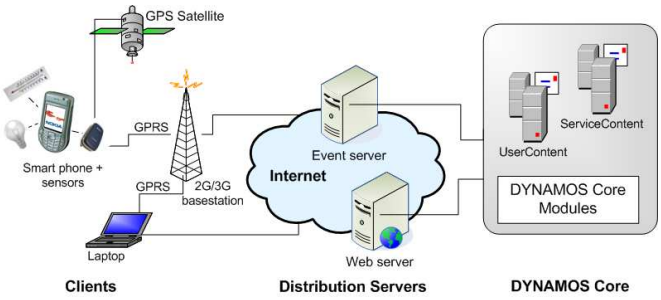

Fig. 3. The DYNAMOS overall reference architecture

of this kind of recommendations resides in the capability of other people to access implicit profile and context data characterizing the user (i.e., content that is system-inaccessible). For instance, think that U2 is visiting a shopping mall and wants to notify her friend U1 that in the shop S1, the sporting shoes she was looking for are now 50% discounted. Or consider a person U1 attending a conference who is informed by an unknown participant U2 that books written by the invited speaker can be bought at the university bookstore S1. We call this kind of recommendations that users create, attach to services, and share with others *service annotations*.

Case 2): In the case U1–U3–S1 involves another type of user-generated content that does not explicitly refer to a service description. We call this content *user note*. A user note can contain warnings notifying dangerous situations, observations related to the environment, notifications about special happenings. Users generate user notes and attach them to the environment. Later on, other users can be automatically provided with this content if it is of interest to them (e.g., it is in their proximity). In particular, a user note attached to a certain location can contain some information that is related, even though not explicitly, to the status of nearby services. This information can further help users to take a decision about which service to access when multiple services are available. For instance, if a driver U3 discovers a traffic jam on the highway he can create a message to warn upcoming drivers. A driver U1 who is directed to a certain restaurant located in the area of the traffic jam, upon finding this warning, might decide to drive to a similar restaurant S1, but located in a less crowded area.

The benefits of supporting generation and sharing of metadata such as service annotations and user notes are twofold: *i)* they enrich and further specify services with information not always supplied by official organizations or providers (such as personal experiences, description of the current status of the service or current state-of-affairs); *ii)* they offer a means to share in a context-aware manner information about services and the surrounding environment (hence, each user becomes a potential service provider for others).

III. Hybrid Service Provisioning Platform

In the DYNAMOS project, the hybrid service provisioning model has been applied to build a platform supporting service provisioning for mobile users of ubiquitous environments. This section describes the DYNAMOS overall reference architecture and how this platform has been designed and implemented.

A. Overview of the DYNAMOS Reference Architecture

As Fig. 3 depicts, the DYNAMOS overall reference architecture consists of three major parts: *Clients*, *Distribution Servers*, and *DYNAMOS Core*.

Clients accessing the system can be of two types. The first type includes any device that supports a web browser. A client like this is intended to support ”heavier” functionality such as profile creation or trip planning with online maps; a large screen is usually preferred. The second type of client includes mobile devices such as smart phones which often offer limited resource capabilities; several types of sensors can be connected to them for collecting context information.

Distribution Servers are employed to enable clients to access the DYNAMOS Core in different ways. Currently, Web Servers and Event Servers are integrated. The Event Server allows client applications running on mobile devices to notify the DYNAMOS Core as soon as relevant changes occur (e.g., context changes, new user content to be shared, etc.).

DYNAMOS Core consists of several DYNAMOS modules and Content Servers. The modules composing the DYNAMOS Core provide key functionalities for supporting context-based content provisioning. The *User Content Server* stores user-related information (profile and context). The *Service Content*

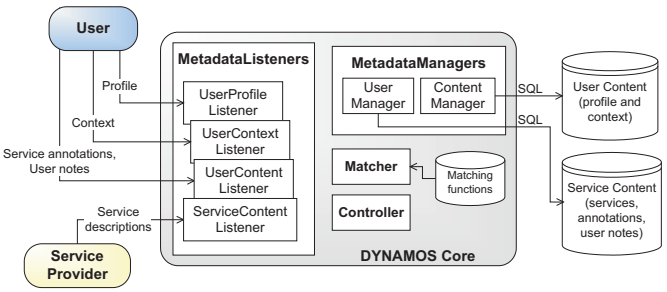

Fig. 4. The DYNAMOS Core architecture

Server stores content provided by service providers (service descriptions) and user-generated content (user notes and service annotations). The service content can be distributed across different repositories according to different logics. For example, single databases can store items relatively to a certain location or topic, and a synchronization mechanism is deployed among all local servers. Additionally, the distribution of context information could follow a distribution logic based on the type of context data, like done in [9].

