POLITECNICO

MILANO

SOFTWARE ENGINEERING 2

TEACHER:
PROF. RAFFAELA MIRANDOLA

GuessBid

PROJECT REPORTING

Authors:

Eliseo Sartori
Alessandro Picca

Sommario

1. **Introduction** 4

2. **Function Point (FP) Method** 5

   2.1 Internal Logical FIles (IFFs) 6

   2.2 External Interface Files (EIFs) 6

   2.3 External Inputs (EIs) 6

   2.4 External Inquiries (EQs) 7

   2.5 External Outputs (EOs) 7

   2.6 FPs Summary 7

   2.7 Result and Analysis 8

3. **COCOMO Model Analysis.** 8

1. Introduction

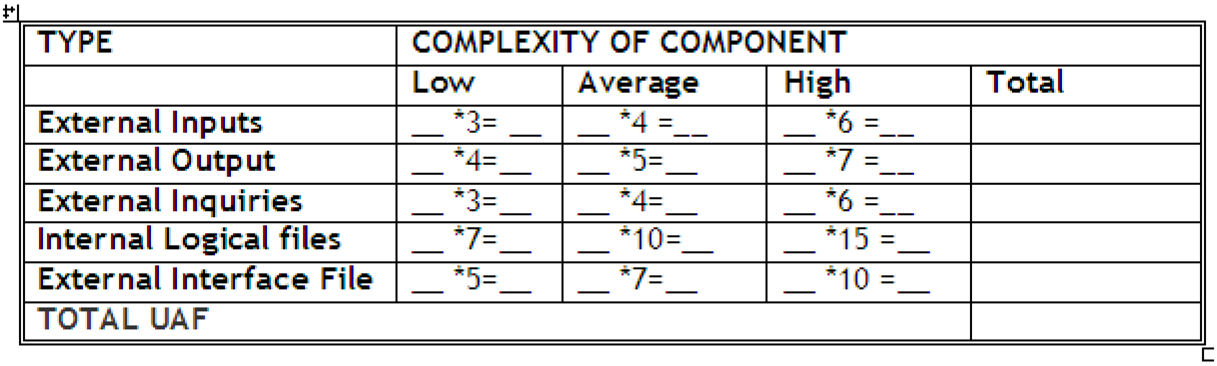
The purpose of this document is to provide an analysis and an estimation of the required effort for GuessBid project:
More specifically, in the first section the Function Point Analysis is carried out, whereas in the second part of the document an effort analysis with COCOMO model has been performed.

2. Function Point (FP) Method

FP is a standard method for quantifying the software deliverable based upon the user view. It is important to keep in mind that Function Point method looks at the logical view, not at the physical one.
For these reasons, things like coding algorithms, database structure, screenshots of transaction are not counted and the effort necessary to develop a software product depends on its functionalities.
A limit of the FP method is that is a very subjective method of analysis, because it depends on the estimator and that' s the reason why is impossible to build an automatic FP estimator.
Below, things that are conventionally keeped in consideration within the FP methodology are listed :

- ILF (Internal Logical File): homogenous set of data used and managed by the application
- EIF (External Interface File): homogenous set of data used by the application but generated and maintained by other applications
- EI (External Inputs): elementary operation to elaborate data coming from the external environment
- EO (External Outputs): elementary operation that generates data for the external environment
- EQ (External Inquiry): elementary operation that involves input and output.

The following table contains the scores that classify the complexity of each of the five categories described above;
For each category there are three possibility of complexity: “simple”, “medium” and “complex”.



TYPE	COMPLEXITY OF COMPONENT			
	Low	Average	High	Total
External Inputs	__ *3= __	__ *4 =__	__ *6 =__	
External Output	__ *4=__	__ *5=__	__ *7 =__	
External Inquiries	__ *3=__	__ *4=__	__ *6 =__	
Internal Logical files	__ *7=__	__ *10=__	__ *15 =__	
External Interface File	__ *5=__	__ *7=__	__ *10 =__	
TOTAL UAF				

The sum of all values represent the function point of the GuessBid project.

