Estimating LOC for OMA primitives: an approach based on correlation and regression analysis

Ednaldo Dilorenzo[1,2], Felipe Furtado[1,2], Mauro Silva[1], Elisabeth Morais[1], Gibeon Aquino[1,2]

[1]Recife Center for Advanced Studies and Systems (C.E.S.A.R)

[2]Federal University of Pernambuco (UFPE)

{edsf, fsfs, mjcs, emms, gsaj}@cesar.org.br

Abstract. *Estimating API size is a very complex work since no approaches address particularly this development unit. Existing approaches are concentrated in estimating the software as a whole. Since many projects need to estimate API size, analyzing previous development is extremely necessary in order to identify which variables lead the API size and this can bring managers to derive cost and effort from it. This paper presents a case study from a OMA protocol client project where those variables were identified and the correlation and multiple regression techniques were used to find a equation that can predict the size of development of API modules for this specific project in Lines Of Code (LOC) unit size.*

1. Introduction

One of the main challenges in software project planning process area is related to size, effort, time and budget estimates. The size estimate accuracy is fundamental because it is the basis for deriving effort, time and budget estimates [CMU/SEI-2002-TR-012 2002]. Resources, budget and time estimates for software development carries inherent risks, so they can affect the accuracy of estimates. Project size is another factor that can impact estimates, as it increases, the interdependency among software elements increases. Size refers to a quantifiable outcome of the software project and can be measured in lines of code (LOC). Experience and access to historical data, and the commitment to quantitative measures can reduce risks and uncertainty [Pressman 2004].

Several researches in software size estimates has been performed and traditional techniques has been applied and/or tailored, such as Function Point Analysis [Symons 1988] and Use Case Points [Karner 1993]. These kinds of techniques make account software size based on subjective parameters, related to requirements complexity and pre-established technical factors, and derives effort applying a productivity factor.

However, the predefined parameters adopted by these existing size estimation methods may not be applicable depending on project nature, being necessary the creation of a specific software size estimation method because parameters of existing methods can be inappropriate or difficult to apply. Alternatively, other elements of software specification can be used as estimation variable to produce an overall size estimate for the entire project, obtaining more accurate estimates. In this way, statistical techniques, more specifically, *correlation* and *regression analysis* can be used to define a software size estimation method for these kinds of projects. The correlation analysis is adopted to define

the relationship degree between project parameters and generated units of size for selecting a set of high correlation parameters. While regression analysis is adopted to define a mathematical equation which describes the correlation between the selected parameters and units of size generated from past development, enabling to predict software size for future development based on a sampling of historical data.

This article purpose is to report the steps performed to define a software size estimation method to predict lines of code for API's development, specifically, a protocol development based on OMA (Open Mobile Alliance) primitives' specification [oma 2006]. Section 2 introduces primitives and transactions based on OMA, Section 3 describes multiple regression analysis, Section 4 details a case study related to the mentioned line of codes estimation method and Section 5 encompasses the conclusions of this work and future works.

2. OMA IMPS

On April 2001 Ericsson, Motorola, and Nokia announced the Wireless Village (WV), the Mobile Instant Messaging and Presense Services (IMPS) initiative. A set of specifications was defined to be a standard for cellphones and mobile devices to use these services across platforms. The standard enables interoperability of mobile instant messaging and presence services by defining open standards for the different vendors.

The WV specification is maintained by Open Mobile Alliance (OMA). The main principle of OMA is to make products and services not dependent on proprietary technologies but instead create open global standards, protocols (currently at version 1.3) and interfaces. Nowadays major mobile phone manufactures often include OMA IMPS[1] compliant applications in their mobile software by default.

The IMPS System is a client/server-based system, where the server is the IMPS Server and clients can be mobile device, other services/applications, or fixed PC clients. For interoperability, the IMPS servers and gateways are connected with a Server-to-Server Protocol (SSP). Based on OMA IMPS Server will be defined four primary features that are applications servives elements.

