ISSN: 0976-3031

Available Online at http://www.recentscientific.com

CODEN: IJRSFP (USA)

International Journal of Recent Scientific Research
Vol. 8, Issue, 11, pp. 21653-21659, November, 2017

International Journal of Recent Scientific Research

DOI: 10.24327/IJRSR

Research Article

GLUE CODE ESTIMATION IN COMPONENT BASED SOFTWARE DEVELOPMENT PROJECTS: A TOOL BASED APPROACH

***Rajul Jain[1] and Pandey R. K[2]**

[1]Department of Computer Science, Mata Gujri Mahila Mahavidyalaya (Autonomous A+) Jabalpur (MP), India
[2]Department of Mathematics and Computer Science, Rani Durgavati Vishwavidyalaya, Jabalpur (MP), India

DOI: http://dx.doi.org/10.24327/ijrsr.2017.0811.1115

ARTICLE INFO

Article History:

Received 15th August, 2017
Received in revised form 25th September, 2017
Accepted 23rd October, 2017
Published online 28th November, 2017

Key Words:

Software Metrics, Size Metric, Component-based software systems, Glue Codes, Integration Cost, Tailoring Costs.

ABSTRACT

Estimation of effort for component based software has been in focus of researchers. Various models have been suggested by number of scientists. The most popular model is Cocots model in which major cost ingredients are integration cost i.e. the cost of glue codes, assessment costs and cost of tailoring the components. Many researchers have proposed formulas for evaluating assessment and tailoring costs theoretically. Major problem found in the existing work is calculation of integration cost. This cost cannot be evaluated through some formula or theoretical calculations as the amount of code required as glue code is not predictable using these methods. This paper proposes the evaluation of the glue code cost using UML diagrams. A Java Parser Tool has been developed to evaluate the glue code by parsing through the XMI file. For the support of the proposed system an existing UCRS system used to evaluate cost through implementation of it.

INTRODUCTION

Effort estimation of software development is an important sub-discipline in software engineering. It has been the focus of much research mostly over the last couple of decades. In recent years, software development turned into engineering through the introduction of component-based software development (CBSD). The industry has reported significant advantages in using CBSD over traditional software development paradigms. However, the introduction of CBSD has also brought a host of unique challenges to software effort estimation which are quite different from those associated with traditional software development (Wijayasiriwardhane T. et al, 2011).

Owing to the increasing tendency to use the CBSD approach in recent years, it is clear that effort estimation of CBSD is particularly an important area of research with a direct relevance to industry. (Wijayasiriwardhane T. *et al, 20*11). CBSE is a process that emphasizes the design and construction of computer based systems using reusable software components. It provides the way of developing very large software systems. It concentrates on both the Commercial-off-the-shelf and in-house components. Component based software engineering has been widely accepted as a new and latest approach to software development. Today's the software systems are very difficult, bulky and unmanageable. This causes in lesser productivity, higher risk management and greater software quality. Software metrics measure different aspects of software complexity and therefore play an important role in analyzing and improving the quality of software.

Accurately predicting software development effort is a critical concern of many organizations even today (Wijayasiriwardha - ne T. *et al, 20*11).

Underestimating development cost and schedule can have a detrimental impact on both the functionality and quality of software products and therefore on the developer's reputation and competitiveness. In extreme, it can even cause to abandon projects in the midstream. In contrast, overestimating the cost and schedule can result in a waste of resources because of redundant allocation or even a missed opportunity particularly when bidding for software contracts.

CBSD requires focus on integration-centric activities named, searching and identifying candidate components, assessing and selecting components based on system requirements and

Corresponding author:* **Rajul Jain
Department of Computer Science, Mata Gujri Mahila Mahavidyalaya (Autonomous A+) Jabalpur (MP), India

architectural and project constraints, tailoring and integrating the selected components into a seamless software system and upgrading the system as components evolve over time with newer versions.(Wijayasiriwardhane T. *et al, 20*11).

Software Models considered in study are-

Algorithmic/model-base approaches

- SAIC model
- Stutzke's model
- Ellis's model
- Aoyama's model
- ABB model
- COCOTS model

Integrated/composite models

- Adjustable cost model

Other Approaches

- Vector-based approach

Saic Model **(**Abts, C., Boehm**,** 2008, Wijayasiriwardhane *T. al, 20*11**)**

Authors have shown that Science Application International Corporation (SAIC) model mainly focuses on the end-user cost of adopting a particular component into a larger system i.e.

