The DYNAMOS Approach to Support Context-aware Service Provisioning in Mobile Environments

Oriana Riva [a,*] Santtu Toivonen [b]

[a] *Helsinki Institute for Information Technology*
P.O. Box 9800, FIN-02015 TKK, Finland

[b] *VTT Technical Research Centre of Finland*
P.O. Box 1000, FIN-02044 VTT, Finland

Abstract

To efficiently make use of information and services available in ubiquitous environments, mobile users need novel means for locating relevant content, where relevance has a user-specific definition. In the DYNAMOS project, we have investigated a hybrid approach that enhances context-aware service provisioning with peer-to-peer social functionalities. We have designed and implemented a system platform and application prototype running on smart phones to support this novel conception of service provisioning. To assess the feasibility of our approach in a real-world scenario, we conducted field trials in which the research subject was a community of recreational boaters.

Key words: service provisioning, context-awareness, recommender systems, mobile devices

1 Introduction

With their small size format and constant connectivity mobile devices such as smart phones and PDAs can offer interesting opportunities for novel services and applications in ubiquitous computing environments. On the other hand, given the huge amount of information and services potentially available in such

* Corresponding author. Tel.: +358 9 191 51296; fax: +358 9 191 51120
Email addresses: oriana.riva@hiit.fi (Oriana Riva), santtu.toivonen@vtt.fi (Santtu Toivonen).

environments, and the limitations anyway present in such devices (e.g., small-sized screens, limited processing power), the ways of searching content and discovering services are also evolving. Mobile users need to locate anytime, anywhere "relevant" content which is available in their daily environments, where relevance has a user-specific definition (e.g., cost, location, accessibility, etc.). According to the I-centric model proposed by the WWRF (Wireless World Research Forum), personalization, ambient awareness, and adaptability are the three core properties that set functional requirements on future service platforms (Arbanowski et al., 2004). For instance, a smart tourist guide should dynamically retrieve maps and information about shops, restaurants, and monuments located in the user's vicinity and depending on the current task, preferences, and social context.

Various research proposals seek to accomplish this user-centric paradigm in mobile environments. Context and context-awareness (Dey et al., 2001) allow services to dynamically adapt to their computing environment. Sensed context information permits inferring user's requirements and needs in order to make applications more personalized, friendly, selective, and responsive (Chen and Kotz, 2000). For example, mobility-aware recommenders such as the PILGRIM system (Brunato and Battiti, 2003) use the user's location to filter web links of interest. Context-based applications such as tourism information services (Cheverst et al., 2000) or remembrance agents (Rhodes, 1997) proactively retrieve content of interest based on the user's current task. Although these systems target different needs, they share a common design: the system collects different types of (contextual) information characterizing the user, and uses it to filter and rank relevant content items. However, this approach does not always reflect our daily life practice. 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 feedbacks upon system usage, or by observing her behavior over time. We argue that in many cases, systems sup-

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

porting 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 persons in the proximity.

Our second motivation originates from the observation that the success of systems like 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. In addition, the information space easily becomes static and out of date as updating information has 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 (Abowd, 1999; Espinoza et al., 2001).

Stemming from these considerations, 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 evaluate such an approach in real-world daily scenarios involving mobile users equipped with off-the-shelf mobile devices, such as smart phones.

This paper [3] describes the system platform together with an application prototype running on smart phones that we designed and implemented in the DYNAMOS project [4] to support our hybrid model of service provisioning. The DYNAMOS system allows 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 geographical locations, and eventually share them with other users; and *(iii)* to annotate official service descriptions with personal observations, comments, or ratings to be shared with others. To evaluate the feasibility and usefulness of the hybrid model, we conducted field trials where a community of recreational boaters used our application during a sailing regatta in the Helsinki region.

[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] A preliminary version of this paper appeared in (Riva and Toivonen, 2006)

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

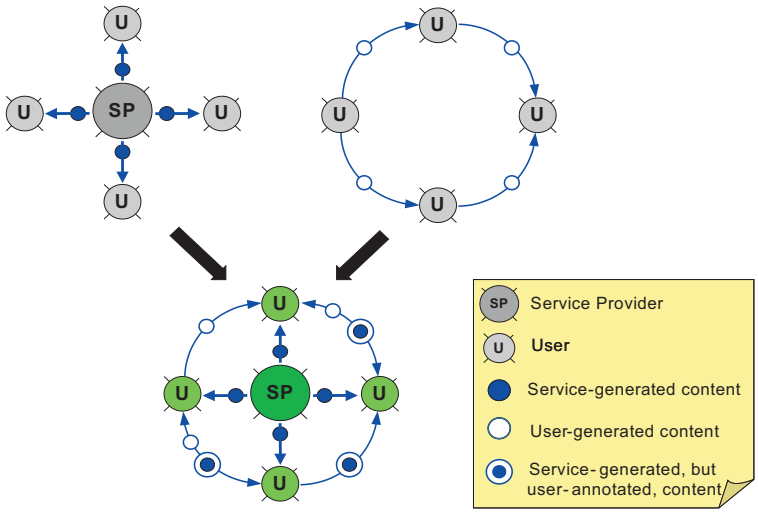

Fig. 1. Hybrid model of context-aware service provisioning

The rest of the paper is organized as follows. Section 2 describes the hybrid model of context-aware service provisioning. Section 3 gives insights on how such an approach has been accomplished by the DYNAMOS platform and Section 4 details its implementation. Section 5 describes the investigated use case and discusses results from the field trials we conducted. Section 6 reviews related work. The paper concludes in Section 7.

2 Hybrid Model of Context-aware Service Provisioning

As Fig. 1 depicts, our hybrid model combines the service provider to consumer (B2C) and the consumer to consumer (C2C) models. The resulting model allows users to receive official content describing available services (service-generated content) and to generate unofficial content such as observations and personal opinions (user-generated content), attach it to service descriptions (service-generated, but user-annotated, content), and eventually share it in a small/medium -sized community of users (e.g., among members of a sports club, friends, or colleagues). 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., Bellavista et al., 2003; Capra et al., 2003)). 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 its

