Pomodroid Project Presentation speech

SLIDE 1 TOM

Good afternoon, welcome to our project presentation. We are team number 2, composed by Daniel Graziotin and Thomas Schievenin. We are going to present our project, named Pomodroid. Pomodroid is an activity manager based on the Pomodoro technique

SLIDE 2 TOM

We will briefly introduce some key aspects of the Pomodoro Technique. Then we will see the mission statement, a brief summary of the functionalities of the program, a high level description of the entities, the system architecture in terms of worflow, a short presentation about the special database we found, the problems faced and finally we will draw some conclusions about the experience.

SLIDE 3 DANIEL

Let's speak about the Pomodoro Technique:
It is a time management methodology that can be used with any type of task, or activity. The aim of the technique is to use time as a valuable ally instead of an enemy, in performing what we want to do in the way we want to do. The fundamental unit of the technique is the Pomodoro, which can be seen as a non-separable unit of time (25 minutes normally).
The basic steps are: choosing an activity, start a pomodoro, work and focus on the activity until the pomodoro finishes and take a break. This not all regarding the technique, for more information please visit the website

SLIDE 4 TOM

Our mission statement was to create an activity manager implementing the basic features of the Pomodoro Technique and to offer a very simple interface to the user.

SLIDE 4 TOM

A description of Pomodroid in 4 points <leggere slide>

SLIDE 6 TOM

In this slide we can see a simple high-level structure of the system entities: we can see the user interacting only with the GUI, that hides all the real behaviors and data that are defined in a MVC pattern. Data obtained from the external services (Trac and PROM) uses XML-RPC. The next slide will go further

SLIDE 7 DANIEL

This is a more detailed, even if high-level description of the system architecture seen from a workflow point of view. These are really our packages. The flow starts with the use of the services layer: TracTicketFetcher retrieves tickets from the Trac system. Then the control is passed to the factories layer: ActivityFactory converts the tickets into our Activity objects. The persistency layer extends the model layer, that simply defines our internal Entities, and is a wrapper around db4o database. More details about db4o will follow in the next slide. After the user facing of Activities with Pomodroid, the flow goes in the opposite side: the metrics gathered during a Pomodoro session, called Events, are passed to factories layer, respectively to the PromFactory, that converts them to the proprietary PROM format, called ini, composed by Entities and Properties. The ini String is then zipped, and passed to the services layer. PromEventDeliverer is responsible to access PROM using XML-RPC and send the zipped Ini file.

SLIDE 8 DANIEL

Let's talk about the particular database we found. Db4o is an opensource object oriented database written in Java in form of a jar library, that permits developers to store, edit, delete and retrieve pure complex Java objects within a few lines of code, without writing a single SQL query. Another nice feature of db4o is its adaptability to schema changes: when we add or modify a class attribute, no changes to the database are necessary. Db4o fully works for Android.

SLIDE 9 TOM

We provide here a tiny example of db4o usage. This Activity method is responsible to return all the completed activities (ie. , set as finished by user). We simply prepare a query object bounded to the Activity class. We then apply a constraint (like in WHERE SQL statement) to select only the Activities for which done is equal to true. Then we ask db4o to order the selected Activities by deadline. As you also see, we catch any exception from all our methods and throw a customized exception called PomodroidExcept, to better handling them at the GUI level.

SLIDE 10 TOM

Android-xmlrpc library does not provide support for HTTP basic auth, used by Trac on babbage. We modified its source code and added a method to provide authentication.

Trac XML-RPC plugin is very limited and poorly documented. The plugin does not provide a write access to Trac. Therefore, it is impossible to create tickets from our device, neither close the tickets within Pomodroid. We spent a lot of time in writing Python scripts to discover the behavior of each exported method.
PROM has been a very large problem for us. First of all, we could only work with it by the end of November. The installation of PROM was done on a private server and was arduous. The non-availability of PROM source code and its database composed by more than 200 tables has also been a big problem, in order to discover its functionalities and test them.

At the beginning of the project, we wanted to implement the possibility to block distractions like incoming sms/phone calls during a Pomodoro session. Unfortunately, this is impossible even with using Android 2.0. Those features are not yet implemented.

The timer we implemented in Pomodoro.java counts two seconds when time slides from 00:01 to 00:00. We don't know why this happens, it seems more like an Android bug.