B. DYNAMOS Core Architecture

Fig. 4 shows the major architectural components of the DYNAMOS Core: *MetadataListeners*, *MetadataManagers*, *Matcher*, and *Controller*.

MetadataListeners allow the specification and update of different types of metadata information characterizing users (*UserProfileListener* and *UserContextListener*), user-generated content (*UserContentListener*), and service content (*ServiceContentListener*). Anytime a Listener receives a new notification, it invokes the MetadataManagers. The two types of MetadataManagers process the received information and stores it in the corresponding ContentServers.

The Matcher is responsible for matching available service content and user's profile/context in order to extract a list of relevant items to provide the user with. When needed, the MetadataManagers invoke the Matcher module. For example, if the notification reports changes to the context of active users, it is necessary to invoke the Matcher for verifying whether new matched content should be sent. If the notification carries information about the status of currently non-matched services, no matching needs to be triggered, but such information is stored in the repositories. The Matcher accomplishes its task by utilizing several types of matching functions, such as category-based matching and location-based matching. The Controller is in charge of enforcing control policies and preferences specified in the user profile. For example, this module can perform a further filtering on the set of matched service items or apply a certain delivery format to the matched content.

User and service content is stored respectively in the *UserContent* and *ServiceContent* relational databases. The MetadataManagers are responsible for coordinating the access to these databases and for translating database queries into SQL format, and vice versa. However, due to performance reasons, content that needs to be frequently accessed (e.g. context data of active users) is also kept in memory, thus avoiding time consuming MySQL operations.

In order to better understand how these architectural components work, in the following, we describe how context/profile information and service content have been modeled, and we shed some light on the Matcher and Controller modules.

1) Service Content and User Content: The three types of content of exchange at the basis of the hybrid service provisioning model have been implemented through three objects with the following properties.

The `Service` object contains the category to which the service belongs, a textual description of the service characteristics, its location, opening times, and contact information. This kind of information is provided by commercial content providers and public administration organizations.

The `ServiceAnnotation` object contains information in textual and multimedia format. The possibility to include multimedia content, such as images and audio clips, allows users to quickly and easily generate comments without the need for typing on a small keyboard, especially if this needs to be done while involved in other activities. The annotation also contains a hyperlink to the service to which it refers; hence, through the service annotation, it is possible to retrieve the whole service description. Moreover, in an annotation, the author can express a rating for the quality of the service, provide his signature, and specify the recipient(s) to whom the annotation is directed. The recipient can be the author himself, if the annotation is meant for private use (typically as a reminder), or it can be anyone, if the annotation is meant to be public. Alternatively, other recipients can be single users or a restricted community such as a club, a team, or a group of friends. Location (if the user allows) and time of creation are also included in the annotation. A default or user-specified lifetime is associated to each service annotation in order to guarantee temporal relevance of the information.

A `UserNote` object is structured as a `ServiceAnnotation` with the main difference that it does not contain the explicit link to any service. User notes belong to different categories that can be further specialized based on the scenario of use. For example, user notes can be routine messages, safety warnings, and emergency alarms.

2) User Profile and Context: The context information we consider includes, in principle, any information that can be used to characterize the situation of a mobile user requesting a service [10]. Context consists of spatial information (location, speed, orientation), temporal information (time, duration), user characterization (activity, social surroundings, personal interests, preferences), environmental information (temperature, light, noise), and resource availability (nearby devices, device status). In our model, we use the term profile to specifically refer to a subset of context information consisting of nearly static and personal information such as preferences, interests, and control policies (i.e., rules that control the system's behaviour).

The profile is specifically structured to support efficient

matching between available services and users. Essentially, in their profile, users *i)* define several types of activity or status, and *ii)* associate multiple interests to each of them. Activity names are user-defined. Interest names are predefined based on the available service taxonomy. For instance, in the following profile code excerpt, the user `Setoivon` specifies his interest in `GuestHarbors` and `Weather` services while `Sailing`, and in `Restaurants` when `AtHarbor`.

```
<Activity rdf:ID="Sailing">
    <hasInterest>
        <ServiceCategory rdf:ID="Weather"/>
    </hasInterest>
    <hasInterest>
        <ServiceCategory rdf:ID="GuestHarbors"/>
    </hasInterest>
</Activity>
<Activity rdf:ID="AtHarbor">
    <hasInterest>
        <ServiceCategory rdf:ID="Restaurants"/>
    </hasInterest>
</Activity>
<User rdf:ID="Setoivon">
    <hasActivity rdf:resource="#AtHarbor"/>
    <hasActivity rdf:resource="#Sailing"/>
</User>
<Service rdf:ID="Foreca">
    <hasCategory rdf:resource="#Weather"/>
</Service>
<Service rdf:ID="KokarGuestHarbor">
    <hasCategory rdf:resource="#GuestHarbors"/>
</Service>
```

