1. Introduction

With the technological breakthroughs on mobile computing, nowadays the mobile devices allow people to carry along with themselves their work, leisure and even communicate with their friends and family, in the different ways anytime, anyplace.

Mobility brings a new dimension to the concept of communication environment. People feel more need to communicate with each other, as their jobs take most of their time.

Another important trend is the growing ubiquity of the Internet’s, as web access expands exponentially, becoming an essential tool for many people worldwide. This tool is so important in our daily lives that many activities that were once made on paper or confined to certain spaces are now replaced by online processes.

This document presents one of the services that can be implemented in the context of mobile social software. Time2me is an innovative offer that intends to work on mobile devices and whose main goal is to promote the communication between family and friends inside a social network.

The proposed application intends to be a way of sharing users’ context information, automatically captured and in the least intrusive way that is possible. The creation of a community that shares location information, routes usually taken in their routines and the time to get to a place or person is the premise on which this service works.

Time2me operates on a social network, which invites every member to share his information so the other members can look it up. That information is saved in a central server whose terminals are the mobile devices that users carry along with themselves.

It intends to offer a service that allows the users to better plan and coordinate their day-by-day lives and work schedule in a way never seen before, prognoses peoples’ arrivals, share their locations with their families, friends and coworkers. All this is achievable by sharing the information, in a model where every member of this community will have access to all these services if all the members collaborate in the growth of the social network.

The remainder of this document is organized as follows. Section 2 presents a vision of the platform’s architecture and all the important aspects in the platform construction. Section 3 presents all the implemented services and its options designs are in the application design. Section 4 discusses new social paradigms in
2. Architecture Overview

Time2me presents a typical client-server architecture, where the terminals can be mobile devices like PDAs or PCs. However, the PC client is outside the scope of this dissertation.

The figure below presents an overview of the system.

![System Platform Overview]

**Fig. 1.** System Platform Overview

Time2me central services are the web services (ws) which will feed three client applications, a website, a Vista Gadget and an application for Windows Mobile.

What distinguishes the website from the PDA gadget is the absence of mobility, in other words, one of the advantages of this architecture is the possibility given to the users to use time2me, even when they're not in front of their PC, adding the mobility paradigm to this project. In fact, mobility extends the frontiers of everything we can do and the services that can be made available and, in this specific case, is the most important element in gathering information that feeds the system, because it has a GPS receiver that improves the information the system collects.

The client-server architecture upon which time2me relies is a connection between processes that are running on different machines. There is a relationship from many-to-one between clients and server, where the client is responsible for starting the dialogue, by issuing a request or subscribing a service, implementing an asymmetric protocol of communication.

The client and the server are poorly connected, meaning that the system supports intermittent connections so it can support mobility. Synchronized communication is impossible because of the mobile environment, since the communication has too many losses which would compromise all the client application functionality.

One of the most important aspects of mobile social software development is that it can be upgraded easily. Its user community is an essential asset, so the architecture has the duty to facilitate the introduction of new servers and the subscription of new clients to their services.
Most of the code and data are kept centrally, reducing the maintenance costs, devaluing the low computational power and storage of mobile devices and contributing to data integrity.

3. Application Design

Time2me, in its simplest form, is a set of context-based services, whose main goal is to participate actively on users’ day-by-day lives. Describing this application is describing the services that it is made of and the way they’re available to the user.

3.1 Sign in

Being an application that runs on client-server architecture, the first step is the connection to the central server to obtain the user’s data. A SignIn service will bring all the initializing data of the application, molding it to the user.

- **SignIn (username, password)**

  A username and a password will be needed to be identified in the system and the server will return all the needed data to personalize time2me.

  After the user has logged in, the server sends in the response the user’s data, which is catalogued as a TimeTag. This was the name chosen to describe the objects that form the information identities of people and places; each one has elements to describe the name, an avatar (graphic representation of each object in the system), a position, etc. This way, each object capable of interacting with another is defined as a TimeTag, which is defined as a label of a place or person in a temporal location.

  The next step is to request the server all of the TimeTags that relate to the user. This request invokes **ws GetTimeTags**, which looks up in the database all information entities that are connected to the SignIn, returning a list of TimeTag objects, separated by type (People or Places).

- **GetTimeTags (username, password, peopleOnly)**

3.2 Buddies/Places List

The chosen representation for the TimeTag list was highly influenced by a very popular design model Windows Live Messenger, which has shown a huge success because of its simplicity and ease of use. The object representation in a list form is used by many operating systems, where the objects per se have a graphic representation. For example, the file system is represented with a folder or icon list.
These are considered the application’s main screens, where the people or places the user has in his TimeTag list is represented. These are two independent lists, but only in terms of the user interface, because both of them arrive in response to TimeTag service. List change is done with the left button, which toggles between the two lists, as can be seen in figures 2a and 2b.

3.2 Basic Services

The application offers basic services, which can be characterized by direct manipulation of the lists’ elements, either people or places. The operations considered are adding an element to the list, removing an element from the list, and changing an element. The simplest operation and the one that requires fewer operations is deleting, appearing only one dialogue box to confirm the operation. The service is DeleteTimeTag and it will erase the user’s reference to that element in the database. The behavior of this service is similar to a garbage collector in the server, in other words, the object will be removed from the database when there are no references to it.

- **DeleteTimeTag** (username, password, timeTagName)

The function’s interface is constituted by many fields, where the server compares the received information with the one that is in the server and only changes the one that is different. In this particular case we can change the FriendlyName and the Description. This service is also used for purposes other than Edit/Rename of people or places. For example, if the user changes his avatar, the application will also invoke this service but will only change the avatar field.
When we want to add elements to the list, the actions taken depend very much on the kind of element we want to add. The process is really simple when we want to add a place, given that the sequence of screens is the same as when we change a place. The same doesn’t happen when we want to add a person because it requires an authorization to do that. The Add friends is the first service that requires interaction between people to work. This is because users must create an invitation and wait for an authorization from the other user.