EC = LC x N + TC + GC

Where EC is Estimated Cost
LC is Licensing Cost
N is Number of Licenses Required
TC is Training Cost
GC is Glue Code Cost

Advantages

It enlists cost factors of components i.e. Licensing Costs & Training Costs

Disadvantage

It doesn't take the effort of searching and selecting components into account. It also does not provide details of determining the effort of the glue code development. (Wijayasiriwardhane T. *et al, 20*11).

Stutzke's Model

This model (Abts, C., Boehm *et al*, 2000) mainly focused on the volatility cost of components, which is the frequency with which a vendor releases new versions of, the components:

EAC = CV x AC x IS (CS+CC)

Where EAC is estimated additional cost, CV is component volatility cost, AC is architectural coupling cost, IS is interface size, CS is cost of screening of components and CC is Cost of changes components.

Advantage

Component Volatility.

Disadvantage

Not implemented (Wijayasiriwardhane T. *et al, 20*11).

Ellis's Model (Abts, C .*2004*, Wijayasiriwardhane T. *et al, 2011*).

Loral Federal Systems, which proposed 17 cost drivers and described the construction of an effort model for predicting the effort of component integration.

WU = fn (size, drivers)
P = LM / WU
Estimated Cost = WU x P

P is Productivity, LM is Labour Months, WU is work units &fn is a function that relates the size of glue code and ratings of cost drivers to work units.

Advantage

Ellis's model is an actual database application with a graphical user interface.

It has been calibrated to a number of CBSD projects and claims an accuracy of +/-15% against an internal dataset of Loral projects.(Abts, 2004).

Disadvantage

Deep details of the modeling function and calibration dataset remain proprietary (Wijayasiriwardhane T. *et al, 20*11).

Aoyama's Model **(**Aoyama, M., 1996, Wijayasiriwardhane T. *et al, 2011)*

In this model (Dagnino, A *et al* 2003) introduced Component acquisition, compositional design and component integration processes and wiped out unit testing process. Aoyama proposed an economic model for CBSD

$$C = \sum_{i=1}^{n} (P_i x\ Q_i)$$

Pi(Conv) = (1/n) for all processes of traditional process model
Pi(Comp) = Po (Conv) except for component acquisition process of CBSD process model
Qi(Comp) = (1-Rr) Qi(Conv)
Cost Ration = 1-[C(Conv) - C(Comp)]
=0.84 - 0.48Rr + CA

Where Pi(Conv) Cost of Conventional Software development
Qi(Conv) unit cost of Conventional Software Development
Pi(Comp) Cost of CBSD process model
Qi(Comp) Unit cost of CBSD process model
C(Conv) cost of conventional software development
C(Comp) cost of CBSD software development
CA cost of component acquisition process

Advantage

Can reduce the total development cost by 50-70% at the best effort. (Wijayasiriwardhane T. *et al, 2011).*

Disadvantage

1. Many assumptions and approximations has been done.
2. Cost of unit testing has been ignored.
   (Wijayasiriwardhane T. *et al, 2011)*

ABB Model (Dagnino, *A et al, 2003* Wijayasiriwardhane T. *et al, 2011*)

Researchers (Dagnino, A.*et al* ,2003) used goal-question-metrics approach. Researchers identified the goals for CBSD as:

Goal 1: Evaluate whether there is a reduction in cost as a result of using CBSD.

Goal 2: Evaluate whether there is a reduction in effort as a result of using CBSD.

Then questions are defined to achieve the goals and sub-questions may be defined for answering the questions. The questions are derived metric and for it they have proposed a measure called measurable units (MU) and the relation for it is as follows:

TENC = size of any fraction of system developed by custom code in MU
TER = CSize X ERLM + CSize XERMM + CSize X ERNM
TESR = CSize X EWR
TECBSD = TENC + TER + TEWR
TECustom = System size in MU

Where TENC is total effort to develop any fraction of system by new code

CSize is size of component in MU
ERLM represent the effort for reuse factors for components that require less than or equal to 25%,
ERMM represent the effort for reuse factors for components that require more than 25%,
ERNM represent the effort for reuse factors for components that require no modifications.
TER is total effort to reuse components
TEWR is total effort to write reusable components
EWR represents the effort factor for developing a reusable component
TECBSD is total effort required for CBSD
TECustom is total effort required for custom software development

Advantage

This model focuses on measuring economic benefits and performing a sensitivity analysis of the CBSD

It can be used as a comparative model to decide whether or not to go for a CBS solution

Disadvantage

Does not provide the details for determining the most central measure of the approach [9]

Cocots Model **(**Abts, C**,** *et al 2000***,** Wijayasiriwardhane T. *et al, 2011)*

It is an extension of COCOMO-II model. The model is based on two defining characteristics of the components:
The source codes of components are not available to the application developer

The future evolutions of components are beyond the control of the application developer.

