POLITECNICO DI MILANO
SOFTWARE ENGINEERING 2

GuessBid

Project reporting

Mirjam Škarica

Milan, June 2015

Contents

Chapter 1

Introduction

In this document a cost-based analysis of the GuessBid project. Chapter 2 provides the size estimate of the applications functionalities using the Function Point Technique. From there we derive the LOC (Lines Of Code) appraisal that is then compared to the actual size. In Chapter 3, using the COCOMO II approach, an estimate is obtained for total effort, duration and the people required for the project. This data is then also compared to the actual ones.

Chapter 2

Function Point Technique

To calculate Function Points, the following table is used:

Function types	**Weights**		
	Simple	**Medium**	**Complex**
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiry	3	4	6
Internal Logic Files	7	10	15
External Interface Files	5	7	10

2.1 Estimating FPs

Internal Logic Files (ILF)

The application stores thee following tables:

- Users table
- Auction table
- Bid table
- Notification table
- Category table

All have a simple structure with few fields, so we assume a simple weight for them, hence FPs = $7 \times 5 = 35$.

External Logic Files (ELF)

None, FPs = 0

External Input (EI)

- Login, log out, registration, (simple weight)
- New auction (medium weight)
- Delete auction (simple weigh)
- Place bid (medium weight)
- Change password, email, username (simple weight)

Hence, FPs = $3 \times 7 + 4 \times 2 = 29$.

External Inquiry (EIq)

- Browse ongoing and own auctions (medium weight)
- Browse own notifications (simple weight)
- View own settings (simple weight)

Hence, FPs = $3 \times 2 + 4 \times 2 = 14$.

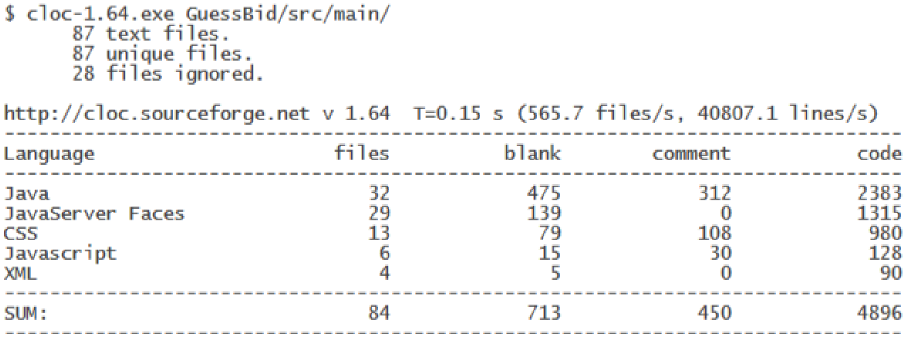
External Output (EO)

- Generate outcome and rank notifications (medium weight)
- Manage credit balance (simple weight)

None, FPs = $5 \times 2 + 4 \times 1 = 14$

2.2 Results and considerations

The total amount of FPs is: 35 + 0 + 29 + 14 + 14 = 92. This number is used to estimate the number of LOCs (*Lines Of Code*) using the AVC average factor[1] for J2EE which is 46:
 92 × 46 = 4232 LOCs.
Using the cloc tool [2] the actual number of lines of code is obtained (fig 2.1).



```
$ cloc-1.64.exe GuessBid/src/main/
      87 text files.
      87 unique files.
      28 files ignored.

http://cloc.sourceforge.net v 1.64  T=0.15 s (565.7 files/s, 40807.1 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Java                            32            475            312           2383
JavaServer Faces                29            139              0           1315
CSS                             13             79            108            980
Javascript                       6             15             30            128
XML                              4              5              0             90
-------------------------------------------------------------------------------
SUM:                            84            713            450           4896
-------------------------------------------------------------------------------
```

Figure 2.1:

On first glance it seems the actual size of 4896 LOC overshot the estimates of 4232 LOC. But if we were to ignore CSS lines of code we come to 3916 LOC. The reason why removing CSS is justified is because our estimates focus on functionality, hence the Function Point name; whereas CSS is used only for aesthetics, the look of the application. In conclusion, although the actual size and the estimated size isn't exactly the same, the estimate is a good indicator.

[1] http://www.qsm.com/resources/function-point-languages-table
[2] http://cloc.sourceforge.net/

Chapter 3

COCOCO II Technique

The COCOMO II Technique is used to estimate the effort, duration and people needed for the project.

3.1 Effort Estimate

To estimate the effort calculated in *Person/Months* needed, the following formula is used:

$$Effort = 2.94 * EAF * (KSLOC)^E$$

E Exponent derived from Scale Drivers. We use a nominal scale driver value of 1.0997

EAF Effort Adjustment Factor derived from Cost Drivers (product of the effort multipliers corresponding to each of the cost drivers for your project) $EAF = 0.91 + 0.01 * (PREC + FLEX + RESL + TEAM + PMAT)$. Where:

PREC Precedentedness reflects the previous experience of the organization with this type of project. Very low means no previous experience, Extra high means that the organization is completely familiar with this application domain.

$PREC = low$, because the team had previous experience with Java but this was the first time working with J2EE and Primefaces

FLEX Development Flexibility reflects the degree of flexibility in the development process. Very low means a prescribed process is used; Extra high means that the client only sets general goals

FLEX = nominal, because there were fixed initial requirements for the project

RESL Risk Resolution reflects the extent of risk analysis carried out. Very low means little analysis, Extra high means a complete a thorough risk analysis.

RESL = low, because the focus of the project was not on risk detection

TEAM Team cohesion reflects how well the development team know each other and work together. Very low means very difficult interactions, Extra high means an integrated and effective team with no communication problems

TEAM = high, because the team was comprised of one person

PMAT Process maturity reflects the process maturity of the organization.

PMAT = low, because this was the first time developing such an application so there was no process to speak of

Finally, using the values from the COCOMO II table for cost drivers[1] we derive EAF:

$$EAF = 0.91 + 0.01 * (4.96 + 3.04 + 5.65 + 1.10 + 6.24) = 1.1199$$

3.2 Results and considerations

With *KSLOC = 3.916* we obtain:

$$Effort = 2.94 * 1.1199 * (3.916)^{1.0997} = 14.77 Person/Month$$

$$Durration = 3.67 * Effort^{0.3179} = 8.64\, Months$$

$$People = Effort/Duration = 1.7\, People$$

In conclusion, the actual size of the team (1) and the estimation (1.7) doesn't differ too much. But the actual duration of the project was 3 months, where the calculations estimate 8.64 months. This may be because COCOMO II is an industry oriented method where a lot more time is spent considering security and scalability issues and projects go through many iterations as the team gets more familiar with the domain of the problem they are facing and because the client's requirements may change over time.

[1] http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf