Pomodroid Project Report

Daniel Graziotin, 4801, daniel.graziotin@stud-inf.unibz.it
Thomas Schievenin, 5701, thomas.schievenin@stud-inf.unibz.it

Problem identification

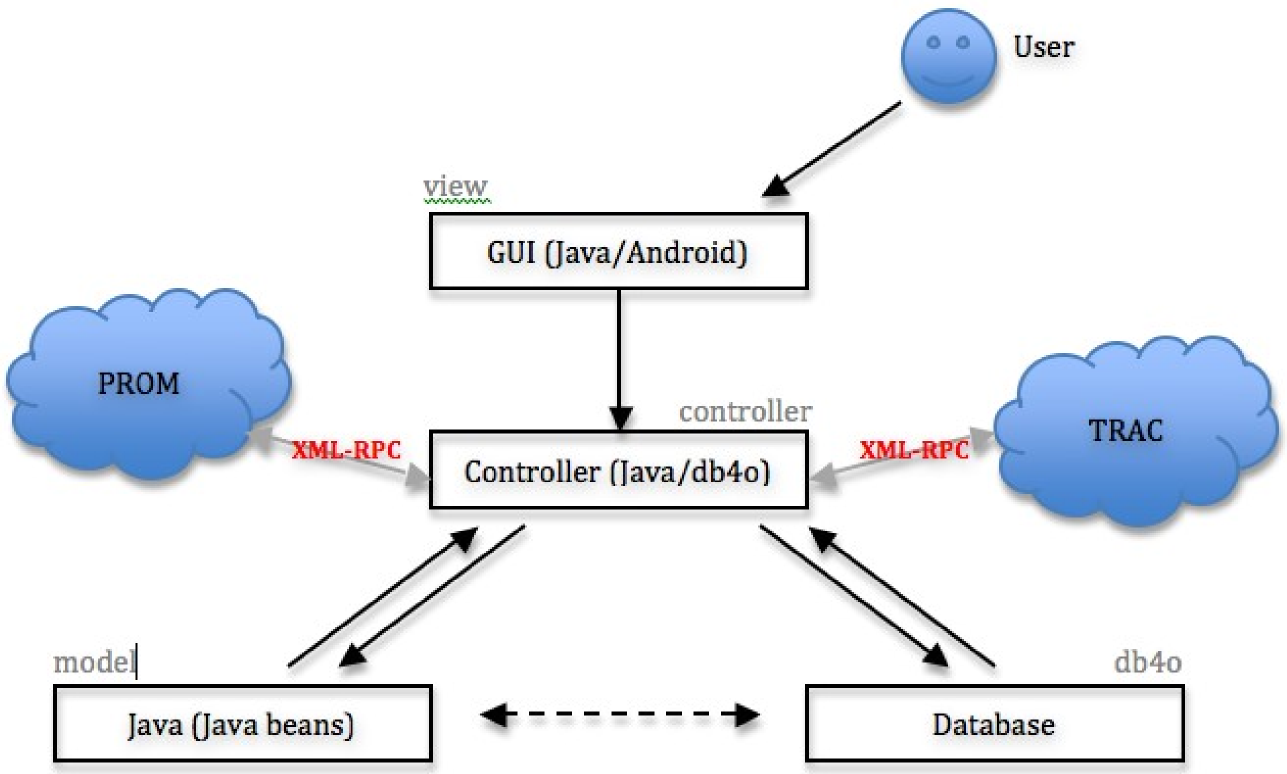
CASE is missing a mobile tool that gives to a developer the possibility to retrieve his tasks from a Trac system and measure their effort using the Pomodoro Technique[1], created by Francesco Cirillo[2]. A connection to PROM to store all metrics gathered during a Pomodoro session is also missing.

The solution

Pomodroid is a Java/Android application that interacts with an internal Trac system of CASE[3], retrieves developer's tasks and lets him work following the basic rules of the Pomodoro technique. All the information collected during developer's activity are then submitted to PROM[4].

Pomodroid focuses on basic features of the technique. It does not focus on advanced techniques, such as the prediction of the number of Pomodoros needed for an activity. See “Features and limitations” for more details.

The following is a very high level description of the entities interacting in Pomodroid:



1 http://www.pomodorotechnique.com
2 http://www.xplabs.it
3 https://babbage.inf.unibz.it/trac/
4 http://www.prom.case.unibz.it/

Glossary

Pomodroid heavily relies on the entities described in the Pomodoro Technique, that are introduced below:

- **Activity** – the tasks that the user wants to face using Pomodroid.
- **Activity Inventory Sheet (AIS)** – the general activity container. Contains all the activities registered in the system.
- **TODO Today Sheet (TTS)** – the daily activity container. Contains just the activities that the user will face in the current day
- **Trash Sheet (TS) –** like the Recycle bin, is a container of the activities finished by the developer.
- **Pomodoro** – an indivisible unit of time (normally 25 minutes) used to measure the resource dedicated to an Activity.