It is consisting of three different sub-models that estimate the efforts of component assessment, tailoring and integration activities of the CBSD.

$$\text{CAE} = \sum(\text{CCF X MIEF}) + \sum(\text{CCA X MDAE})$$

$$\text{CTE} = \sum(\text{NCT X MTE X TCQ})$$

$$\text{CIE} = \text{A (Esize)}^{\text{B}} \prod \text{EM}$$

Total Effort = CAE+CTE+CIE
Where CAE is efforts of component assessment
CTE is efforts of component tailoring
CIE is efforts of component integration

Advantages

The estimates are done by grouping components into classes as opposed to the individual components or the system (Minkiewicz, A. *et al*, 2004)

Disadvantages

1. It does not address schedule estimation directly or allow users to customize the integration process, which can differ from organization to organization and even project to project
2. It does not account for lifecycle issues beyond initial development or post development maintenance issues for its estimation
3. Model has tendency to produce underestimates.

Adjustable Cost Model **(**Naunchan, P. *et al,* 2007, Wijayasiriwardhane T. *et al, 2011)* It is an adjustable cost model for estimating the effort and duration of the component integration. The model (Naunchan, P *et al*, 2007) integrates three approaches, namely effort multipliers of the COCOTS model to identify and determine productivity factors, system dynamics to simulate the software process and communication overhead assumptions to adjust the development team's productivity.

$$\text{EDT} = \frac{size(e + RE)}{[wf\ X\ AP\ (1 - R_{co})]}$$

$$AP = P\ X \prod PF$$

Advantage

1. Model correlates the usage of workforce to development time.
2. Model allows users to adapt the integration process pattern and specify productivity factors as appropriate for their organization
3. Model provides minimal effort needed as well as the optimal team member allocation for the component integration process.
4. It also addresses the absence of schedule estimation in the COCOTS model.

Disadvantage

1. Model covers only it presumes that the team size assigned at the beginning of a project does not change throughout its development costs of CBSD.
2. Model assumes that the size of the glue code to be written can be predictable in terms of LOC

Vector Based Approach **(**Wijayasiriwardhane T. *et al, 2011)*
This approach is to account for the increase of effort for writing wrappers or adapters by means of glue code to make the interaction assumptions of components compatible with those

of the system's architecture to which components are integrated.

In this approach (Yakimovich *et al* 1999) the interaction assumptions of individual components and systems architecture area presented as interaction vectors V = (P, C, I, S, B) where variables P, C, I, S and B represent the inter-component interaction assumptions for packaging, control information flow, synchronization and biding respectively.

Advantage

1. The information required to employ it can be easily obtained from the system's architectural description and components specification.
2. This approach can be used to decide whether or not of go for a CBS solution to select the best components and to determine the amount and type of the glue code to be written.

Disadvantage

Only provides comparative results for the component integration but not the required integration effort in terms of any measurable unit such as man hours. (Wijayasiriwardhane T. *et al, 2011*)

Problem Statement

COCOTS model provides a great details of the existing system in the area of effort estimation for CBSE. Particularly the effort estimation in before implementation, during implementation and post implementation has been focused by the researchers. The major costs contributing in effort estimation for CBSE includes:

1. Assessment Cost of proper and selected components
2. Tailoring Cost of the components to fit to the requirements of the new system
3. Integration Cost for Components

This is found from the existing system that evaluation of the assessment cost is very much dependent on the new system. Similarly tailoring cost is fully dependent on the new system. Researchers have proposed formulas for evaluating assessment and tailoring costs theoretically. Major problem found in the existing works is calculation of integration costs. This cost cannot be evaluated through some formula or theoretical calculations as the amount of code required as glue code is not predictable using these methods. This works proposes to evaluate the glue code cost using UML diagrams. For the support of the proposed system UCRS (Jawwad W.Shareef *et al* 2012) has been implemented to evaluate LOC metric.