Member privacy was always taken in consideration in the application design and sharing of information for selective mechanisms were built to protect the users. The add example shows such a privacy mechanism: to add a friend to his buddy list the user has to build a message that will take a form of an invitation. Depending on the user’s choice (e-mail or username) to send an invitation a different ws is invoked. For the e-mail the InviteOtherDestination is used, and for the username, InviteTimeTag.

- InviteOtherDestination (username, password, OtherDestination)
- InviteTimeTag (username, password, TimeTagName)

The server is in charge of forwarding a message to the receiver, and how this is done also depends if the user is already a member or not. If the user is not a member he will receive an e-mail with the message defined by the user, if he is a member the message process depends on the message architecture that was defined. In this case, the message arrives to the receiver on his behalf, meaning each user can check if he has new messages in the server from time to time, and if so download them. To support multiple messages it’s necessary to implement a mechanism that gathers and saves messages for each user, and when a message is delivered to one of the users, the server takes it out of that list and waits for the user to ask for new messages again.

### 3.4 Dynamic Information

The buddy list has two types of dynamic information that demands frequent updates, the times and movement icons that appear in the avatar of the user’s contacts. This information does not appear for the places because it doesn’t make much sense since they are static, and we also have to take in consideration that the update of this information is very expensive both in bandwidth usage and local processing.

The result of this dynamic information is illustrated in figure 2a. These icons that appear on the avatars represent users’ movement. Play, pause and stop symbols were chosen to ease the reading. Figure 3 represents different possible status for each user.
In the figure’s left side are the time intervals considered for each icon, where a user is considered to be moving if his position has changed in an interval of zero to five minutes. When we use the GPS module that is presented in the application to move through the city, it sends user position updates to the server in case he is moving. With this information we can build social information that allows your friends to know your position and movements in a non-intrusive way. This is an example of a service that can be obtained from another service that is unrelated to the first one and allows the application social module to enrich itself.

4. Social Interaction Paradigms

Many of the services explained here need social interaction, because they need to have communication between two users to take the most advantage from this social application.

Time2me defines two types of interplay, an intentional interplay, using the message platform, and an automatic interplay, with the concept of TimeFence around each user. The message platform created for this application is considered intentional interaction, because it requires the user to initiate a conversation to establish an interaction between two users. This platform is very useful to other services that use time2me, because it is used as an interface to all the services where there’s the need for a request/response communication between two players.
Figure 4 illustrates a notification message. In this particular case, Isabel intends to be added to the user’s buddy list.

The construction of the notification is the responsibility of the mobile application, which from time to time invokes the ws GetNotification to know if there are any new messages.

- GetNotification (username, password)

In case they exist, these messages can be from one of these four types: Message, Share, Ask or Invite, and the application has the responsibility to build the text that the receiver will see.

On the other hand, the TimeFence concept intends to define the temporal distance between two TimeTags, working as proximity alert. These notifications are automatically produced and may have many purposes upon which we can implement new services. This interaction is considered automatic because in case the user has his TimeFence on, he will receive proximity notifications both when users and places that are near him.

The idea of creating a TimeFence surrounding each user came up from an already implemented idea named GeoFence, where a virtual fence is created and when any element enters that perimeter a response is issued. GeoFence’s basic metric is spatial distance, for example, every element that is in ten meter radius is considered to be within the virtual fence.

Starting from this premise and adapting it to reality of this application, it has been proposed that the basic metric would be the temporal distance between two elements instead of the spatial distance, and these elements can be people or places. One of the most significant differences between these two models is the virtual fence’s radius. In a GeoFence model the radius is constant and never changes through time and the fence is similar to a well defined circumference. On the other hand, the fence is not a circumference in a TimeFence because a temporal distance can correspond to many spatial distances that can change through time. Translating temporal distance into spatial distance normally presents a significant variation in time intervals; however, it makes the context information much more richer. To illustrate the relevance of the
temporal distance we can consider a spatial distance of 10km inside a busy city where it is easy to see that the time we take to travel that distance depends on the time of the day.

Time rules the perception of distance, and the user can define his TimeFence’s radius. In case an element enters the user’s perimeter it can origin an action. This application also has a mechanism denominated a doorbell; the user receives a notification on his PDA telling him that he is at a pre-defined temporal distance from an element of his list, being that a place or a person.

5. Conclusion

The subject of this document was mobile social software with the realization of an application working on a PDA device, which intends to promote the communication between family and friends “connected” in a virtual community. This application intends to offer a platform that promotes the sharing of information between users. This information is captured from the user’s context and then shared with the purpose of facilitating the planning of user’s day-by-day tasks, offering services that allow sharing of locations and temporal distance predictions between users.

As was seen, the capture of context information is made automatically and in the least intrusive way possible. The necessary services were developed to make sure that information reaches, in an intelligible way, the user’s buddies.

From a design perspective, one important option was the to divide the application into blocks, which facilitates the introduction of new services, given that a large and dynamic set of services can coexist in a mashup application. This design by blocks eases the addition of new services that can enrich the platform in the future, given that only a few of them were implemented and that there are many more that can fit this kind of social software.

Also, this platform makes extensive use of the notion of time, namely of temporal distance between two places or people, promoting the idea that this information is more rich than spatial distance. As an example to illustrate the power of this context information, note that if depending on the time of the day, we may vary the time we take to travel the same distance, and that this variation depends on the context. However, it is very hard to use time as an exact measure, but as we have the possibility to introduce more and more context information to these measurements we can minimize the error and provide a powerful tool to users for better organize their day-by-day lives.
References