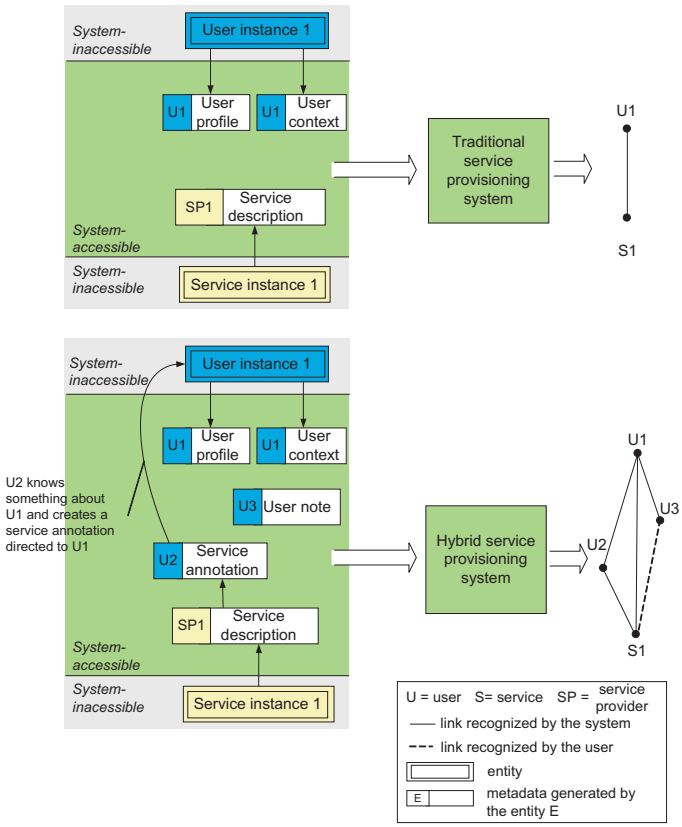

Fig. 2. Comparison of a traditional service provisioning model and the hybrid model

dynamic characterization. In other words, users can decide to disclose some aspects of their internal representation by specifying preferences, interests, and control policies into profiles, and by granting the access to context information such as location, activity, and social surrounding. Likewise, *service descriptions* represent metadata elements characterizing services. This information is 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` (i.e., provide the user `U1` with the service `S1`) upon matching metadata elements that are system-accessible. In this case, accessible metadata are user profile, user context, and service descriptions. These metadata are either explicitly submitted by users and service providers or implicitly learned by the system based on previous observations and inference algorithms. In our hybrid model, the matching involves a larger variety of metadata, which are provided also by other users. In addition to links entirely recognized by the system, our model allows also other peers to establish (implicitly or explicitly) links between services and users. As depicted in the bottom part of Fig. 2, two additional links are established.

- 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 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*.

- 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, etc. 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. Therefore, this information can further help users to take decisions about which service to access. 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 permit enriching the description of services with information not always supplied by official organizations or providers (such as personal experiences, description of the current status of the service or present state-of-affairs); and *(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).

3 DYNAMOS Platform

In the context of the DYNAMOS project, we built a system platform and application prototype designed for smart phones to support the functionalities of the hybrid model. This section gives insights on the design of such a system.

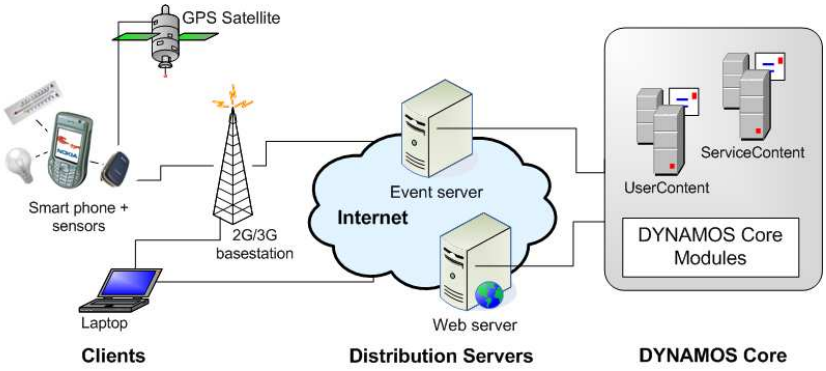

Fig. 3. DYNAMOS overall reference architecture

3.1 Overview

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

Mobile users can interact with the system using a common web browser or by installing and running an application on their smart phone. Accordingly, the communication occurs through two different types of Distribution Servers: Web Servers and Event Servers. Given the resource constraints of mobile phones, in terms of power, memory storage, screen size, etc., the application running on mobile phones needs to constantly adapt based on the client's execution context. Several types of sensors can be connected to the phone so that sensed context changes are constantly notified to the DYNAMOS Core (through the Event Server) and are used to accomplish context-aware service interactions. Moreover, as in any publish-subscribe system, Event Servers also allow clients (publishers) to publish into the system new user-generated content to be made available for usage to other active users (subscribers).

The Distribution Servers support the access to DYNAMOS Core. This consists of several DYNAMOS software modules and Content Servers. The modules composing DYNAMOS Core provide key functionalities for supporting context-based content provisioning in highly dynamic environments. The *User Content Server* stores user-related information (profile and context). The *Service Content 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. Or, the distribution of context information can also follow a distribution logic based on the type of stored context data, like done in (Großmann et al., 2005).

Fig. 4 shows in more detail the architectural components of DYNAMOS

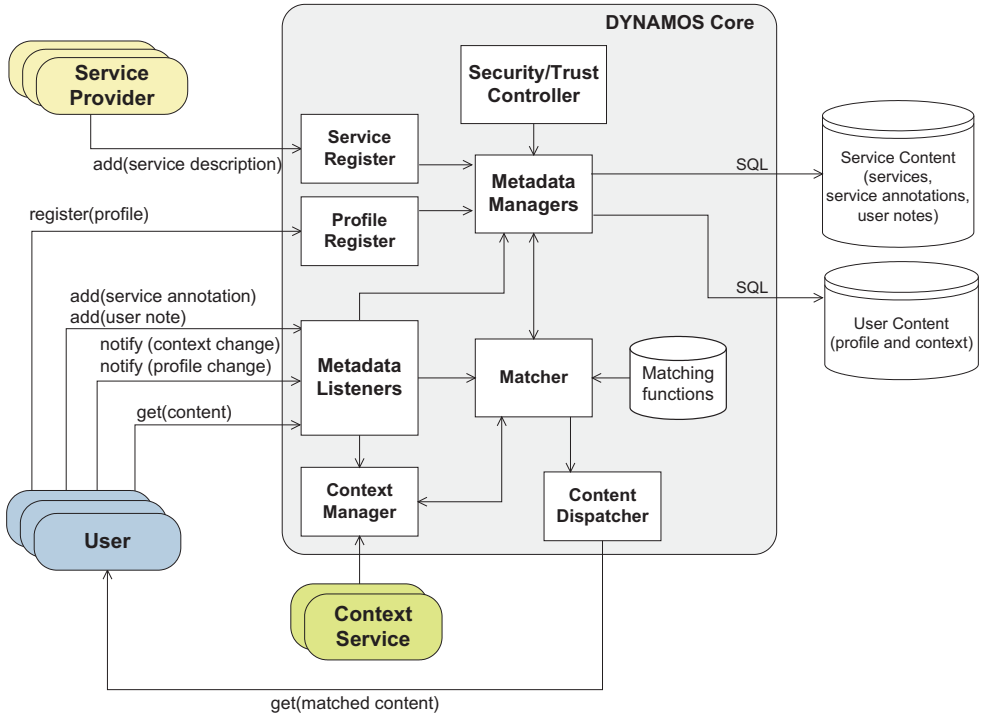

Fig. 4. DYNAMOS core architecture

Core. DYNAMOS Core provides the following functionalities: *Registration and Profiling*, *Context Management*, *Proactive and Reactive Service Provisioning*, *Content Matching*, and *Security and Trust*. In the following, we shed some light on how these functionalities have been accomplished.