Proposed System

Integration cost is the cost of generating glue code for adding two or more components to acquire a new working system. This includes cost of integrating any existing system with the new components to be used and cost of adding two or more new components together. This cost is found to be most difficult in calculating majorly when the cost and effort are to be calculated before the inception of the project implementation. In this research, focus is to evaluate the complete effort required in implementation of any new software system. The effort estimation in this work is pre implementation so that the organization can predict the cost beforehand. The cost of assessment and tailoring are being taken from the formula and cost of integration through generation of the glue code is being calculated by analyzing the UML diagrams through XMI files. The complete work is provided with the use of case study of UCRS detailed in next section. The steps used in the work are as follows:

Step 1: Case Study of UCRS (Jawwad W.Shareef *et al* 2012) has been taken and deployment diagram of the same has been drawn using the ArgoUML tool available over the internet.

Step 2: After drawing the deployment diagram XMI file of the same has been generated using ArgoUML export to XMI option. (About XMI files, its usage and advantages have been elaborated in the coming section)

Step 3: A Java Parser tool has been implemented to parse through the XMI file and collected information from XMI file such as components, interfaces, operations, parameters etc.

Step 4: The collected information has been analyzed and processed to evaluate amount of glue code required for each interface based on parameter processing detailed in section followed.

Step 5: The amount of glue code values has been applied with the weights to reduce the inaccuracies

Step 6: The average of the weighted glue code amount for all interfaces is taken to calculate the final expected occurring if any. glue code effort required to implement the system.

Step 7: Total effort has been evaluated using the formula provided in COCOTs model

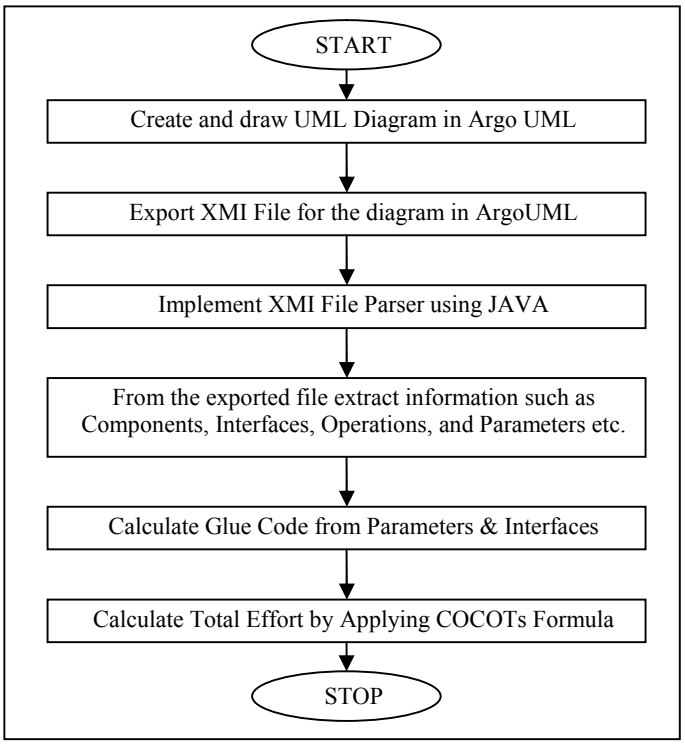

Figure 1 Flow of the proposed system

Glue Code Calculation

Glue code is measured in this work as lines of code required to integrate two components based on per interface per parameter.

For including the possibilities all parameters are taken to be both single values parameters and multi-valued parameters. The assumption is justifiable as during the integration a component cost of the single values parameters is taken as follows:

Every single value will be either assigned to receiving parameter directly which will include a single assignment statement. If type conversion is required then every language provides a set of statements for converting types of values, which is found to be between 2-3 additional statements e.g., in JAVA we have wrapper classes to convert types which requires two statements to convert from one type to another. Most involved type conversion is from String type to any primitive type which requires parsing of String i.e. O(c) processing time where c is number of digits, hence requiring n number of statements in conversion.

The processing may need to receive a simple value or a compound value from other components. Since evaluation of these parameters might be complex therefore it is being assumed further that the component implementation handles these complexities during further processing but do not provide any type of conversions required between two components in respects of values and types. For reducing the possible error, weighted processing has been considered.

The processing cost of compound values is a combination of the ingredients of the compound types which themselves are single values or compound values. Therefore cost of these can be k * cost of single values where k is number of single values involved in each compound values.