Note that even though in the current implementation, the user is requested to explicitly specify her interests and activities, this could in principle be partly or entirely replaced by a learning module. This could infer the user's characterization by monitoring the user's behavior and by subsequently associating recurring patterns to certain activities. For instance, if every morning a certain person checks news while having breakfast and the bus timetable before leaving, interests like `News` and `Bus` can be automatically gathered and associated to an activity labelled as `morning-7am'`. This approach was applied in [11] to infer the context of use of a mobile device.

To support the collection of context information on smart phones, we utilize the Contory middleware [12]. Contory has been specifically deployed to support alternative mechanisms of context provisioning on resource-constrained devices such as smart phones. Supported context provisioning strategies are internal sensor-based provisioning, external infrastructure-based provisioning, and distributed context provisioning in ad hoc networks. The advantage of employing this middleware is that to gather context information of interest, the application can *i)* rely on its own sensors, if available; *ii)* interact with external context infrastructures; or *iii)* exploit neighboring devices willing to share their context information. The programming API of Contory is designed in such a way that applications submit SQL-like context queries, and Contory return matching results to the application. Context queries permit to specify the type and quality of the desired context items (e.g., accuracy, correctness, freshness, etc.), context sources to be employed, push or pull mode of interaction, and query lifetime.

3) Matcher: To select content of interest based on the current user's situation and needs, it is necessary to match all available content with user's profile and context.

As far as service items, the matching is achieved in two steps. Let I be the set of possible user interests and S the set of available service descriptions. First, each active interest i (i.e., associated to the current activity) is matched with the set S_{s_j} of service categories, which are associated to each available service s_j. As shown in the previous example, service categories generally specify the business branches of the service (e.g., `GuestHarbors`, `Weather`). If we call $M: I \times S \rightarrow \{0,1\}$ the function for matching interests and service categories, and we define the function $s^*: \mathbf{R}^+ \rightarrow \mathcal{P}(S)$ which returns the set of matched service items at a certain time t, we can express the filtering process as:

$$s^*(t) = \{s \in s(t) \mid M(i(t), s_c) = 1\} \tag{1}$$

Subsequently, to each service s_i^* is associated a matching rank value $r_{s_i^*}$ which is calculated by matching the user context information c_k and the service description s_i^*. Specialized filters $f_k(c_k, s)$ are deployed for each type of context information. For example, `Close(location,s)` uses the location to select nearby services, `Open(time,s)` uses the time to select open services, and `MatchWeather(weather,s)` establishes if a service is compliant to the weather conditions (e.g., indoor services if it rains). Moreover, to each type of context information a different weight w_k is associated, so that more relevant context information can have a major impact on the final matching rank. Formally:

$$r_{s_i^*} = \sum_{k=1}^{N} w_k * f_k(c_k, s_i^*) \tag{2}$$

Matched service items are then ordered based on the computed r_s values and delivered to the user.

Service annotations directed to anybody (that is, of public use) are filtered according to the matching result of the associated service items. Service annotations directed to single recipients or a community are filtered based on the control policies specified by the recipient. As far as user note items, the matching process consists only of category-matching (like for service items), time-filtering, and location-filtering.

The matching subsystem is designed as a *Strategy* pattern [13]. Different matching algorithms are appropriate at different times and for different types of content. Hence, it is necessary to easily interchange these matching algorithms. M and f_k can implement exact-match based on simple boolean logic or best-match (i.e., provides a ranked list of content items based on relevance). Furthermore, the system must be able to employ different matching algorithms based on the quality and quantity of context information available. For example, certain sources of context could be temporarily unavailable and therefore rendering not applicable all f_k filtering functions.

4) Controller: As in most open approaches, the risk of attacks from malicious users and spamming are crucial issues

also in our system. For example, a malicious user could create wrong warning notes to her advantage or attach false recommendations to certain services. In the case of many users of this type, the system would be filled with too many spam messages and thus become too intrusive and in the end unusable. To partly address these problems our system employs autonomous and collaboration-based control policies.

Single users can specify autonomous control policies based on personal preferences and needs. These policies are directly enforced by the *Controller*. Control policies express conditions about which type of content the user wishes to receive, from which type of content providers, and with which frequency. For example, a user can request to be informed about emergency warnings anytime, but about leisure happenings only when in freetime. Additionally, the user can choose to operate in different modes of interaction: she can instruct the system to deliver only official service content (i.e., coming from service providers), only informal service content (i.e., coming from other peers), or both.