2.1. Applications Services Elements

The Wireless Village Instant Messaging and Presence Service is composed of four Application Service Elements [wv: 2005] that are accessible via the Service Access Point (SAP)[2]: Presence; Instant Messaging (IM); Groups; and Shared Content.

- **Presence Service Element**: is a function element of the OMA IMPS 1.x Server and provides functionality for Presence information management. It includes client device availability (my phone is on/off, in a call), user status (available, unavailable, in a meeting), location, client service capability (voice, text, GPRS, multimedia) and searchable personal statuses such as mood (happy or ungry) and hobbies (football, fishing, computing, dancing);

[1] In this paper, the terms OMA IMPS and Wireless Village are used interchangeably.

[2] The Service Access Point (SAP) serves as the interface between the OMA IMPS 1.x server and its environment.

- **IM Service Element**: is a familiar concept in both the mobile and desktop worlds. Desktop IM clients, two-way SMS and two-way paging are all forms of Instant Messaging;
- **Group Service Element**: or chat are a fun and familiar concept on the Internet. Both operators and end-users are able to create and manage groups. Users can invite their friends and family to chat in group discussions. Operators can build common interest groups where end-users can meet each other online;
- **Shared Content Service Element**: allows users and operators to setup their own storage area where they can post pictures, music and other multimedia content while enabling the sharing with other individuals and groups in an IM or chat session;

These features, taken in part or as a whole, provide the basis for innovative new services that build upon a common interoperable framework. Based on the interoperability that is necessary to define and describe transactions and primitives that are used to provide Wireless Village service.

2.2. Transactions and Primitives

A WV transaction is a basic communication mechanism between a WV client and a WV SAP. A transaction consists of a request and a response primitive usually.

A transaction defines an mechanism of communication, most of times synchronous, between client and server. The Client Server Protocol (CSP) is supported by IMPS WV implementation. An simple and possibly a most common message is that of a client sending an instant message. The flow of this is depicted in Figure 1.

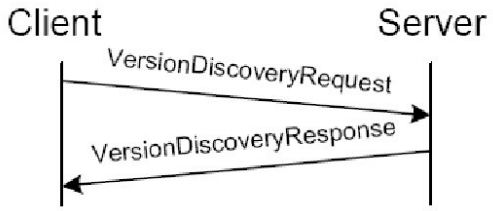

Figure 1. Version Discovery

A transaction is considered as closed when an final response primitive has been received (as shown in Figure 1 by *VersionDiscoveryResponse*), a time out waiting for a response primitive has occurred, or the underlying transport has been detected as *broken*. The Wireless Village CSP is composed of primitives, each of which represents a single function. In other words, a primitive represents a single request from the client to the server or from the server to the client. For example, a client can send a login request to the server and the server can send a login response to the client (see Figure 2).

3. Correlation and Multiple Regression

The estimative method presented here was created based on a statistical technique called *Multiple Regression Analysis* [Montgomery et al. 2001]. However, before presenting it, it is necessary to explain the concept of correlation because it is the bases of regression studies.

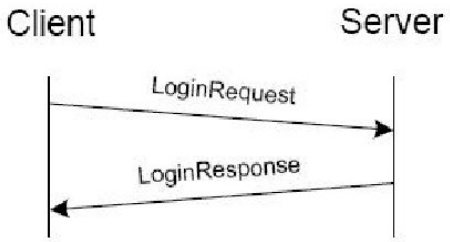

Figure 2. Login access control

3.1. Correlation

The correlation aim is to determine if there is, or not, a statistically significant relationship among variables. Only if a relationship exists, we can describe it with an equation that can be used for predictions [Triola 2005]. In this study were considered only quantitative and linear data.

There are two mechanisms used to analyze the correlation among variables. The first one is through a graphic called *Dispersion Diagram* which the data are plotted on a horizontal axis *x* and a vertical axis *y*. Each pair is plotted as a single point, as shown in Figures 3 and 4. Because of this analysis is only visual and, hence, subjective, the second mechanism is the study of linear correlation coefficient *r*. This coefficient measures the direction and intensity of linear association among variables and represents a measure of how well the regression line fits the data. The *r* value always is between -1 and +1. If the *r* value is close to zero, means that there is no significant linear correlation between variables.