3.2 Registration and Profiling

In order to get access to the system services, the user needs to register her profile through the *ProfileRegister*. In the profile, the user specifies personal information such as preferences, interests, and control policies (i.e., rules that control the system's behaviour). Profile updates can be communicated to the system anytime through the profile *MetadataListener*. The following example sketches the description of a common user profile along with typical content items provided by the system.

```
<Service rdf:about="SuomiForeca">
  <hasCategory rdf:resource="Weather"/>
</Service>

<Service rdf:about="KokarGuestHarbor">
  <hasCategory rdf:resource="GuestHarbor"/>
</Service>

<ServiceAnnotation rdf:about="Annotation220">
  <hasCreator rdf:resource="Setoivon"/>
  <refersTo rdf:resource="KokarGuestHarbor"/>
```

```
</ServiceAnnotation>

<UserNote rdf:about="Emergency389">
  <hasCategory rdf:resource="Emergency"/>
  <hasCreator>
    <Community rdf:about="TurkuSailingClub"/>
  </hasCreator>
</UserNote>

<Activity rdf:about="Sailing">
  <hasInterest>
    <ServiceCategory rdf:about="Weather"/>
  </hasInterest>
  <hasInterest>
    <ServiceCategory rdf:about="GuestHarbor"/>
  </hasInterest>
  <hasInterest>
    <UserNoteCreator rdf:about="TurkuSailingClub"/>
  </hasInterest>
</Activity>

<Activity rdf:about="Outdoor">
  <hasInterest>
    <ServiceCategory rdf:about="Restaurant"/>
  </hasInterest>
  <hasInterest>
    <UserNoteCategory rdf:about="Routine"/>
  </hasInterest>
</Activity>

<User rdf:about="Setoivon">
  <belongsToCommunity rdf:resource="TurkuSailingClub"/>
  <hasActivity rdf:resource="Sailing"/>
  <hasActivity rdf:resource="Outdoor"/>
</User>
```

The profile is specifically structured to support efficient matching between available service content and user. In their profile, users specify several types of activities and status, and associate multiple interests to each of them. Activity names are user-defined. Interest names are predefined based on the service taxonomy. In the example, the user `Setoivon` expresses his interest in `Weather` and `GuestHarbor` services as well as user notes provided by the `TurkuSailingClub`, when in `Sailing` activity; when in `Outdoor` activity, his interests are for `Restaurant` services and user notes of category `Routine`.

3.3 Context Management

While profile information represents the nearly static characterization of a user, context information represents the dynamic characterization. In Dey's terms (Dey et al., 2001), context consists of any information that can be used to characterize the situation of an entity. Specifically, context consists of spatial information (location, speed, orientation), temporal information (time, duration), user activity (physical activity, social surroundings), environmental information (temperature, light, noise), and resource availability (nearby devices, device status).

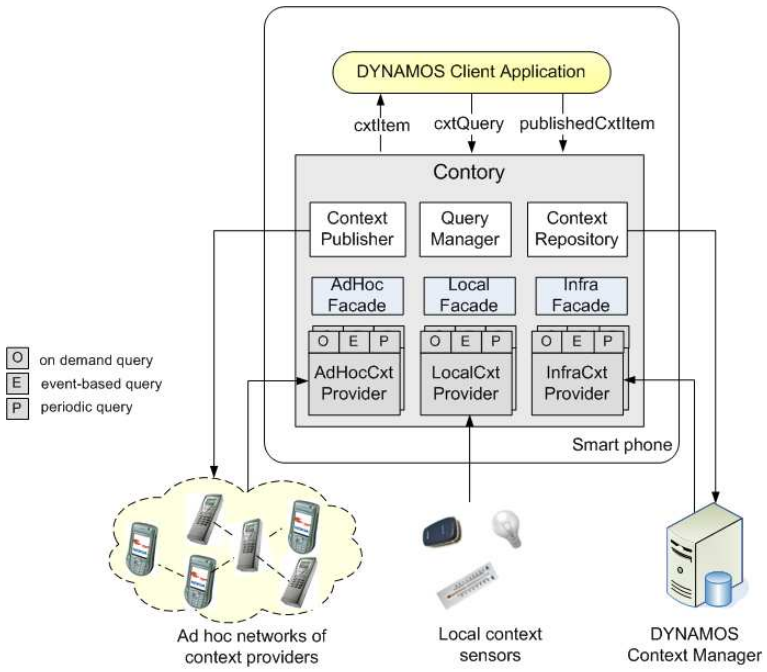

Fig. 5. Contory architecture

Context provisioning on smart phones is achieved using Contory (Riva, 2006). Contory is a middleware specifically designed to provide alternative mechanisms of context provisioning on resource-constrained devices such as smart phones. As shown in Fig. 5, three context provisioning strategies are available: internal sensor-based provisioning (through LocalCxtProvider and LocalFacade modules), external infrastructure-based provisioning (through InfraCxtProvider and InfraFacade modules), and distributed context provisioning in ad hoc networks (through AdHocCxtProvider and AdHocFacade modules). 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 an external context infrastructure such as the DYNAMOS *ContextManager* and *ContextServices*; or *(iii)* exploit neighboring devices willing to share their context information. The other architectural modules perform query management, provide means to remotely store context data, and allow publishing context data in ad hoc networks.

Contory offers an SQL-like interface to generate context queries, in which applications can specify type and quality of the desired context items, context sources, push or pull mode of interaction, and other properties. The query template has the following format:

```
SELECT <context name>
FROM <source>
WHERE <predicate clause>
FRESHNESS <time>
DURATION <duration>
EVERY <time> | EVENT <predicate clause>
```

SELECT specifies the type of the requested context items. DURATION specifies the query lifetime as time (e.g, 1 hour) or as the number of samples that must be collected in each round (e.g., 50 samples). The FROM clause can either be unspecified thus allowing the middleware to autonomously and dynamically select the context provisioning mechanism to be employed. Alternatively, the FROM clause specifies which context provisioning mechanism to employ: internal sensor-based (intSensor), external infrastructure-based (extInfra), and distributed context provisioning in ad hoc networks (adHocNetwork). In the case of adHocNetwork provisioning, the FROM clause also tells multiplicity (numNodes) and distance (numHops) of the context source nodes. WHERE permits specifying additional filters to apply to the selection of context items (e.g., accuracy, correctness, precision, etc.). FRESHNESS specifies how recent the context data must be. Finally, long running queries are expressed by means of EVERY and EVENT clauses. The EVERY clause allows the application to specify the rate at which context data should be collected (periodic query). The EVENT clause determines the set of conditions that must be met at the context provider's node before a new result is returned (event-based query).

In the example below, the client application specifies a query requesting, for one hour and every 10 seconds, wind data collected from at least one node at maximum distance of 2 hops; data must not be older than 50 seconds and be accurate.

```
<query name="wind20648"
    select="wind"
    <struct name="from">
        <String name="typeSource">"adHocNetwork"</String>
        <int name="numNodes">1</int>
        <int name="numHops">2</int>
    </struct>
    <struct name="where">
        <boolean name="accurate">true</boolean>
    </struct>
    freshness=50
    duration=3600
    every=10
</query>
```

Along with the simplicity of its programming interface, Contory provides more flexibility and reliability to context provisioning. For example, temporary disconnections of GPS devices from the phone can be compensated by relying on other neighboring devices. Moreover, the context sensing range of the application can be extended to any geographical region and service. For example, weather information collected by a network of local devices in a certain region is stored in the remote infrastructure and later accessed by remote users interested in getting information about such a region.