After summing up we get the following formula for evaluating the glue code cost:

Glue Code Cost

n1 + n2*c + n3 * s + k1 + k2 * s + k3 * c;

Where
n1 is number of single values parameters
n2 is number of single values requiring type conversion
c is cost of conversion
n3 is number of single values requiring conversion from String to numeric values
s is cost of conversion from String to numeric
k1 is number of single values requiring no conversion and involved in multi-values parameter
k2 is number of single values requiring conversion and involved in multi-values parameter
k3 is number of single values requiring conversion from String to numeric and involved in multi-values parameter

Application of weight mechanism:

Since glue code cost evaluated using parameters may involve some complex processing during assignment on receiving parameters therefore we need to apply some weight to reduce the possible error. The weight value should be higher than one always and must be kept as small as possible. As the weight is increased the value is considered to be more on assumption based then on the actual cost i.e.
Weight $1/\alpha$ Accuracy

Calculation of weight in this work is being done again on the basis of the number of parameters and it is being found that as the numbers of parameters are increased weight value is reduced.

About Argouml & XMI

ArgoUML is an UML diagramming application written in Java and released under the open-source Eclipse Public License. By virtue of being a Java application, it is available on any platform supported by Java SE.

XMI is a compressed file format created by XMill. XMI files are XML files, which usually contain metadata information, which have been compressed. XMI stands for "XML Metadata Interchange." While .XML files tend to be very large, the compression used by XMill into .XMI files will make the files around half the size of other compression techniques. XMI files can be decompressed using XMill or similar XML compression software.

About UCRS

UCRS (Jawwad W.Shareef *et al* 2012) is a automation system for the universities and provides various facilities to the students, faculties and staff. Within this system, a student registers for classes. Once given access, the students may select a term and build a class schedule from the offered classes. The system passes information about a student's schedule to the billing system. A student can also register, add, or drop a course. An instructor may use the registration system to print a student class list and to submit grades for her/his class. The administrator may maintain student and teacher information.
This model provides an overall view of the system and helps to demonstrate the extraction of existing component assembly complexity metrics.

The component, Registration System, has seven provided interfaces namely, IMakeSchedule, IUpdateSchedule, IRegisterCourse, IView Result, ISubmitGrades and ILogin which in ArgoUML tool are linked by an arrow known as (Abstraction). Similarly there are four required interfaces of component 'Registration System', linked by an arrow known as (Dependency). These required interfaces serve as provided interfaces for the following.

- ICourseMgt by component 'Course Management'
- ITermMgt by component 'Term Management'
- IPersonMgt by component 'Person Management'
- IBillMgt by component 'Billing System'

After the modeling of UCRS(Jawwad W. Shareef *et al* 2012) is completed, the metrics are derived using ArgoUML tool, the XMI 1.2 file is generated with the help of Export XMI option (ArgoUML using Netbeans XMI Writer version 1.0),. Using this XMI file, the metrics are derived by parsing the XMI 1.2 file.

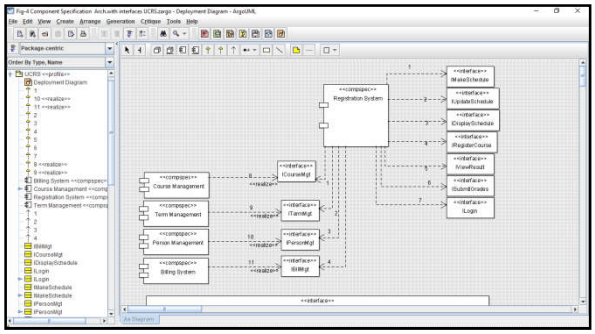

Figure 2 Component Diagram of UCRS Registration System (Mehmood& Lai, 2006)

The UCRS model in XMI is identified by a unique id (UML:Modelxmi.id). The XMI file contains information of all components by assigning a unique (UML: Component xmi.id) to each component.

The component provided and required interfaces are shown as a link pointed to a stereotype <<interface>>, here in XMI file the component which provides an interface to other components is identified by (UML: Dependency.client) by assigning a unique (UML: Componentxmi.idref)to each component, the link which carries this dependency to the stereotype<<interface>> is identified by (UML: Abstraction) assigning a unique(xmi.idref), similarly for a required interface of a component the link whichcarries this dependency to the stereotype <<interface>> is identified by(UML: Dependency. supplier) in the system.

The XMI files stores all necessary information regarding UCRS model (Jawwad W.Shareef *et al* 2012). This file is parsed through Java Parser tool developed with the help of ArgoUML parser; to derive different metrics related to component assembly, using a Java API tool.