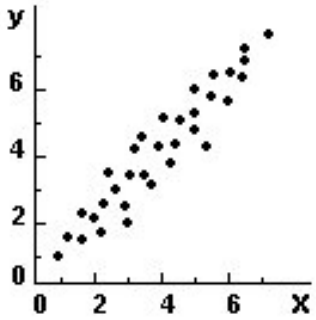

Figure 3. Example of a dispersion diagram with positive correlation (r around +1)

The *r* value is calculated by Equation 1 and can be easily find through specific tools, as will be shown in Section 4.

3.2. Regression Analysis

The regression analysis is used to describe the relationship among variables, through the equation and graph that represent the relationship [Triola 2005]. We must use the regression equation only if the correlation coefficient *r* indicates that there is a linear correlation.

In the case of multiple regression, the equation express a linear relationship between a single variable response *y* and two or more predictors variables *(x1, x2 ,..., xn)*. The general form of a multiple regression equation is given by:

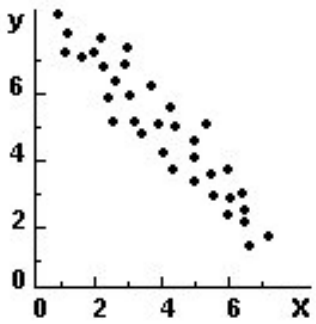

Figure 4. Example of a dispersion diagram with negative correlation (r around -1)

$$y = b_0 + b_1 x_1 + b_2 x_2 + ... + b_k x_k \tag{1}$$

To calculate a regression equation, it is necessary to examine some elements that indicate how the equation will be useful in predictions: p-value, R^2, R^2 adjusted and R^2 predicted.

The p-value is a measure of overall significance of multiple regression equation. A smaller value indicates that the equation has good significance and is useful for predictions.

The R^2 represents the coefficient of multiple determination. It is a measure of how well the multiple regression equation fits the data. A perfect fit would result in $R^2 = 1$. However, this coefficient shows a defect: as more variables are included, R^2 grows. But the best regression equation does not use the, necessarily, all variables. Because of that failure, the comparison of different models of multiple regression is made with the adjustment of the determination coefficient.

The R^2 predict value is used to indicate how well the model provides answers to new observations, while the R^2 indicates just how well the model fits the data. Predicted R^2 can prevent an overfitting model and can be better than R^2 adjusted to compare models because it is calculated using observations not included in the model estimation. Overfitting model is related to models that appear to explain the relationship between the predicts and response variables to the data used to calculate the model, but it failure to provide a valid predictions for further data.

The predicted R^2 is calculated by removing systematically each original data observation, estimating the regression equation, and determining how well the model predicts the observation removed. Predicted R^2 varies between 0% and 100%. High values suggest a model with good ability to predict.

3.3. Outliers and Residuals Analysis

In addition to the information cited in the previous item, it is necessary to examine the outliers and residuals [Montgomery et al. 2001]. Outlier is a point that is away from others and can be identified through the dispersal diagram cited in Section 3.1. The residual is the difference between an observed value and the predicted value with the regression equation.

Based on these two elements, the results provided by the regression equation should be interpreted to ensure that the model is in fact statistically reliable. The residuals must have three properties: normally distributed, constant variance and independent of each other. All these properties were applied in the case study (Section 4).

4. Case Study

This case study was made in order to find some method for estimating the development of OMA primitives in Lines Of Code (LOC) unit size. The development of those primitives were made in a previously defined framework responsible for receiving data from an application, translate this data into a XML request file, send this request file to an OMA server, receive the XML server response, and translate this response in data to be consumed by some application. The results obtained with this method not only could bring the size in LOC for the development, but based on it we could derive the development cost and effort.