However, spam and trust problems cannot be entirely solved using rule-based mechanisms, even advanced once based on Bayesian inference mechanisms (see for example [14]). To alleviate trust issues and increase opportunities for user collaboration, we offer the user the possibility of creating communities. Every user interacts with one or more communities of users with whom she decides to share information. Each user can group users of her buddylist in different communities and define appropriate actions for every type of community. For example, messages from some communities are blocked in some situations, while higher priority is given to other communities. The trust on any member of the community must then be regulated by reputation mechanisms such as the one described in [15]. Any user in the community can send warnings about possible creators of spam messages. Then, each user locally maintains triplets as (u_i, m_s, m_t): for each user u_i of the community, m_s indicates how many spam messages such user sent to the community, while m_t represents the total number of messages exchanged in the community. When m_s exceeds a certain threshold, u_i is blocked.

C. Implementation Details

The whole platform has been implemented in Java. In the current implementation, event-based communication is realized through the Fuego Core Event Server [16]. The application running on mobile phones has been implemented using Java 2 Micro-Edition (J2ME) with the Connected Limited Device Configuration (CLDC) 1.0 and Mobile Information Device Profile (MIDP) 2.0 APIs. All the development was done using Nokia Series 60 phones. The mobile phone platform was selected since it currently represents a powerful computing platform which is already widely carried by many people. Disadvantages of this platform are the limited debugging support, the limited programming environment, slow storage access, and scarce availability of commercial sensor technology that can be connected to the mobile phone for gathering context information.

IV. Example Application: Recreational Boaters in the Turku Archipelago

To demonstrate the feasibility and usefulness of our hybrid model, we selected a community of recreational boaters as scenario of study. Recreational boaters are people who have either sailboating or motor boating as a hobby. This scenario was selected since it represents a typical case in which, given the intensiveness of the task and the dynamism of the situation, proactive and context-aware service provisioning can provide significant benefits.

Boaters constantly need up-to-date information about the surrounding environment. Typically this information is related to routes and weather conditions, but it can concern other services too. For example, in the evening, the boaters are likely interested in nearby guest harbors and the availability of spaces in them. The boaters can take advantage of the functionalities of our system before a trip (for planning an itinerary and select places of interest), during a trip (for being informed about available services and their real-time status as well as weather forecasts and ongoing events), and after a trip (for sharing their boating experience with others). In our study, we first conducted user interviews to gather requirements for the application design, and then implemented an application prototype running on smart phones, which specifically targets the needs of recreational boaters.

A. Requirements Study

We interviewed 20 Finnish boaters. We first described the basic concepts behind our model and presented the case application in more detail by means of several mock-ups illustrating the client interface[4]. Each interview lasted approximately 50 minutes and each interviewee was asked 30 general questions related to their personal boating habits/conventions and also opinions about the proposed application.

All interviewees mentioned using a mobile phone as a communication device while boating. In addition, the phone was used for checking weather conditions and forecasts (95%), following news (40%), and finding out about services (40%). 80% of the interviewees reported the usage of specific methods to find out about services in the territory, while 20% stated to uniquely rely on their own knowledge and experience. Specifically, 70% of the interviewees consult mainly printed material of some kind (maps, sightseeing guides) to find out about services and 92% of these explicitly mentioned the usage of an annual harbor book. Generally, people praise these harbor books[5]; the only downside to them is the lack of event-related information, their outdating in general, and their inefficiency in dynamically guiding sailboaters when, during the trip, they are forced to change their routes for some reason. Finally, the interviewees were asked about their frequency of visiting guest harbors (or in general harbors offering services of some kind) as opposed to nature harbors: 90% of the boaters' visits are to harbors with services.

[4]Questionnaires and mock-ups are available at http://virtual.vtt.fi/virtual/proj2/dynamos/interviews/