Architecture

This section contains a brief description of the architecture of Pomodroid. For detailed information, please look at our exported Javadocs contained in `docs/javadoc` folder.

A more detailed description of the packages working under the GUI can be found in the image located under `docs/model-controller-design.jpg`

it.unibz.pomodroid.models

Contains the basic models of the system (Activity, Event, User), implemented as Javabeans. They are basically just attributes, constructors, getters and setters. No other features are implemented.

it.unibz.pomodroid.persistency

The classes in this package extend the models previously described (Activity, Event, User). They define all the methods necessary to have the object persistently stored in the database, as well as methods for retrieving/deleting objects, etc. Pomodroid uses these classes and does not touch the models. We decided to follow this implementation strategy because of the choice of the database. We adopted db4o[5], an easy, yet powerful object-oriented database that lets the developer store/retrieve/delete/edit complex objects without writing a single query.

This package also contains `DBHelper`, a class that provides all the basic configuration and behaviors of the database (open/close, commit, configure, defragmentation, backup etc.)

5 http://www.db4o.com/

it.unibz.pomodroid.services

This package contains all the definitions of the Internet services used by Pomodroid.
`XmlRpcClient` is a wrapper for an external opensource library, `android-xmlrpc`[6]. It defines the methods necessary for (authenticated) XML-RPC calls for fetching either single or multiple objects. It also contains an utility method that checks if Internet is available on the device.
`TracTicketFetcher` contains the methods necessary to connect to a Trac system and retrieve the opened tickets owned by the user.
`PromEventDeliverer` defines the methods necessary to send Entities and Properties to Prom through XML-RPC.

it.unibz.pomodroid.factories

The package holds the classes responsible for converting external entities in Pomodroid Activities, and Pomodroid Events in Prom's Entities and Properties. The class `ActivityFactory` does also fill all the attributes missing in case of little detailed tickets, like missing deadlines or descriptions.

it.unibz.pomodroid.exceptions

This package only contains `PomodroidException.java`, a common shared Exception caught and thrown in the whole program. This is for better handling exceptions from the GUI. It also defines some utility methods to display alert dialogs from the GUI.

it.unibz.pomodroid.test

The Test Suite of Pomodroid. It contains multi-threaded tests that cover all the most important methods of all the other packages. Tests are run by the program itself.

The exported design class diagram can be found under docs/class-diagram.png

Features and Limitations

A full implementation of the Pomodoro technique in 3 months is impossible. The basic features of the Pomodoro technique implemented in Pomodroid are the following:

- Organization of activities in an Activity Inventory Sheet, including the deadline
- Pickup of activities from the AIS and put them in the Todo Today Sheet
- Organization of activities in the TTS (order of execution)
- Face an activity (Run a Pomodoro/Break a Pomodoro)
- Registration of a completed Pomodoro in the activity

6 http://code.google.com/p/android-xmlrpc/. Contained in org.xmlrpc.android package

- Short Break and Long Break alerts (after 4 completed Pomodoros)
- Postpone an Activity (i.e. remove it from TTS but not from AIS)
- Finish an Activity
- Remove an Activity

Other features that are implemented are the following:

- Retrieval of user's activities from Trac
- Gathering of metrics during a Pomodoro session
- Sending the metrics to PROM
- Detailed description of the activity when user performs a long click on it
- Easy and fast access to all the features within a couple of clicks.

Problems encountered

- Android-xmlrpc library does not provide support for HTTP basic auth, used by Trac on babbage.
  → we modified its source code and added `setBasicAuthentication()`, to provide the feature. We also modified its standard Exception to extend `PomodroidException`, to fully integrate the library in Pomodroid
- Trac XML-RPC plugin is very limited and poorly documented. The plugin does also 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.
  Anyway, we don't think that the possibility to modify the description or other aspects of an activity would be of real interest. A developer should already have his tasks defined by himself or his responsible using traditional methods
- PROM has been a very large problem for us. First of all, we could only work with it by the end of November. We met a very polite PHD student that gave us the binaries of PROM server, a database dump, and explained briefly the tables of the database. Although, the installation of PROM was done on a private server and was arduous. The non-availability of PROM source code has also been a big problem, in order to discover its XML-RPC functions
- Unfortunately, the possibility to block distractions like incoming sms/phone calls is impossible even when using Android 2.0. Those features are not yet implemented. The only thing we could have done was the block of the Internet connection, which blocks incoming mails etc. But by doing this, we would have also disabled too many functionalities of Pomodroid.
  → these feature, that was also considered a nice to have, have been dropped down by our implementation strategy.
- 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

- It may rarely happen that an exception is thrown when trying to retrieve tickets from Trac. This is because of SSL handshaking failures. However, the exception is correctly handled and the user is warned with a dialog that tells him that a connection error happened, and to try again.