The main strategy was to observe the variables related to the development of previously 30 primitives implemented. Analyzing existing data we could find that the protocol XML tags changed between the server and the client applications had some relationship, since the developed framework had to treat each tag in a determined way. So, it could be a good parameter in order to verify how existing data would fit with developed primitives. Figure 5 code listing show us the primitive *GetAttributeList-Response*, that is a response from the server when a logged user tries to see what presence attributes are public to him from another user.

After the decision about using the XML protocol tags, the next step was to verify the different types of tags that could have influence in the number of lines of code. Analyzing the OMA protocol [oma 2006] we identified the following types of tags:

- **Degenerated Tags:** tags that have no information and no content. These tags are only the change of flags from the server to the client and from the client to the server. Ex.: from Figure 5 XML file, the *GetAttributeList-Response* primitive has a degenerated tag $< OnlineStatus/ >$ that tells the user who requires what attributes he can see from another user.
- **Content Tags:** tags that have a content. These tags are sent from the server and the client to show some information. Ex.: The *GetAttributeList-Response* XML file exemple has $< Code > 200 < /Code >$ that brings the client that the transaction was successfully executed. The client code should read this value from the XML and decide what to make with it.
- **List Tags:** multivalue tags. This type of tags the client and server use to send lists of attributes. Ex.: The *GetAttributeList-Response* XML file $< PresenceList >$ tag that has a list of users and their presence attributes. This kind of tags takes more effort to be treated since they are structures that must be read and sometimes persisted.

4.1. Defining Weights for Tags

Since we have evaluated the possible tags that are changed in the protocol, for the sake of clarity we decided to define weights for the tags according to each type previously defined and the level of complexity to treat them in order to see the relation between the tags, weights, and lines of code. The tag weights were defined as follows:

```
<GetAttributeList-Response>
    <Result>
      <Code>200</Code>
   </Result>
  <DefaultAttributeList>
   <DefaultNotify>F</DefaultNotify>
      <PresenceSubList
          xmlns="http://www.openmobilealliance.org/DTD/IMPS-PA1.3">
         <UserAvailability/>
         <StatusMood/>
      </PresenceSubList>
   </DefaultAttributeList>
   <PresenceList>
      <Presence>
         <UserID>wv:another_one@nowhere.com</UserID>
         <UserNotify>T</UserNotify>
         <PresenceSubList
             xmlns="http://www.openmobilealliance.org/DTD/IMPS-PA1.3">
            <UserAvailability/>
            <StatusMood/>
         </PresenceSubList>
     </Presence>
     <Presence>
        <ContactList>wv:john/My_friends@smith.com</ContactList>
        <ContactListNotify>T</ContactListNotify>
        <PresenceSubList
            xmlns="http://www.openmobilealliance.org/DTD/IMPS-PA1.3">
           <UserAvailability/>
           <StatusMood/>
       </PresenceSubList>
    </Presence>
  </PresenceList>
</GetAttributeList-Response>
```

Figure 5. GetAttributeList-Response primitive XML file

- **Low level tags:** here we classified the degenerated tags, since they are just a flag of communication between the server and the client, and the content tags, because each content tag does not require a significant effort to be captured.
- **Medium level tags:** here we considered the tags with a list of content tags, once they require a bigger effort than the degenerated and simple content tags because the framework should read every item from the list and decide what to make with them.
- **High level tags:** here we considered the tags with a list of structures, since the framework has not only the work to treat each list, but each item of the list as well.

Another separation was done for the statistical evaluation. We considered the difference among these levels of tags for the framework request, where it sends the XML to the server, and the response, where it receives the XML from the server. This separation was defined because the framework has different steps from preparing the XML to be sent to the server, receiving the XML file from the server, and decide what to make with it. With all these definitions, we could identify six variables having influence in the number of lines of code for the development of a single primitive. These are: The *Quantity of Request Low Level Tags (ReqLowLT)*, *Quantity of Request Medium Level Tags (ReqMedLT)*, *Quantity of Request High Level Tags (ReqHighLT)*, *Quantity of Response Low Level Tags (ResLowLT)*, *Quantity of Response Medium Level Tags (ResMedLT)*, and the *Quantity of Response High Level Tags (ResHighLT))*.

