Politecnico di Milano

Software Engineering 2 Project

MeteoCal

Acceptance Testing Document

Authors:
Andrea Celli
Stefano Cereda

May 16, 2015

Contents

Part I
Introduction

The assigned project is `https://code.google.com/p/meteocal-iodicefinardi/`.

In this document we will describe the issues we found in the project. We will start by describing the behavior that does not match with the assigned project, or at least with what we think to be the most rational behavior (II). Then we will discuss some design problems (III) and we will conclude with a report on the test (IV) and with a quick comment on code and documentation quality(V).

Part II
Requirements

1 Different cities with the same name

The rasd does not clarify how the system will deal with different cities that have the same name (for example `http://openweathermap.org/help/city_list.txt` contains 7 different Dublin). Neither the DD explains this issue.

Looking at the code of WeatherHttpClient.java we saw that the meteo is obtained passing the direct name of the city. Therefore if the user inserts Dublin as event location the forecast will be downloaded for Dublin, US (it comes first in the city list) instead of the more reasonable Dublin, IE. The user could overcome that by inserting "Dublin, IE" as location, but that's not stated anywhere and the application does not specify in any way which is the picked city, so the user can't be aware of the error.

2 Weather forecast

In the RASD (page 4 section 1.1 for the first time) it's stated that the user can choose a kind of weather, while the project assignment clearly states that the system has to handle *bad* weather forecast. This is a problem because it's impossible to obtain a behavior that matches the one of the assigned system, suppose a scenario like this:

Event weather	assigned behavior	wanted=sunny	wanted=cloudy
sunny	good	good	bad
cloudy	good	bad	good

This is assuming that cloudy is not a bad weather. The only situation where the two behavior matches is when we consider only sunny to be a good weather, but we think that this is a great limitation, because in most of the cases we don't reschedule our appointments only because the weather is partially cloud. A more reasonable approach is to consider sunny,partially cloud and cloud as good weather and everything else as bad weather.

This approach is reflected in the domain assumptions (page 6 2.2) where it's stated that "A person desires at most one type of weather for an event."

The real problem is how the partially cloud forecast is handled, and that's not clearly stated in any part of the rasd.

3 Bad weather alert for the event creator

The RASD states (page 10 section 3.2.1) that registered users should "Be notified (one day before the event) whenever an outdoor event they accepted an invitation for has an unfavourable weather forecast."

We don't think that this is the assigned behavior. The assignment states that the first type of notification should be sent to all the event's participant, but we think that even the event's creator should be considered a participant.

Part III
Design

4 Date of birth and city

In the dd (page 7 section 3.1.1-User) it's stated that a user should insert his date of birth and the city where he lives. These attributes are not necessary for any of the requirements in the rasd. Moreover, using the application we noticed that a user can only be older than 10 or without a date of birth (during the registration the date of birth is not mandatory but if present can only be set prior to 10 years ago). We think that this makes no sense.

5 Closest day with expected weather

The dd assumes (page 7 section 3.1.1-WeatherNotification) that a WeatherNotification (the one sent three days in advance to the event organizer) should contain an information about the closest day with the expected weather. We don't think this is a good choice, because from the time when the notification is created to the time when the notification is visualized the forecast could change, making the notification incorrect. We think that the closest good day should be searched when the notification is visualized from the user.

6 Limited forecasts

The system retrieves the forecasts via a call to `http://api.openweathermap.org/data/2.5/forecast/daily?q=[city]`, obtaining daily forecasts that cover 10 days. OpenWeatherMap also offers forecasts with a time coverage of 3 hours that are much more accurate for the purpose of such a system. Furthermore the daily forecasts are available for up to 16 days, so we think that a better solution is to use the 3 hours forecasts when possible (up to 5 days) and the daily forecasts for the remaining days.

Part IV
Project testing

7 Automated tests

Here we will describe only the automated tests with some problems.

7.1 SetDate instead of SetHours

The two tests `EventManagerTest.newEventShouldBeSavedOnce` and `CalendarManagerIT.eventsGetAddedToCalendarOfCreator` fail when executed after 12.00 a.m. with this error: `You have inserted a not feasible interval time.`

That's due to an error at lines 55 and 119 (respectively) where instead of `startingTime.setDate(8);` there should be `startingTime.setHours(8);`.

8 Manual test

We tried the manual tests proposed in the System Testing document and we also tried all the rasd's requirements with strange input. Here we will report only the things that does not work as expected, everything that is not stated works well.

8.1 Registration

After completing the registration the browser proposes to save the login credentials, but for username it takes the date of birth instead of the email.

8.2 Incorrect forecasts notifications during updates

During the periodical update (done automatically every 10 minutes) the system downloads the wrong forecast, getting the one for the following day even for events that are scheduled in three days.

We think that's due to a bug in WeatherChecker.java at line 89, where a String is compared with == instead of using .equals().

Part V
Code and documentation quality

Some methods are missing the javadoc documentation, even though they are the most simple and straightforward.

The documentation rate (according to `http://cloc.sourceforge.net/`) is a little low (30% on the Java classes, 11% on the test classes and 6% in the xhtml files).

We would have liked more meaningful comments (especially in the javadoc, where many methods have as comment something like *method_x(p) does x on the given parameter x*). This will influence the maintainability of the software product, since the mechanisms of the most complex classes are not so clear for someone who did not write them.