[5]See, for example: http://www.satamakirja.fi/

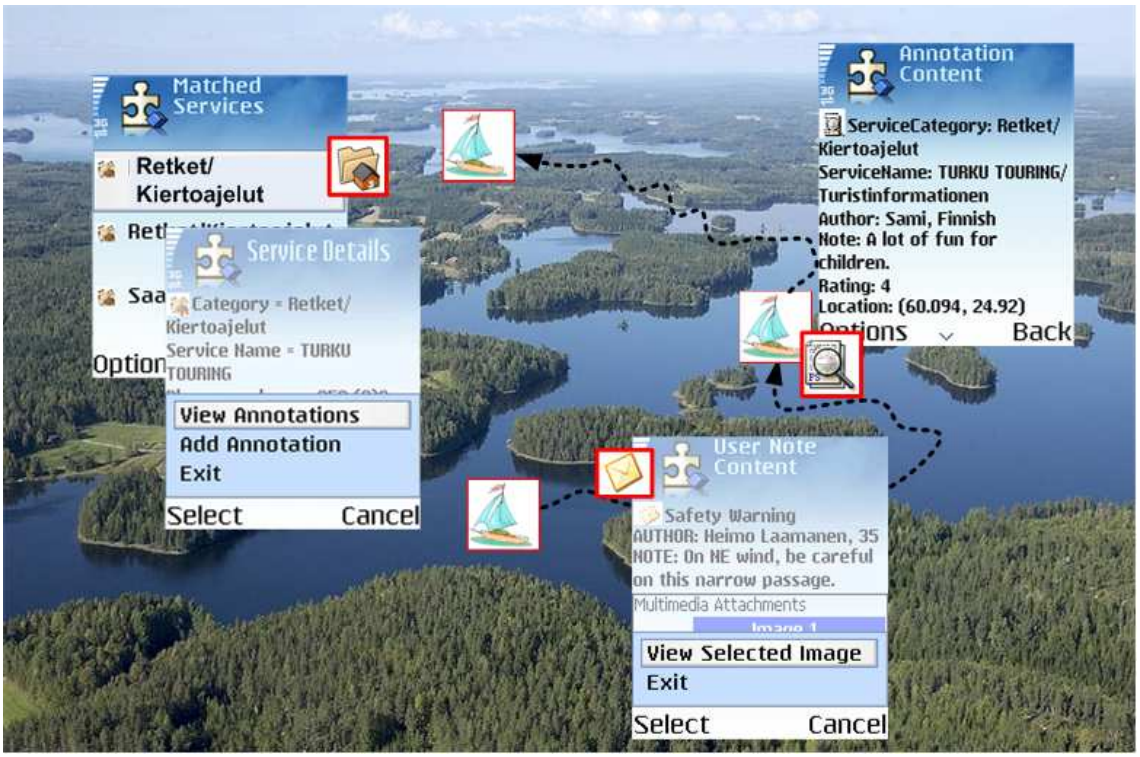

Fig. 5. Example of sailing scenario and screenshots from the phone interface

B. Application prototype

An example of how our application prototype works is given in Fig. 5. The sailboater is first provided with a safety message that warns him about the narrow passage next to him. Subsequently, he receives a recommendation from his friend Sami who suggests him to try sightseeing tours (Kiertoajelut in Finnish) organized in some islands in the surroundings. The sailboater checks the location and decides to follow the advice of his friend and try the place out. Once he has visited the place he can also further annotate the service description and share it with other friends.

C. Field Trials Evaluation

Two preliminary field trials were conducted in order to evaluate the feasibility and technical deployability of our hybrid approach through the implemented sailing application. On a meta-level, we were interested in finding out whether our approach was effective in supporting sharing of information about services in the surrounding environment, especially when people are engaged in other activities (in our case sailing). On a technical level, we were interested in investigating how reliable and robust our application was.

The archipelago in the South-Western Finland has over twenty thousand islands and skerries. Many of the bigger islands are partly inhabited and provide services the boaters can make use of. In addition, the coastline has cities and harbors which provide a number of services. We provided our ServiceContentServer with a database of about 1000 tourist services located in the Turku archipelago[6]. In June and August 2005, we conducted two preliminary field trials. The first one was a short excursion on a sailboat involving 4 users. The second one was a one-day regatta, which is organized every year by a club of sailboaters of Helsinki; this involved 9 sailboats with a total of 28 people. Both trials took place in the archipelago of the Helsinki region.

The first experiment was mostly aimed to investigate technical problems and performance of our application. The application was in use for 5 hours by two users while sailing. The performance of the application was affected by the continuous and not-optimized traffic of events sent to/from the phone. In some cases, this caused problems with phone memory consumption. Furthermore, occasionally, the GPS device disconnected from the phone due to Bluetooth (BT) disconnections (reported as a known issue in the NOKIA Forum [17]), and thus causing the system to collapse.

To improve our prototype application we highly reduced the traffic of events by transmitting only relevant location changes (e.g., only reliable and sufficiently different location updates were transmitted) and by delivering matched content as a differential from the previously delivered content. Since we could not solve reliability issues of the BT connection between phone and GPS, we simplified the client interface for connecting the GPS device to the phone and memorized addresses of previously accessed GPS devices. This allowed to quickly reconnect the phone to the GPS device and avoid time

[6]TurkuTouring, URL: http://www.turkutouring.fi

consuming BT service discovery, every time a disconnection occurred. We also introduced an alert message to notify the user every time a GPS disconnection was detected by the client application. The downside with alert messages in intensive environments such as a boat on the move is that they do not always manage to get the user's attention.