4.2. Formula Calculation

Since we had six variables to find a formula that defines how to calculate the LOC for each primitive based on the found variables, we should search a technique for achieving the results. Consulting literature about software estimation and software estimation specilists we could find that the *Multipe Regression Analysis* technique would fit this purpose. In order to manipulate and make complex calculations in defined variables a tool called Minitab was used for aiding us in achieve the exepected results.

Initially was performed the analysis of correlation coefficient between each predictor variable and the amount of lines of code. The Figure 6 shows the dispersion diagram between the predictor variable that represents the amount of request tags with low complexity and the response variable that represents the amount of lines of code. The linear coefficient correlation r is 0.769 and is relatively close to number one, as shown below. This individual correlation is not so strong (equal to number one) because of, for example, the two outliers.

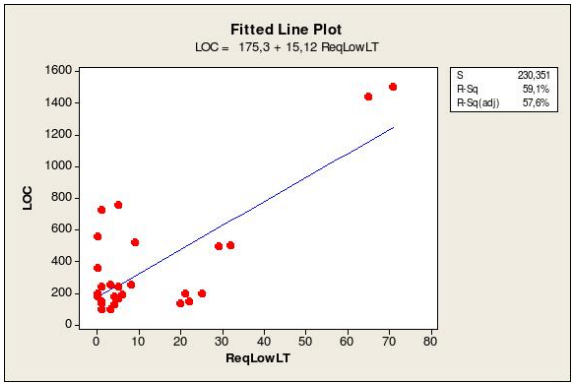

Figure 6. Dispersion Diagram between the predict variable ReqLowLT and LOC

This analysis was performed for each pair predictor variable and response variable. The Table 1 shows the summary results.

Table 1. Results of individual variables

Variable	*r* value	P-value	Alpha (5%) = 0.361
ReqLowLT	+ 0.769	0.000	$\|r\| > a$
ReqMedLT	- 0.076	0.690	$\|r\| < a$
ReqHighLT	+ 0.103	0.590	$\|r\| < a$
ResLowLT	+ 0.851	0.000	$\|r\| > a$
ResMedLT	- 0.124	0.513	$\|r\| < a$
ResHighLT	+ 0.294	0.115	$\|r\| < a$

Besides the need for a low p-value, another parameter that needs to be examined is the relationship between the r and alpha value. If the absolute r value exceeds the alpha value, concluded that there is a significant linear correlation. Otherwise, there is not still enough evidence to support that conclusion. For a sample of 30 observations, the value of alpha is 0.361, according to the statistical table of critical value of Pearson's correlation coefficient [Triola 2005].

In an individual analysis, only the variables of low complexity (ReqLowLT and ResLowLT) meets all the conditions to indicate a good correlation. However, the analysis will be continued with all the other variables, since it is also necessary to study the behaviour with all together. A strong correlation can be shown when the variables are put all together.

The multiple regression equation was calculated through Minitab [Inc. 2008] software. This equation predicts the amount of lines of code according to the independent variables described in the beginning of this section. The resulting equation is describe as follows:

$$\begin{aligned} LOC &= 117 + (5.94 \times ReqLowLT) + (2.1 \times ReqMedLT) + \\ &\quad (30.2 \times ReqHighLT) + (6.01 \times ResLowLT) + \\ &\quad (19.0 \times ResMedLT) + (31,1 \times ResHighLT) \end{aligned} \tag{2}$$

4.2.1. Results Interpretation

As described in Section 3, after determining the regression equation, it is necessary to interpret some results to ensure that the model is in fact statistically reliable. The following aspects were analyzed:

- R^2, R^2 adjusted and predict R^2 values;
- P-value;
- Outliers;
- Residuals normally distributed;
- Residuals with constant variance;
- Residuals independent of each other.