The possibility of gathering different types of information from a network of mobile devices in the same proximity can provide useful support in terms of service provisioning, e.g., to find out more about dynamic service informa-

tion. For example, consider a user who wants to know about the status of a restaurant service in terms of free parking places in the vicinity, number of free tables, and estimated waiting time. An ad hoc network of mobile phones owned by persons who are already in the restaurant can be used to infer how many empty places are left and how long those people have already spent in the restaurant. An ad hoc network of parking meters and cars outside the restaurant can compute where free parking lots are available at the moment. This kind of information can be collected and processed by intermediate stations, like the restaurant in this case, and then be transferred to the DYNAMOS repositories.

Context data collected on the mobile phone are communicated to the DYNAMOS system through the context *MetadataListener*. The *ContextManager* is then responsible for processing received context data and making them available for usage to the *Matcher*. In order to infer higher-level context information, several reasoning mechanisms can be supported by the *ContextManager*. Additionally, the *ContextManager* can rely on external *ContextServices* (e.g., weather stations, office infrastructure) to gather additional context information.

3.4 Proactive and Reactive Service Provisioning

Along with profile and context, service descriptions, service annotations, and user notes are the service metadata at the basis of the hybrid model. *ServiceRegister* and *MetadataListeners* allow the specification and update of metadata content. Anytime a *Listener* receives a new notification, it invokes the corresponding *MetadataManager*. The *MetadataManager* processes the received information and stores it in the appropriate *ContentServer*. User and service content is stored respectively into the *UserContent* and *ServiceContent* relational databases. *MetadataManagers* are responsible for coordinating the access to these databases and for translating database queries into SQL format, and vice versa. Additionally, 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.

The user can interact with the system in a proactive or reactive manner. In the proactive mode, the client is constantly provided with content of interest based on the associated profile and context information. Every time a notification is received by a *MetadataManager*, this invokes the *Matcher* module, which in turns can use the *ContentDispatcher* to communicate new content to the client. Alternatively, the client can submit on-demand queries to the *MetadataListener*, for example to receive more information about a specific service item (e.g., to retrieve all service annotations attached to a certain

service description) or about a certain region (e.g., to be informed with the weather conditions in proximity of a certain service).

3.5 Content Matching

The *Matcher* is responsible for matching available service content and user's profile/context in order to extract a list of relevant items to pass to the *ContentDispatcher*. When necessary, the *MetadataManagers* or *MetadataListeners* invoke the *Matcher* module. For example, if a notification reports changes to the context of active users, it is necessary to invoke the *Matcher* in order to verify whether new matched content has be sent. If a notification carries information about non-matched services, no matching needs to be triggered, but such an information must be stored in the repositories. The *Matcher* accomplishes its task by performing category-based and context-based matching.

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

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

If the service content is organized in a hierarchy, such as a service taxonomy, the Matcher can recognize also close matches. A way to acknowledge close matches is to resort the relationships in the taxonomy, and use various algorithms for recognizing these relationships. Examples of such can be found in (Stojanovic et al., 2001; Corby et al., 2006). As an example, consider the functioning of the Object Match (Stojanovic et al., 2001) algorithm. It recognizes the paths from two concepts in a hierarchy to a "root" concept, and calculates the match as a ratio between the number of shared concepts found in the paths (intersection), and the total number of concepts after combining both paths (union). Suppose a service taxonomy consisting of the following seven concepts: `Service`, `Restaurant`, `ItalianRestaurant`, `Trattoria`, `Pizzeria`, `JapaneseRestaurant`, and `SushiBar`. The relationships between these are so that `Restaurant` is a subcategory of `Service`, `ItalianRestaurant` and `JapaneseRestaurant` are subcategories of `Restaurant`, `SushiBar` is a subcategory of `JapaneseRestaurant`, and finally `Trattoria` and `Pizzeria` are

subcategories of `ItalianRestaurant`. The user in this case has expressed direct interest in Pizzerias. The object match algorithm assigns the relevance of $\frac{3}{5}$ to a service identified as a `Trattoria`, whereas a `SushiBar` would receive the relevance of $\frac{2}{6} = \frac{1}{3}$. Restaurants described on a more general level would receive the following relevance values: `ItalianRestaurant` would receive the relevance of $\frac{3}{4}$ and `JapaneseRestaurant` the relevance of $\frac{2}{5}$.

Once category-based matching has been performed, context-based matching assigns to each service s_i^* a matching rank value $r_{s_i^*}$. This is calculated by matching every available user's context information $c_k \in C$ and the service description s_i^*. Specialized filters $f_x(c_x, s)$ are deployed for each type of context information. For example, `Close(location,s)` uses the location to filter nearby services, `Open(time,s)` uses the time to filter open services, and `MatchWeather(weather,s)` establishes if a certain service is compliant to the weather conditions (e.g., indoor services if it rains). Moreover, to each type of context information $c_k \in C$ a different weight w_k is associated, so that more relevant context information can have a major impact on the final matching rank. Formally:

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

Therefore, matched service items are ordered and further filtered based on the computed r_s values and then delivered to the user. The weights w_k can be statically specified by the user in her profile or dynamically computed by the system. The second solution provides more flexibility and efficiency. For example, if there is plenty of content available in the near surroundings, more weight can be given to the results of `Open` and `MatchWeather` functions. On the other hand, in a remote area with few content items available, location-based matching becomes more relevant. One way to infer user's w_k without relying on explicit user's feedbacks is to monitor the type of content the user produces. It is likely that the user produces service annotations for services whose category and characteristics are of more interest to her.

Service annotations available to anybody (that is, of public use) are filtered according to the matching result of the associated service items. As far as user note items, the matching process consists only of category-based matching (like for service items), time-based filtering, and location-based filtering. Service annotations directed to single recipients or a community and user notes are then filtered based on the control policies specified in the user profile (this will be discussed in the following section).

The *Matcher* module is designed as a *Strategy* pattern (Gamma et al., 1995). Different matching algorithms are appropriate at different times and for different types of content. Hence, it is necessary to easily interchange these match-

ing algorithms. The system can perform exact-match based on simple boolean logic or best-match (i.e., provides a ranked list of content items based on relevance and close matches). 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 thus rendering not applicable all f_x matching functions.

3.6 Security and Trust

The risk of attacks from malicious users, spamming, and trust are crucial issues in our system (Wang et al., 2002; Dimmock et al., 2005). 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 address these and other security issues, we decided to partly rely on a central authority which controls and authenticates the content of exchange and partly deploy both autonomous and collaborative decision-making mechanisms. Therefore, we utilized both infrastructure-specific and client-specific control mechanisms.