The second trial was of major impact. 28 persons participated to the event. Among those, 9 expert sailboaters from 30-40 years old installed and used our application. The rest of the people had varying sailing expertise, ranging from very low to very experienced. Each boat was equipped with a Nokia 6630 phone connected to Bluetooth GPS Receiver InsSirf III[7]. The day before the event, sailboaters were instructed about how to use our application and connect the GPS via BT to the phone. A 2-page instruction to consult on the boat was also distributed. The regatta lasted approximately 7 hours.

At the beginning of the regatta, sailboaters easily connected their phone to the GPS device and logged into the system. However, during the sailing route, 2G/3G handover caused some technical problem. Every time a 2G/3G handover occurred while a HTTP connection was open, the smart phone switched off permanently, thus requiring the user to re-login and re-connect the GPS. 2 sailboaters who had set their phone to operate in 2G network mode did not encounter this problem, and their location track was fairly continuous. Occasionally, some BT disconnection problems still occurred (typically an average of 1 disconnection every hour). However, thanks to the modifications introduced after the previous regatta, sailors could quickly reconnect the GPS and the interruption in the location monitoring was of irrelevant duration. Therefore, despite these network problems, from a technical point of view, the system performed successfully and sailboaters were constantly provided with available matched content.

The sailing activity alternates periods of extreme intensiveness (e.g., tacking against a heavy wind) to periods of low activity (e.g., cruising on a mild tailwind). The field study revealed how sailboaters enjoyed filling those not-busy time intervals by creating observations, posting contextual messages, and thus generating a sort of user trace in the environment, as if it was a message like "I passed from here, I saw this, and I visited this". Almost all generated content had at least one image as attachment and generally a short text. This revealed how the possibility of leaving social traces, especially in unknown and vast territories, is rather appealing in these kind of activities. This is also associated to a sense of "curiosity" in knowing what others are doing (e.g., " Are they moving faster?"), or they have done in the same situation (e.g., "Have they visited this guest harbor or taken this route?"). During the regatta, there was a reciprocal interest in people passing by. Moreover, people found the possibility of knowing about past experience and finding signs left by others especially useful in this kind of silent, not highly populated, and typically unknown places.

During the regatta, people were advertised about services in a range of 5km. Even though all sailing boats followed the same route and thus passed closed the same services, the presence of contextual notes attached to the environment and annotations attached to services was considered a potential means for decision-making regarding which route to take, which guest harbor to visit, or in which phase of the trip explore new places. For this kind of activity, finding out about services in the surroundings is generally not a trivial task, especially for people not very experienced or new to the region. Because going ashore with a sailboat requires a lot of work (such as taking down the sails and packing them, taking out the chest ropes, starting the engine, and so forth), people considered highly important to know in advance about the status of some services that they intended to visit.

People who (directly or indirectly) used our application during the sailing trials or in other public demonstrations of the system, were asked to fill in a questionnaire concerning usefulness of the system, privacy and trustworthiness, advantages and disadvantages, technical problems encountered while using the application. The feedback we received was very positive. Summarizing, among the functionalities offered by the system, people found particularly interesting the possibility of expressing and sharing recommendations about services (80%) and of creating informative notes for other users (75%), for personal use as reminders (62%) or for coordinating a trip with friends (60%). They also said that a system like ours (also applied to different scenarios) could be very useful in many concrete situations and they would be ready to pay something for using it. The risk of spam as well as untrustworthy recommendations were the main threats and disadvantages observed about the system.

In the summer 2006, the DYNAMOS project is scheduled to conduct field trials of the final system. We are confident to solve most of the technical problems we encountered in these trials by employing newer models of smart phones and operating in 2G mode. The event will involve a larger number of users and more functionalities of the system will be evaluated. In particular, we plan to add other context-based functionalities, such as the support for "weather notes"; these will allow users to query the system about weather conditions in a certain region.