2.1. Internal Logical Files (IFFs)

In our application, data are stored in some database tables, so we' ll use them as the basis of this analysis. Hereafter, a list of the database table names used in GuessBid is shown:

- Users
- Auction
- Notification
- Groups
- Bid

Each of these tables is composed by few fields, so its complexity can be defined as low, and the total value for IFF would be : FPs ILF: 5 x 7 = **35 FPs**

2.2 External Interface Files (EIFs)

In our application there aren't data coming from external sources, so the total value for EIF would be: FPs EIF: **0 FPs**

2.3 External Inputs (EIs)

In this category all elementary operations introducing changes in system internal data are listed:

- Login / Logout
- Registration
- Create a new auction
- Generate Notification
- Place a bid
- Historical

Only the creation of a new auction and placing a bid (user must pass through auction list before placing bid) can be considered operations of medium complexity, while all other operations are low-complexity operations.
According to this classification, the total value for EI would be: FPs EI: 2 x 4 + 4 x 3 = **20 FPs**

2.4 External Inquiries (EQs)

This category includes all operations that deal with system data without modifying them; more in details, these operations are:

- Request info about all active auctions
- View End-Auction-Notification
- View Change-Position-Notification
- View User's historical

While the first of these operations is a simple operation, the other three EQs have a medium complexity, so the total value for EQ would be: FPs EQ: 1 x 3 + 3 x 4 = **15 FPs**

2.5 External Outputs (EOs)

As we said before, our application doesn't interface with external systems, so the total value for EO would be: FPs EO: **0 FPs**

2.6 FPs summary

The total function points of the system is given by the sum of all partial FPs, more in details:

FPs IFFs: 35

FPs EIFs: 0

FPs EIs: 20

FPs EQs: 15

FPs EOs: 0

Total FPs system: 70 FPs

2.7 Result and Analysis

Given total FPs of the system, with a simple formula is possible to calculate the estimated size for a project:

Size (LOC) = Total FPs * C

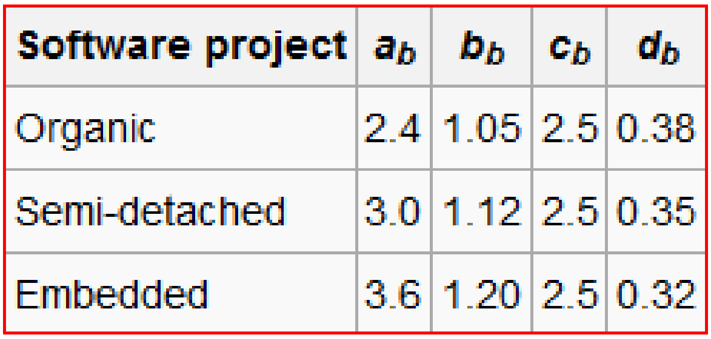
Where C is a constant, depending on which programming language is used in the project; specifically, for high level programming languages like JAVA and JAVA EE, the value of C is 53, so the estimation for the LOC of the project is: 70 * 53 = 3710.

Our effective number of code lines is 2190, so the estimation provided can be considered quite good.

3. Cocomo Model Analysis

In this section we applied COCOMO model to the actual size of the project in order to provide estimation of Effort (M) and Time (T) necessary to develop it.

We classified our application as organic, because it has been our first project using JavaEE, but we have some past experiences with Java language, HTML and CSS.



Software project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

We computed following estimations:

M (Man-Month) = a_b * S^(b_b) = 2.4 * 2.190 ^1.12 = 5.774

T(months)= c_b * S^(d_b)= 2.5 * 2.190 ^0.38 = 3.3675

Finally, we computed the estimated number of people necessary to develop the entire project simply dividing M by T:

N = M / T = 5.774/3.3675 = 1.7146, that can be approximate with 2, the number of person that worked on this project.