The R^2 value shows that the variables explain 94.1% of the LOC variance. The value of R^2 adjusted is 92.6%. Both are close to 100% and indicate that the model fit the data.

The R^2 predict value is 85.64%. Since it is close to the two values above, the model does not appear to be overfit and is predictive appropriate.

Predictor	Coef	SE Coef	T	P
Constant	116.69	28.55	4.09	0.000
ReqLowLT	5.938	2.059	2.88	0.008
ReqMedLT	2.09	11.40	0.18	0.856
ReqHighLT	30.244	6.258	4.83	0.000
ResLowLT	6.0081	0.9305	6.46	0.000
ResMedLT	19.02	11.21	1.70	0.103
ResHighLT	31.101	3.796	8.19	0.000

S = 96.3835 R-Sq = 94.1% R-Sq(adj) = 92.6%

PRESS = 521496 R-Sq(pred) = 85.64%

Figure 7. Minitab results for the variables and values entered

Analysis of Variance

Source	DF	SS	MS	F	P
Regression	6	3416700	569450	61.30	0.000
Residual Error	23	213665	9290		
Total	29	3630364			

Source	DF	Seq SS
ReqLowLT	1	2144637
ReqMedLT	1	57205
ReqHighLT	1	173176
ResLowLT	1	418037
ResMedLT	1	59
ResHighLT	1	623586

Figure 8. Minitab analysis of variance result

The P value (0.000) in the analysis of variance above indicates that the model estimated by the regression is significant.

This is also shown by the P-value of variables 1, 3, 4 and 6, indicating that they are significantly related to the response variable (LOC).

But it does not occur for the variables 2 and 5: values 0.856 and 0.103, respectively. This indicates that they are not well related with LOC, as already provided in individual analysis shown in the beginning of this section.

The analysis to identify if the residuals are normally distributed can be made through the chart "Normal probability plot of residuals". In this chart the points should generally form a straight line if the residuals are normally distributed. For cases in which the data are less than 50 observations, the chart should show a curve in the tails even if the waste is normally distributed.

In this case are 30 observations available and the chart shows a linear pattern consistent with the normal distribution.

The three points in blue confirm the previously identified outliers.

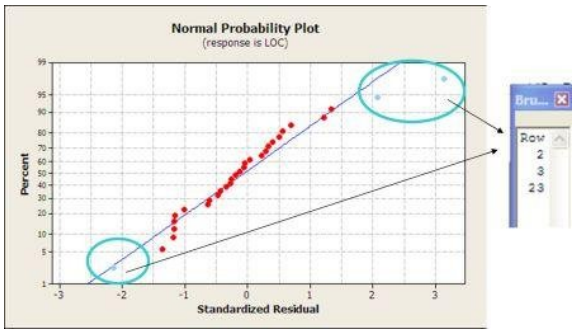

Figure 9. Normal probability plot of residuals

The graph "Residuals Versus Fitted Values" must display a random pattern of residuals on both sides of zero value. There can not be a recognizable pattern in this chart, for example, if the residual tend to rise as the adjusted values increase, then this will violate the premise of constant variance.

In this case, the graph shows that there is not a pattern between the residuals and the adjusted values. This indicates that the residuals have a constant variance.

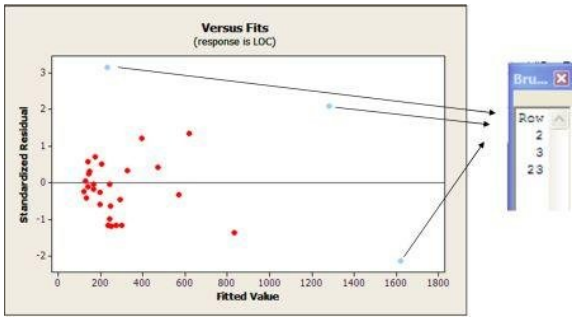

Figure 10. Residuals Versus Fitted Values

The three points in blue confirm the previously identified outliers.