V. Related Work

Our research shares some concepts and techniques used in conventional recommender systems. With the emergence of e-services, recommenders have been often applied in various forms and fields, and especially in information retrieval. Recommenders are capable of ranking a list of similar items with respect to a certain criteria in order to provide recommendations, predictions, opinions that can assist a user in evaluating items [18]. The two most applied recommendation techniques are the content-based approach (see for example SISTER [19]) and the collaborative filtering approach (see for example Phoaks [20]). In content-based recommendation, the system suggests items that best match a specified criteria. The system has to understand the main features describing the

[7]http://www.insmat.com

items of interest in order to establish their relevance, and they usually maintain domain-specific user profiles. Collaborative filtering is based on the assumption that if the interests of a certain user *a* or a community *A* are similar to those of an another user *b*, the items preferred by *b* can be recommended to *a*. Hybrid systems (see for example Yoda [21]) combine collaborative filtering and content-based querying to achieve higher accuracy.

Lately, some attempts have also moved towards the integration of context-awareness in recommenders. In [2], the authors employ a historical database of user locations and URLs to determine where and how often certain links have been accessed with the aim of providing accurate recommendations. In the recommender system described in [22], the authors exploit a wide and dynamic range of context information, and specifically focus on designing a learning algorithm to identify the user contextual preferences when ranking items. As in these two approaches, our model utilizes a similar context-based recommendation strategy. However, we differentiate from conventional recommender systems by combining the ranking process of recommenders with peer-to-peer functionalities. Moreover, the most consistent part of this type of research has traditionally been applied to relatively static scenarios and to Internet communities, while in our study we target mobile users in dynamically changing mobile environments.

The use of context information in applications running on mobile devices has received large attention in many research areas such as ubiquitous computing, mobile computing, augmented reality, and human-computer interaction. In particular, the most publicized context-aware systems are location-based systems such as navigator assistants, location-aware information services (e.g., weather services and tourist guides), location-sensitive games. Examples include GUIDE [23], which focuses on implementing a context-sensitive tourist guide for visitors in the city of Lancaster, and WebPark [24], which provides location-based services in protected and recreational areas, such as coastal, rural, and mountainous regions. Except for few narrow exceptions, most of the context-aware applications so far implemented are based on technologies that are not supported by commercial smart phones, or they assume more resources than those that smart phones can offer. For example, most of the existing prototypes run on laptops and PDAs. In our work, all the development was done using Nokia Series 60 phones. Furthermore, it was our intention to exploit not only location, but a wider range of contextual information, to accomplish more diversified content matching.

Finally, as far as mobile technologies for tourism, [25] pointed out how typical location-based systems do not allow "pre-visiting" and "post-visiting". Pre-visiting is about planning the visit; post-visiting is about reminiscing and sharing the experience. To accomplish this, our work reuses past research experiences about attaching signs or leaving marks to visited environments as done in the tour of Disneyland application of Pascoe [26], GeoNotes [5], E-graffiti [27], and virtual graffiti [28]. The possibility to attach content of different types to contextual situations permits to store content for future usage by the user herself, friends, or anyone else.

VI. Conclusions

In this paper, we have proposed the hybrid service provisioning model, a novel approach to access and share information and services available in the surrounding environment. We have described the design and implementation of a platform that supports this model, as well as results from initial field trials. Our model has proved so far to be an interesting and useful approach for supporting a novel conception of service provisioning in mobile environments. Experiences in the real field of action allowed us to investigate the user's reception to a system like this in real-world use. Based on numerous feedbacks we received when presenting the system to (both boaters and non-boaters) potential users, many of the functionalities of the sailing application were found interesting and potentially extensible to other daily life scenarios. Interviewees themselves suggested to deploy similar applications for traffic services, nature trail, cultural event information, public transportation helper, sport games, conferences, and travel guides. Additionally, the field trial also allowed us to identify and solve technical problems of location detection, network connectivity, and system performance.

Acknowledgment

The authors would like to thank Tapio Pitkäranta for his contribution to the implementation of the DYNAMOS system and organization of the field trials. We also thank Jaakko Kangasharju and Sasu Tarkoma of the Fuego Core project for their assistance during the system development. Special thanks also go to Heikki Helin and Michael Przybilski for their advice and useful comments. Thanks to the Turku Touring for providing the project with a service database for test use and to Insmat for supplying us with GPS devices for the field trial. Finally, we also would like to thank TEKES, ICT-Turku, Suunto, TeliaSonera, and VTT for funding the DYNAMOS Project.

References

[1] S. Arbanowski, P. Ballon, K. David, O. Droegehorn, H. Eertink, W. Kellerer, H. van Kranenburg, K. Raatikainen, and R. Popescu-Zeletin, "I-centric communications: personalization, ambient awareness, and adaptability for future mobile services," *IEEE Communications Magazine*, vol. 42, no. 9, pp. 63–69, September 2004.

[2] M. Brunato and R. Battiti, "PILGRIM: A Location Broker and Mobility-Aware Recommendation System," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*. Fort Worth, Texas, USA: IEEE Computer Society, March 23-26 2003.

