MeteoCal Project Reporting Document

Simone Graziussi

10th February 2015

Function Points Method

In this section the Function Points approach is used to estimate the size of the *MeteoCal* project.

The system function types are:

- **Internal Logic Files**: the system stores data about
  - users
  - calendars
  - events
  - notifications

  All these entities can be considered "simple" since they have few attributes: 4x7=28

- **External Interface Files**: the only external interface is the weather API that provides the forecasts for a given city. It can be considered "simple": 1x5=5

- **External Inputs**: the user is able to:
  - register
  - log-in/log-out
  - create a calendar
  - browse the calendar
  - manage events
  - manage notifications and invitations

  We can consider event management as "medium", all the others as "simple": 5x4+1x5=25

- **External Inquiries**: the application allows users to request information about
  - the created events ("medium" complexity)
  - unread notifications/invitations ("simple" complexity)

  Total: 1x3+1x4=7

- **External Outputs**: the system manages the creation of notifications (including bad weather alerts). It can be considered "medium": 1x5=5

The total number of Unadjusted Function Points is 70, which, given this conversion table should mean more or less 3200 Java EE lines of code. The actual SLOC of the project are 2000 so the Function Point result is an overestimate.

COCOMO Method

In this section the COCOMO approach is used to estimate the effort of developing *MeteoCal*. First of all we need to define the values for Scale Drivers and Cost Drivers.

Scale Drivers:

- **Precedentedness**: Low (I never developed a calendar before)
- **Development Flexibility**: Nominal (the assignment was precise but left some flexibility in the development)
- **Architecture/Risk Resolution**: Low (there hasn't been any formal risk analysis)
- **Team Cohesion**: Extra High (being a one-person project team cohesion is of course maximum)
- **Process Maturity**: Low

Cost Drivers:

- **Required Software Reliability**: Low (no particular reliability ratio was required)
- **Data Base Size**: Low
- **Product Complexity**: Low
- **Developed for Reusability**: Nominal (some effort was spent to construct components to reuse internally)
- **Documentation Match to Lifecycle Needs**: Nominal
- **Analyst Capability**: Low
- **Programmer Capability**: Nominal
- **Personnel Continuity**: Nominal
- **Application Experience**: Low
- **Platform Experience**: Low
- **Language and Toolset Experience**: Low
- **Time Constraint**: Nominal (the assignment didn't require particular time constraints)

- **Storage Constraint**: Nominal (the assignment didn't require particular storage constraints)
- **Platform Volatility**: Nominal (platform is quite stable)
- **Use of Software Tools**: Nominal (basic lifecycle tools were used)
- **Multisite Development**: Low
- **Required Development Schedule**: Nominal (there was no schedule stretch-out or acceleration)

Given that the project SLOC are 2000, the resulting effort is of 5.7 person-months. This value was computed using this online tool, setting the "SLOC" field to 1500 New and 500 Reused with 10% integration, since some functions (like registration and log-in) are very similar to the examples provided during the course excitations. The actual project duration was of 3 months and, since it was developed by one person, the COCOMO result is an over-estimate.