The graph "Residuals Versus Order of Date" is used to find errors not randomized and helps to check the assumption that the residuals are not correlated with each other:

- A positive correlation would be indicated by a group of residuals with the same signal;
- A negative correlation would be indicated by rapid changes in consecutive residuals sign.

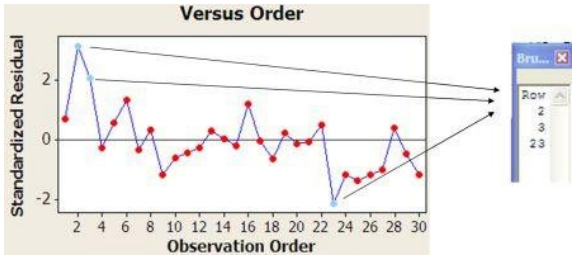

Figure 11. Residuals Versus Order of Date

As shown in Figure 11, there is not a positive residuals correlation, neither a negative correlation. The three points in blue confirm the previously identified outliers.

4.2.2. Models changes

Based on the results interpretation described in the previous section, it was necessary to make some adjustments:

- Exclusion of variable ReqMedLT due to the high value of P (0.856) which shows low relevance to the response variable;
- Exclusion of variable ResMedLT due to the high value of P (0.103) which shows low relevance to the response variable;
- Analysis for exclusion of points 2, 3 and 23 that indicates outliers.

Initially the model was adjusted to remove the two variables of complexity average (ReqMedLT and ResMedLT) and a new equation of regression was generated with the remaining 4 variables:

$$\begin{aligned} LOC &= 144 + (6.07 \times ReqLowLT) + (28 \times ReqHighLT) + \\ &\quad (5.8 \times ResLowLT) + (29.7 \times ResHighLT) \end{aligned} \tag{3}$$

The new values and interpretation are described below:

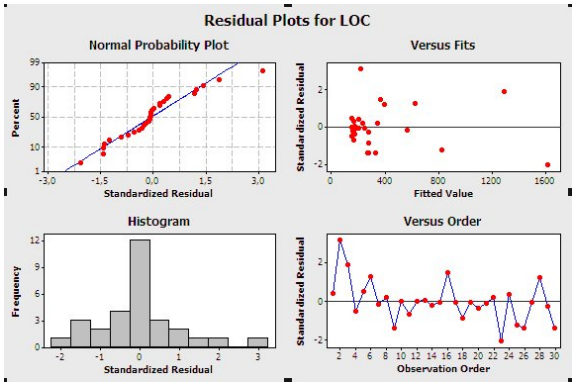

Figure 12. The new values related to the second model

- The R^2 and R^2 adjusted values had a small change and still indicate that the model fits the data in a good way (93.2% and 92.1%).
- The R^2 predict value improved a little and still indicate that the predictive model is appropriate.
- The P-value of each factor and of the equation has appropriate values (zero or very close to zero). The variables chosen are significantly related to the variable response and the estimated regression model is significant.
- The analysis of the unusual observations shows that the observation number two may indicate an outlier. But when we remove this item and regenerated the model, the new data showed no significant improvements regarding this. So, it was decided to keep all data.
- The other charts show that the residuals continue normally distributed, with variance constant and independent of each other.

For all these reasons, the second model was considered more appropriated.

5. Conclusion

Software estimation size is a historical complex problem. This complexity occurs, mainly, by the difficulty in quantify the produced result in a software project in terms of size, complexity or aggregated value to the client. More complex is to estimate this size using LOC unit, because it is in an abstraction level away of the human capability of represent the problem. For this reason, the software size estimation has been explored over the years in a way that allows an approximation of human abstraction through of creation of alternative methods to count the software size [J. 1979,

Karner 1993, Juan J. Cuadrado-Gallego 2008]. Despite this, the software costs estimative models, in general, wait by a size input in LOC[Boehm 1981, Boehm et al. 2000, Panlilio-Yap 1992], what demands the use of size in LOC during the estimation process.