[3] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou, "Developing a context-aware electronic tourist guide: some issues and experiences," in *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, USA: ACM Press, 2000, pp. 17–24.

[4] B. J. Rhodes, "The wearable remembrance agent: A system for augmented memory." in *ISWC*, 1997, pp. 123–128.

[5] F. Espinoza, P. Persson, A. Sandin, H. Nyström, E. Cacciatore, and M. Bylund, "GeoNotes: Social and Navigational Aspects of Location-Based Information Systems," *Ubicomp 2001: Ubiquitous Computing, International Conference*, pp. 2–17, September 30 - October 2 2001.

[6] G. Abowd, "Classroom 2000: An experience with the instrumentation of a living educational environment," *IBM System Journal*, vol. 38, no. 4, pp. 508–530, October 1999.

[7] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli, "Context-aware middleware for resource management in the wireless internet." *IEEE Trans. Software Eng.*, vol. 29, no. 12, pp. 1086–1099, 2003.

[8] L. Capra, W. Emmerich, and C. Mascolo, "Carisma: Context-aware reflective middleware system for mobile applications." *IEEE Trans. Software Eng.*, vol. 29, no. 10, pp. 929–945, 2003.

[9] M. Großmann, M. Bauer, N. Hönle, U.-P. Kappeler, D. Nicklas, and T. Schwarz, "Efficiently managing context information for large-scale scenarios." in *PerCom*, 2005, pp. 331–340.

[10] A. K. Dey, D. Salber, and G. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Human-Computer Interaction (HCI) Journal*, vol. 16, no. (2-4), pp. 97–166, 2001.

[11] J. Himberg, J. A. Flanagan, and J. Mntyjrvi, "Towards context awareness using symbol clustering map," *Workshop on Self-Organising Maps*, September 11-13 2003.

[12] O. Riva and C. di Flora, "Contory: A smart phone middleware supporting multiple context provisioning strategies," *2nd International Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet (SIUMI'06)*, 4-7 July 2006.

[13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[14] N. Dimmock, J. Bacon, D. Ingram, and K. Moody, "Risk Models for Trust-Based Access Control (TBAC)," in *iTrust*, 2005, pp. 364–371.

[15] D. Zhu and M. W. Mutka, "Promoting cooperation among strangers to access internet services from an ad hoc network," in *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, Orlando, FL, 2004, pp. 229–240.

[16] Fuego Core Project, "http://www.hiit.fi/fuego/fc/."

[17] FORUM NOKIA, "2nd Edition Platforms: Known Issues," http://www.forum.nokia.com/main/0,6566,58_10,00.html, November 10 2005, version 2.8.

[18] P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[19] J. Mostafa, S. Mukhopadhyay, M. Palakal, and W. Lam, "A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation," *ACM Transactions on Information Systems*, vol. 15, no. 4, pp. 368–399, October 1997.

[20] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter, "Phoaks: a system for sharing recommendations," *Commun. ACM*, vol. 40, no. 3, pp. 59–62, 1997.

[21] C. Shahabi, F. B. Kashani, Y.-S. Chen, and D. McLeod, "Yoda: An accurate and scalable web-based recommendation system," in *CoopIS '01: Proceedings of the 9th International Conference on Cooperative Information Systems*. London, UK: Springer-Verlag, 2001, pp. 418–432.

[22] G.-E. Yap, A.-H. Tan, and H.-H. Pang, "Dynamically-optimized context in recommender systems," in *Proceedings of the 6th international conference on mobile data management (MDM'05)*. New York, NY, USA: ACM Press, 2005, pp. 265–272.

[23] "Guide Project," http://www.guide.lancs.ac.uk.

[24] "Webpark project," http://www.webparkservices.info/.

[25] B. Brown and M. Chalmers, "Tourism and mobile technology," in *Proceedings of the 8th European Conference on Computer Supported Cooperative Work*. Kluwer Academic Publishers, 14-18 September 2003.

[26] J. Pascoe, "The stick-e note architecture: extending the interface beyond the user," in *Proceedings of the 1997 International Conference on Intelligent User Interfaces*. ACM Press, 1997, pp. 261–264.

[27] J. Burrell and G. Gay, "E-Graffiti: Evaluating real-world use of a context-aware system," *Interacting With Computers: Special Issue on Universal Usability*, vol. 14, pp. 301–312, 2001.

[28] R. Järvensivu, R. Pitkänen, and T. Mikkonen, "Object-oriented middleware for location-aware systems," in *SAC '04: Proceedings of the 2004 ACM Symposium on Applied Computing*, 2004, pp. 1184–1190.