Infrastructure-specific control mechanisms ensure authentication and content integrity. Authentication aims to verify the identity of the content originator. This can be carried out outside the DYNAMOS domain. If a public key infrastructure (PKI) exists, DYNAMOS users can sign messages using their private keys. The *Security/Trust Controller* can then verify the identity of the originator by checking the digital signature attached to the message. Digital signatures also enable content integrity. The reception of a signed message indicates that the content has not been changed since it is signed and that the message originated from that source. For other security issues, we assume that users trust the DYNAMOS system to maintain information confidentiality and user anonymity.

Client-specific control mechanisms utilize autonomous and collaboration-based control policies. Individual clients specify autonomous control policies based on user preferences and needs. These policies are directly enforced by the *ContentDispatcher*. Control policies express conditions about which type of content the user wishes to receive, from which type of content providers, and at which frequency. For example, a user can request to be informed about emergency warnings anytime, but about leisure happenings only when in free time. 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. Resolution of conflicts among different control policies can be

performed partly statically and partly dynamically. The dynamic resolution selects the action that satisfies the largest number of conditions.

To accomplish collaboration-based control and increase opportunities for user collaboration, we offer the user the possibility of specifying buddy-lists and communities (Singh et al., 2001). To each person present in the personal buddy-list, a trust grade is assigned. The user can then also decide to group persons of her buddy-list in different communities and assign different filters and priorities to each community. For example, content coming from some communities is blocked in some situations, while higher priority is given to other communities. In addition, personal details of the content's creator can be combined with the details of the community he is representing.

An extensive model capturing several aspects which influence trust establishment can be incorporated to our hybrid model. Trust is explained in terms of a relationship between a trustor, who trusts a certain entity, and a trustee, the trusted entity (Grandison and Sloman, 2000). In our case, based on his trust in the trustee, a trustor can decide whether to accept or refuse content provided by the trustee. (Toivonen et al., 2006) gives a formalized description of the trust model we adopt. The basic idea of such a model is that trust is established by combining profile, context aspects, reputation information, and recommendations. Reputation indicates one's own opinion of the content creator in question, and is accumulated by each interaction instance with it. Reputation introduces a history-dependent dimension to the trust evaluation. Recommendation is communicated reputation information, meaning that it is the reputation of the same content creator, but as observed by someone else. Formally:

$$trust_{A,\sigma} : Quality \times Context \times TValues \times 2^{Tvalues} \rightarrow TValues \qquad (3)$$

The function $trust_{A,\sigma}$ returns a measure $m \in TValues$ of A's trust in relation to a target σ. $TValues$ can be a set of binary values (e.g., trusted, not trusted) or discrete (e.g., strong/weak trust, strong/weak distrust). In the function's inputs, the trustee is described using: *(i)* a set of quality attributes of the trustee, such as profile and any other known information about the trustee; *(ii)* a set of context aspects that concern the trustor, the trustee, and their interaction; *(iii)* a trust value which represents the trustee's reputation in the viewpoint of A; and *(iv)* a set of trust values which represent recommendations about the trustee in the viewpoints of some recommenders. Note that the adoption of some context information in this case can help adapt the trustworthiness evaluation depending on the situation. For example, a positive recommendation might be directed to someone's role as a professional in a certain domain, but might be of low value in another domain.

The model can be further complicated by weighting recommendations about

the trustees (i.e., the $TValues$) expressed by other recommenders based on the trust of A on those recommenders. For each recommender j, the trustor A locally stores a (c_j, w_j) pair. c_j is the number of times the recommender j has given a correct trust value to a certain item and w_j is the number of times the trust evaluation was wrong. The probability that j gives a correct trust evaluation to a certain item is then given by $\frac{c_j}{c_j+w_j}$.

It can also be the case that no prior knowledge about the content creator is available, nor any (previously known) person giving recommendations (Toivonen et al., 2006). In these cases, social or professional networks can be utilized. One well-known solution in the literature is to ask for recommendations. Alternatively, similarities between entities or contexts can be exploited. For example, if the current content creator has no reputation in the current context, he might have it in similar enough conditions and that could be used. Or, if he is completely unknown, maybe some other entity like him was previously known and information characterizing him can be utilized. The same approach can be applied also to unknown recommenders. In addition, the "distance to the unknown recommender can be considered. By distance we mean the number of links in the network of entities, either weighted or not. Weighting indicates that more emphasis is put to a kind of link that is relevant with regard to the current task. For example, when seeking for advices about guest harbors, more emphasis is given to links related to boating than to kinship. The benefit of having a trust grade assigned to every (known or unknown) content originator permits ordering received content items based on the trust grade of the sender.

Cases of spam occur when in the queue of items to be delivered to a certain user there are many unwanted items. This means that there is an active filter or profile rule or context status that matches many unwanted items, but that it is also supposed to match interesting items. To prevent these situations from occurring, the client can specify more specific filters, can use spam detector filters, and can maintain blacklists. However, client-specific rule-based mechanisms cannot entirely solve spam problems, even when advanced mechanisms based on Bayesian inference are employed (Dimmock et al., 2005). Conversely, a collaborative network of peers can be used to detect spam. The method proposed in (Dimmock and Maddison, 2004) to detect spam in incoming e-mail can be extended to recommender systems. To identify potential creators of spam items, each user locally maintains triplets in the form (u_i, m_s, m_t): for each content originator u_i, m_s indicates how many spam items such a user sent, while m_t represents the total number of items received by the user in a certain time interval. When m_s/m_t exceeds a certain threshold, u_i is added to a blacklist and a warning is sent to the DYNAMOS system.

4 Implementation

The DYNAMOS system and the prototype application on mobile phones have been implemented in Java. Event-based communication is realized through the Fuego Core Event Server (Tarkoma et al., 2006). The Fuego middleware is implemented in Java and provides a scalable distributed event framework and XML-based messaging services. This middleware also runs on mobile phones supporting Java MIDP 1.0. 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 running Symbian OS. Symbian OS is the de facto standard operating system for smartphones. It has a lightweight 32-bit pre-emptive kernel that is based on a hybrid design combining characteristics from both micro-kernel and monolithic kernel architectures. Symbian OS design focuses on open mobile phones and thus is optimized for efficient resource constrained operation.

Currently, the whole DYNAMOS application's jar size is 307KB (the size of Contory and the Fuego Core middleware architectures is 257KB). The application has been tested on Nokia 6630 (Symbian OS 8.0a, 220 MHz processor, WCDMA/EDGE, 9 MB of RAM), Nokia 6680 (Symbian OS 8.0a, 220 MHz processor, WCDMA/EDGE, 9 MB of RAM), and Nokia 7610 (Symbian OS 7.0s, 123 MHz processor, GPRS, 9 MB of RAM) phones.

The mobile phone platform was selected since it currently represents a relatively powerful computing platform which is already widely carried by many people. Smart phones offer already the functionalities of a music player or a digital camera and this permitted supporting not only textual, but also multimedia content. Disadvantages of this platform are limited debugging support, limited programming environment, slow storage access, and not yet available sensor API[5] to manage and control sensors connected to the phone. Nevertheless, it has proven very difficult experimenting with real-world sensors that can be connected to the phone. Despite common BT-GPS devices, there is a scarce availability of (price-contained) sensors that can be connected to the mobile phone for gathering context information in real-time (many available sensors can record context data and produce logs for later analysis).