An alternative mode to estimate the software size in LOC unit which is commonly cited by the literature and used by the industry is the conversion of high level size units to LOC. It occurs through pre-defined tables. An most popular example of this case is the Function Points to LOC conversion, largely studied by Jones [Jones 1995]. Despite this workaround to be very simple, it is based in average calculus of big sample of projects then is not so accurate to all systems kinds [Aggarwal et al. 2005].

As a result of this common necessity of estimate the software system size in LOC, this paper propose a method to this purpose, based in some product characteristics which is easier to quantify before the development phase. In particular the proposed method in Section 4 was based in some information existent on the problem specification, these information were quantity of each request and response tag types for each protocol primitive. Additionally two variables were removed at a second analysis and as result there was a significant improvement at the predictor model whose result is represented by the Equation 3.

Altogether the statistic use to resolve this kind of problem has been so suitable, how explained in Section 3. Moreover, the application of regression analysis for software estimation is not so new. The firsts empirics studies, based in US army software projects data, arose in mid 60's at System Development Corporation [Farr and Zagorski 1964, Farr and Nanus 1964]. After that another similar studies had happened based in same principles.

Furthermore, the resultant equation of this study can be used in other projects which has similarity with communication protocol development as was presented in Section 2. The method for defining the LOC size equation could be reused by different kinds of project if is possible identify the relevant variables involved in the final system size result.

The results of this work prove that is possible to get good responses in estimation model when regression analysis is used at historical software projects data. Moreover, for each project context is necessary to identify the main factors (predictor variables) which influences in the estimation result (response variables).

References

(2005). *WV-042 Client-Server Protocol Session and Transactions*. Open Mobile Alliance, version 1.2 edition. http://www.openmobilealliance.org.

(2006). *OMA Client-Server Protocol XML Syntax*.

Aggarwal, K. K., Singh, Y., Chandra, P., and Puri, M. (2005). An expert committee model to estimate lines of code. *SIGSOFT Softw. Eng. Notes*, 30(5):1–4.

Boehm, B. W. (1981). *Software Engineering Economics*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Boehm, B. W., Clark, Horowitz, Brown, Reifer, Chulani, Madachy, R., and Steece, B. (2000). *Software Cost Estimation with Cocomo II*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

CMU/SEI-2002-TR-012, editor (2002). *Capability Maturity Model Integration (CMMI-SW)*. SEI, 6th edition. Version 1.1.

Farr, L. and Nanus, B. (1964). Factors that affect the cost of computer programming. Technical Report TM-1447/000/02, System Development Corporation, Santa Monica, California.

Farr, L. and Zagorski, H. J. (1964). Factors that affect the cost of computer programming: A quantitative analysis. Technical Report TM-1447/001/00, System Development Corporation, Santa Monica, California.

Inc., M. (2008). Minitab statistical software. http://www.minitab.com/.

J., A. A. (1979). Measuring application development productivity. In *Proc. Joint SHARE/GUIDE/IBM Application Development Symposium*, pages 83–92, Monterey, CA.

Jones, C. (1995). Backfiring: Converting lines-of-code to function points. *Computer*, 28(11):87–88.

Juan J. Cuadrado-Gallego, Fernando Machado-Piriz, J. A.-P. (2008). On the conversion between ifpug and cosmic software functional size units: A theoretical and empirical study. *Journal of Systems and Software*, 81(Issue 5):661–672.

Karner, G. (1993). Resource estimation for objectory projects. Master's thesis, University of LinkÃ¶ping, Sweden.

Montgomery, D. C., Peck, E. A., and Vining, G. G. (2001). *Introduction to Linear Regression Analysis*. J. Willey.

Panlilio-Yap, N. (1992). Software estimation using the slim tool. In *CASCON '92: Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research*, pages 439–475. IBM Press.

Pressman, R. S. (2004). *Software Engineering: A Practitioners's Approach*. McGraw-Hill, 6th edition.

Symons, C. R. (1988). Function point analysis: Difficulties and improvements. volume 14, pages 2–11, Piscataway, NJ, USA. IEEE Press.

Triola, M. F. (2005). *Introducao a Estatistica*.