[5] The JSR 256 ("Mobile Sensor API") will be available in the near future: http://jcp.org/en/jsr/detail?id=256.

5 Case Study: 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. We chose this scenario 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. Such an application prototype was used during a sailing regatta in the Helsinki region.

5.1 Requirements Study

In carrying out the requirements study, we followed a process of semi-structured interviews. 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 [6] . The motivation was to learn about their habits and activities while boating and thus identifying crucial requirements for scoping the development of the application. In addition, as it is typically the case with qualitative research (Shadish et al., 2002), the interviews aimed at discovering and exploring possible explanations for their habits.

The interviews were carried out in a one-to-one fashion between the interviewer and the interviewee. The applied research method was ethnographic and qualitative. The interviewer asked open questions and wrote the answers down. With the interviewee's permission the session was also recorded in order to aid further analysis. One interview lasted approximately 50 minutes. After

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

some general questions (i.e., age, occupation, hobbies), the interviewees were asked 30 general questions related to their boating habits and conventions, and also their opinion 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%). The interviewees were explicitly asked about "boating intensiveness", and whether it has an impact on getting in touch with someone while boating or not. The boating intensiveness was defined as a phenomenon that varies based on certain factors, such as weather conditions and traffic intensity in the current route. 95% of the interviewees indicated that in poor conditions (such as stormy weather or a narrow and heavily trafficked route) it is better to postpone the communication.

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 [7] ; 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.

5.2 Design Principles

Based on the requirements study, we derived the following design principles to guide the development of the prototype application:

- *always-on, always running service*: to be considered useful and updated, the users must have the impression that the system is always online.
- *present status of services*: in some situations, users are mostly interested in knowing about the current status of services such as special happenings, waiting times, parking availability, weather information, traffic conditions, etc.
- *personalization*: users must have means for personalizing their interaction with the system; preferences are saved in the user profile, and the most

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

important settings must be accessible also through the phone application so that the user can anytime change mode of interaction.
- *control access*: avoid privacy concerns by disclosing only information that the user agrees to share and let the user control how and for how long the content he generated is shared with other peers.
- *trust and social collaboration*: the support for buddy-lists and communities can greatly improve the trust the user has in the application; furthermore they help stimulate collaboration, generation of new content, and sharing of information.
- *friendly and easy user interface*: given that users are typically involved in other activities while using such a kind of applications, it is necessary to provide means for quick generation of new content; moreover, open connections with external sensors such as GPS devices should be totally transparent to the user.

5.3 Application Prototype

The main functionality of the sailing prototype application is to proactively inform users about services of interest in the surrounding environment. A *service description* object contains the category to which the service belongs, a textual description of the service characteristics, its location, opening times, and contact information. Upon receiving a service description of an available service the user can take different actions. The most evident usage is to use the suggested service. Service descriptions often offer hyperlinks to some further material characterizing the service in question or to additional functionalities. For example, a restaurant service description can contain a link to allow booking a table.

The user can also annotate received service descriptions and share them with other users who might be interested in those services. A *service annotation* 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 to type on a small keyboard, especially if this needs to be done while involved in other activities. An annotation also contains a hyperlink to the service to which it refers. 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 ser-

Fig. 6. WeatherWatcher screenshots

vice annotation in order to guarantee temporal relevance of the information. Service annotations can range from very light additions to very rich comments on the service. The light annotations can sometimes be automatically generated without the active input by the user. For example, if a user decides to forward an interesting service description to her friend, she can do so without any active annotating. Still, when sending the description, it is annotated for example with the fact that it is she who sent it. And when her friend receives it, she will notice that the service description did not come straight from the service provider, but from a friend instead.

Along with service annotations, the user can generate *user notes*, which are similar to annotations with the main difference that they do not contain links to any service. User notes have different priorities and belong to different categories that can be further specialized based on the scenario of use. For example, in a sailing scenario, user notes can be routine messages, safety warnings, and emergency alarms.

Most of the interactions with the system occur in a proactive manner. However, on-demand interactions are also supported. One example is the *WeatherWatcher* (see Fig. 6). This service allows users to retrieve weather information in a certain geographical region (specified in terms of latitude and longitude). In a sailing scenario, where weather conditions can vary rather quickly, this represents an important element to select the sailing route. Weather information consists of temperature, wind, speed, humidity, and atmospheric pressure.

To summarize, Fig. 7 shows an example illustrating the main functions supported by our prototype application. 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 can also use the WeatherWatcher to know about weather conditions in that region. Finally, he decides to follow the advice of his friend and try the place out. Once he has visited the place, he can also annotate the service description and share it with other friends.

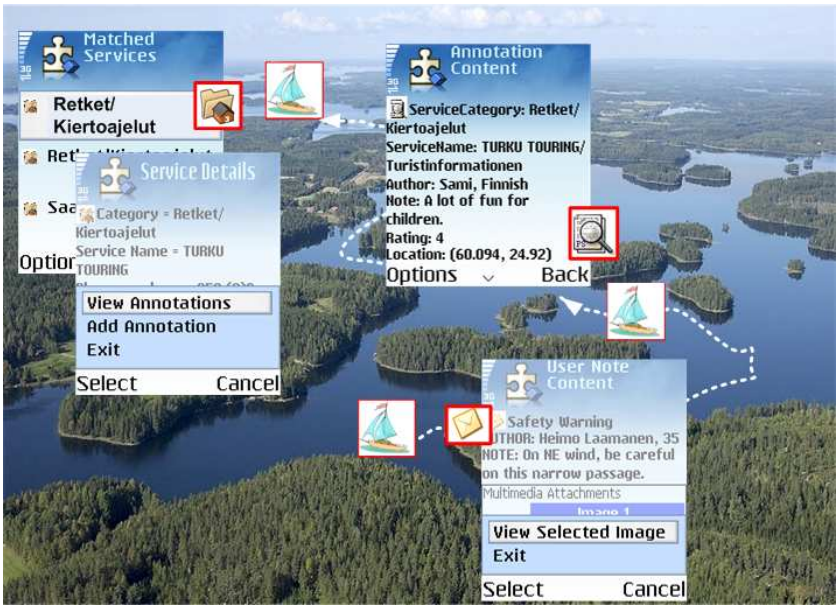

Fig. 7. Example of sailing scenario and screenshots of the phone interface

5.4 Field Trials Evaluation

Two 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 [8] . In June and August 2005, we conducted field trials. A first preliminary trial consisted of a short excursion on sailboat with 4 users. The second trial 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 application prototype employed during the trials did not include context management and did not support the WeatherWatcher service, which were developed during the second phase of software development. Location-awareness was simply achieved by connecting GPS devices via Bluetooth to the phone. In the trails,

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

Nokia 6630 phones and Bluetooth GPS Receivers InsSirf III [9] were used.

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 (FORUM NOKIA, 2005)), 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 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. 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

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

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 and 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 daily situations and they would be ready to pay something to use it. The risk of spam as well as untrustworthy recommendations were the main threats and disadvantages observed about the system.

6 Related Work

Our research shares some concepts and techniques typically 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 users in evaluating items (Resnick and Varian, 1997; Schafer et al., 2002).

The two most applied recommendation techniques are the content-based approach (Mostafa et al., 1997; Jian et al., 2005) and the collaborative filtering approach (Resnick et al., 1994; Shardanand and Maes, 1995; Terveen et al., 1997; Breese et al., 1998; Herlocker et al., 1999). In content-based recommendation, the system suggests items that best match a specified criteria. The system has to understand the main features describing the items of interest in order to establish their relevance, and it usually maintains domain-specific user profiles. Collaborative filtering aims to find a neighborhood for a target user through other users who share the same tastes. 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. Pearson correlation (Resnick et al., 1994) is often used to calculate the similarity. Hybrid systems (Balabanovic; and Shoham, 1997; Rucker and Polanco, 1997; Good et al., 1999; Shahabi et al., 2001; Burke, 2002) combine collaborative filtering, content-based querying and other recommendation techniques to achieve higher accuracy.

In recent years, some attempts have also moved towards the integration of context-awareness in recommenders. In (Brunato and Battiti, 2003), the authors employ a historical database of user locations and URLs to determine where and how often certain links have been accessed and use this information to provide accurate recommendations. In the recommender system described in (Yap et al., 2005), 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. (Adomavicius et al., 2005) presents a multidimensional approach to provide recommendations based on several types of context information and accomplish hierarchical aggregation of recommendations.

Compared to this kind of recommender systems, our approach goes beyond by

combining the ranking process of recommenders with peer-to-peer social functionalities. Moreover, the most consistent part of research on recommender systems has traditionally been applied to relatively static scenarios and to Internet communities, while the objective of our study was a community of mobile users in highly dynamic mobile environments. Also from a technical point of view, with the exception of some recent example (Yu et al., 2006), recommender systems have never been implemented on smart phone platforms.

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), and location-sensitive games. For example, GUIDE [10] is a context-sensitive tourist guide for visitors in the city of Lancaster; WebPark [11] provides location-based services in protected and recreational areas, such as coastal, rural, and mountainous regions; LoVEUS [12] aims to provide users with location- and orientation-sensitive multimedia information; and EurEauWeb [13] seeks to increase the commercial viability of Europe's waterways.

However, except for few exceptions, most of the context-aware applications so far implemented are based on technologies that are not supported by common smart phones, or they assume more resources than those that smart phones can offer. Most of the existing prototypes run on laptops and PDAs, while in our work, all the development was done using a real mobile phone platform. The ContextPhone (Raento et al., 2005) is an open-source prototyping platform built on the Nokia Series 60 phones platform. It can be used to sense, process, store, and transfer context data. The blackboard-based framework of Korpipää (Korpipää, 2005) implements a ContextManager which provides a publish-subscribe mechanism and a database for context data on mobile devices. However, in both works the context sensing relies on information locally available on the device or through BT sensors, whereas Contory provides flexible access to various types of internal and external sensors. In addition, Contory permits exploiting not only location information, but a wider range of contextual information, to accomplish more diversified content matching.

Finally, as far as mobile technologies for tourism, (Brown and Chalmers, 2003) 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

[10] GUIDE Project. URL: http://www.guide.lancs.ac.uk
[11] WebPark Project. URL: http://www.webparkservices.info
[12] LoVEUS Project. URL: http://loveus.intranet.gr/
[13] EurEauWeb Project. URL: http://www.softeco.it/softeco/en/eureauweb1.html

reuses past research experiences on attaching signs or leaving marks to visited environments as done in the tour of Disneyland application of Pascoe (Pascoe, 1997), GeoNotes (Espinoza et al., 2001), E-graffiti (Burrell and Gay, 2001), and virtual graffiti (Järvensivu et al., 2004). The possibility to attach content of different types to contextual situations permits storing content for future usage by the user herself, friends, or anyone else.

7 Conclusions

The hybrid model of context-aware service provisioning represents a novel approach to access and share information about services and content available in our daily environments. We designed and implemented a platform along with a prototype application running on smart phones, which support this model. We evaluated the feasibility of our approach through field trials in which the research subject was a community of recreational boaters. Our model has proved so far to be an interesting and useful approach to support 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 integrate similar functionalities in systems for traffic information, nature trail, journey planner, sport games, cultural events, conferences, and travel guides. Additionally, the field trials allowed us to identify and solve many technical problems of location detection, context provisioning, network connectivity, and general system performance which will help us improve future application development on mobile phones.

Acknowledgments

The authors would like to thank Tekes, ICT-Turku, Suunto, TeliaSonera, and VTT for funding the DYNAMOS Project. Thanks also to the Research and Training Foundation of TeliaSonera.

References

Abowd, G., 1999. Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment. IBM System Journal 38 (4), 508–530.

Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A., 2005. Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. ACM Trans. Inf. Syst. 23 (1), 103–145.

Arbanowski, S., Ballon, P., David, K., Droegehorn, O., Eertink, H., Kellerer, W., van Kranenburg, H., Raatikainen, K., Popescu-Zeletin, R., 2004. I-centric Communications: Personalization, Ambient Awareness, and Adaptability for Future Mobile Services. IEEE Communications Magazine 42 (9), 63–69.

Balabanovic;, M., Shoham, Y., 1997. FAB: Content-Based, Collaborative Recommendation. Commununication of ACM 40 (3), 66–72.

Bellavista, P., Corradi, A., Montanari, R., Stefanelli, C., 2003. Context-Aware Middleware for Resource Management in the Wireless Internet. IEEE Transactions on Software Engineering 29 (12), 1086–1099.

Breese, J., Heckerman, D., Kadie, C., 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98). pp. 43–52.

Brown, B., Chalmers, M., 2003. Tourism and Mobile Technology. In: K. Kuutti, E. H. K. e. a. (Ed.), Proceedings of the 8th European Conference on Computer Supported Cooperative Work (ECSCW'03). Kluwer Academic Publishers, pp. 335–355.

Brunato, M., Battiti, R., 2003. PILGRIM: A Location Broker and Mobility-Aware Recommendation System. In: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03). IEEE Computer Society, Fort Worth, Texas, USA, pp. 265–272.

Burke, R., 2002. Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction 12 (4), 331–370.

Burrell, J., Gay, G., 2001. E-Graffiti: Evaluating Real-World Use of a Context-aware System. Interacting With Computers: Special Issue on Universal Usability 14, 301–312.

Capra, L., Emmerich, W., Mascolo, C., 2003. CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications. IEEE Transactions on Software Engineering 29 (10), 929–945.

Chen, G., Kotz, D., 2000. A Survey of Context-Aware Mobile Computing Research. Tech. rep., Hanover, NH, USA.

Cheverst, K., Davies, N., Mitchell, K., Friday, A., Efstratiou, C., 2000. Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences. In: Proceedings of the SIGCHI conference on Human factors in computing systems (CHI'00). ACM Press, New York, USA, pp. 17–24.

Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., Gandon, F., 2006. Searching the Semantic Web: Approximate Query Processing Based on Ontologies. IEEE Intelligent Systems 21 (1), 20–27.

Dey, A. K., Salber, D., Abowd, G., 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction 16 (2-4), 97–166.

Dimmock, N., Bacon, J., Ingram, D., Moody, K., 2005. Risk Models for Trust-

Based Access Control (TBAC). In: Proceedings of the 3rd International Conference on Trust Management (iTrust'05). pp. 364–371.

Dimmock, N., Maddison, I., 2004. Peer-to-peer Collaborative Spam Detection. ACM Crossroads 11 (2), 4–4.

Espinoza, F., Persson, P., Sandin, A., Nyström, H., Cacciatore, E., Bylund, M., 2001. GeoNotes: Social and Navigational Aspects of Location-Based Information Systems. In: Proceedings of the 3rd International Conference on Ubiquitous Computing (Ubicomp'01). Springer-Verlag, pp. 2–17.

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

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

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J., Riedl, J., 1999. Combining Collaborative Filtering with Personal Agents for Better Recommendations. In: Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Conference on Innovative Applications of Artificial Intelligence (AAAI'99/IAAI'99). American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 439–446.

Grandison, T., Sloman, M., 2000. A Survey of Trust in Internet Applications. IEEE Communications Surveys and Tutorials 3 (4).

Großmann, M., Bauer, M., Hönle, N., Kappeler, U.-P., Nicklas, D., Schwarz, T., 2005. Efficiently Managing Context Information for Large-Scale Scenarios. In: Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom'05). pp. 331–340.

Herlocker, J. L., Konstan, J. A., Borchers, A., Riedl, J., 1999. An Algorithmic Framework for Performing Collaborative Filtering. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'99). ACM Press, New York, NY, USA, pp. 230–237.

Järvensivu, R., Pitkänen, R., Mikkonen, T., 2004. Object-oriented middleware for location-aware systems. In: Proceedings of the 2004 ACM Symposium on Applied Computing (SAC'04). pp. 1184–1190.

Jian, C., Jian, Y., Jin, H., 2005. Automatic Content-Based Recommendation in e-Commerce. In: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service. IEEE Computer Society, Washington, DC, USA, pp. 748–753.

Korpipää, P., 2005. Blackboard-based Software Framework and Tool for Mobile Device Context Awareness. PhD Thesis. VTT Publications: 579, VTT Electronics, Espoo, http://www.vtt.fi/inf/pdf/publications/2005/P579.pdf.

Mostafa, J., Mukhopadhyay, S., Palakal, M., Lam, W., 1997. A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation. ACM Transactions on Information Systems 15 (4), 368–399.

Pascoe, J., 1997. The Stick-e Note Architecture: Extending the Interface Be-

yond the User. In: Proceedings of the 1997 International Conference on Intelligent User Interfaces. ACM Press, pp. 261–264.

Raento, M., Oulasvirta, A., Petit, R., Toivonen, H., 2005. ContextPhone: a Prototyping Platform for Context-aware Mobile Applications. IEEE Pervasive Computing 4 (2).

Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., Riedl, J., 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proceedings of the ACM Conference on Computer Supported Collaborative Work. Chapel Hill, North Carolina, USA, pp. 175–186.

Resnick, P., Varian, H. R., 1997. Recommender Systems. Commun. ACM 40 (3), 56–58.

Rhodes, B. J., 1997. The Wearable Remembrance Agent: A System for Augmented Memory. In: Proceedings of the 1st IEEE International Symposium on Wearable Computers (ISWC'97). pp. 123–128.

Riva, O., 2006. Contory: A Middleware for the Provisioning of Context Information on Smart Phones. In: Proceedings of the 7th ACM International Middleware Conference (Middleware'06). Vol. 4290 of LNCS. Springer, pp. 219–239.

Riva, O., Toivonen, S., 2006. A Model of Hybrid Service Provisioning Implemented on Smart Phones. In: Proceedings of the 3rd IEEE International Conference on Pervasive Services (ICPS'06). IEEE Computer Society, pp. 47–56.

Rucker, J., Polanco, M. J., 1997. SiteSeer: Personalized Navigation for the Web. Commununication of ACM 40 (3), 73–76.

Schafer, J. B., Konstan, J. A., Riedl, J., 2002. Meta-Recommendation Systems: User-Controlled Integration of Diverse Recommendations. In: Proceedings of the 11th international conference on Information and knowledge management (CIKM'02). ACM Press, New York, NY, USA, pp. 43–51.

Shadish, W. R., Cook, T. D., Campbell, D. T., 2002. Experimental and Quasi-Experimental Designs for Generalized Causal Inference. Houghton Mifflin, New York.

Shahabi, C., Kashani, F. B., Chen, Y.-S., McLeod, D., 2001. Yoda: An Accurate and Scalable Web-Based Recommendation System. In: Proceedings of the 9th International Conference on Cooperative Information Systems (CooplS'01). Springer-Verlag, London, UK, pp. 418–432.

Shardanand, U., Maes, P., 1995. Social information filtering: Algorithms for automating word of mouth. In: Proceedings of the SIGCHI conference on Human factors in computing systems (CHI'95). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 210–217.

Singh, M., Yu, B., Venkataraman, M., 2001. Community-Based Service Location. Comm. ACM 44 (4), 49–54.

Stojanovic, N., Maedche, A., Staab, S., Studer, R., Sure, Y., 2001. SEAL: A framework for developing SEmantic PortALs. In: Proceedings of the international conference on Knowledge capture (K-CAP'01). ACM Press, New York, NY, pp. 155–162.

Tarkoma, S., Kangasharju, J., Lindholm, T., Raatikainen, K., 2006. Fuego: Experiences with Mobile Data Communication and Synchronization. In: Proceedings of the 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'06). pp. 1–5.

Terveen, L., Hill, W., Amento, B., McDonald, D., Creter, J., 1997. PHOAKS: A System for Sharing Recommendations. Commun. ACM 40 (3), 59–62.

Toivonen, S., Lenzini, G., Uusitalo, I., 2006. Context-aware Trustworthiness Evaluation with Indirect Knowledge. In: Proceedings of 2nd International Semantic Web Policy Workshop (SWPW'06), held in conjunction with the 5th International Semantic Web Conference (ISWC 2006).

Wang, C., Carzaniga, A., Evans, D., Wolf, A., 2002. Security Issues and Requirements for Internet-Scale Publish-Subscribe Systems. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02). Vol. 9. IEEE Computer Society, Washington, DC, USA.

Yap, G.-E., Tan, A.-H., Pang, H.-H., 2005. Dynamically-Optimized Context in Recommender Systems. In: Proceedings of the 6th International Conference on Mobile Data Management (MDM'05). ACM Press, New York, NY, USA, pp. 265–272.

Yu, Z., Zhou, X., Zhang, D., Chin, C.-Y., Wang, X., Men, J., 2006. Supporting Context-Aware Media Recommendations for Smart Phones. IEEE Pervasive Magazine 5 (3), 68–75.