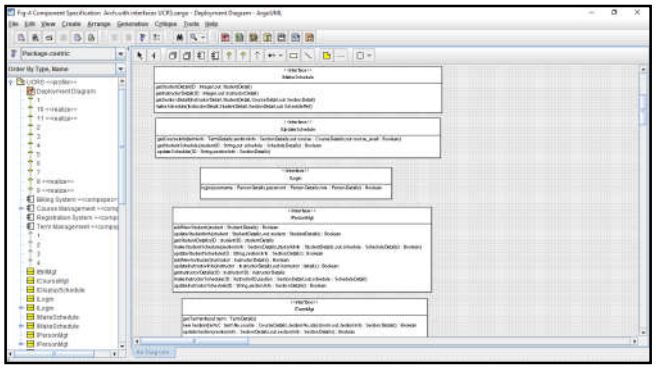

Figure 3 Various Interfaces with Operations and Parameters of UCRS Registration System (Mehmood& Lai, 2006)

IMPLEMENTATION AND RESULTS

A JAVA based application has been developed to evaluate the glue code for the case study of UCRS [(Jawwad W.Shareef *et al* 2012) to estimate the cost of glue code.

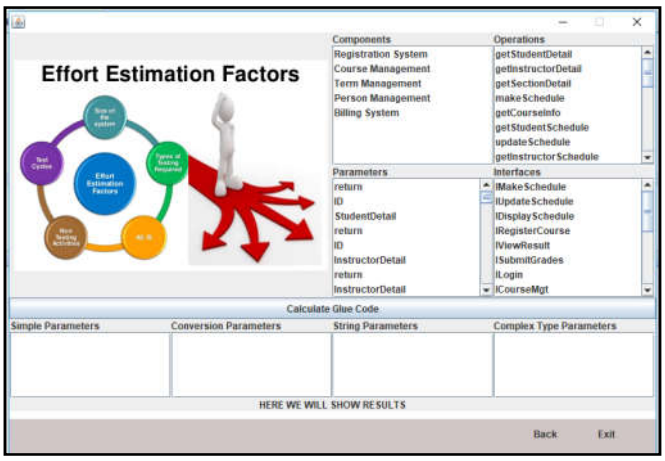

Figure 4 Java Tool showing the various Components, Interfaces, Operations and parameters in the exported XMI file.

The implementation has been done to list all the components, interfaces, operations and parameters in the system by parsing the XMI file. As proposed in section above for evaluation of glue code formula, various statistics regarding the parameters, their types and their requirement of lines of codes has been calculated as stated.

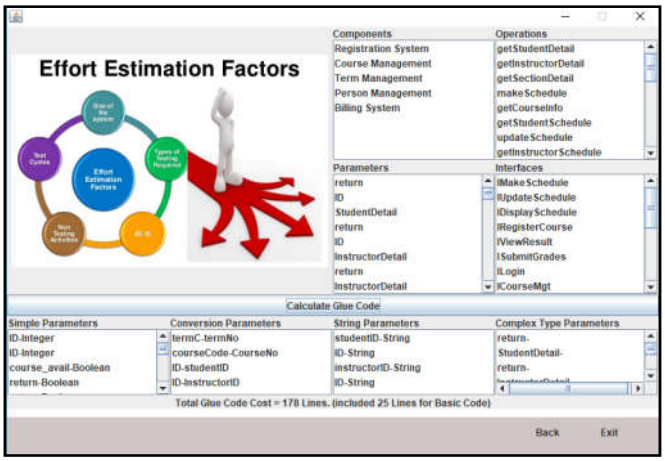

Figure 5 Java Tool showing the various types of parameters evaluated from the parsing of XMI file and estimated lines of code required for glue code generation for the components of the registration system of UCRS.

From the figure 5 above, it is shown that estimated lines of code to be written for integration of the components of registration system of UCRS (Jawwad W.Shareef *et al* 2012) are 178. The implementation provides an idea of tentative cost of glue code before actual implementation of the system. This is useful in calculating the tentative overall cost of the system in advance and helpful in making organizational decisions.

CONCLUSION

Calculation of glue code cost i.e. the cost of integration has been proposed to be based on number of parameters involved in the various operations of the components of the system. The parameters have been categorized in three categories and method has been proposed to calculate the cost. The case study of UCRS based calculation in this work evaluated glue code cost in lines of codes, which can be used to calculate the efforts using the existing methods. The categories of the parameters

have been done to match the most practical requirements of the implementation and hence after weighted adjustments cost is found to be realistic. The glue code cost can be combined with assessment and tailoring cost to get the full effort. The work is providing the estimation before actual implementation and therefore they are going to be helpful in pre-implementation estimations of the software.

Future Research

- The tool developed work for only component models developed in Argo UML, this can be further upgraded for other UML tools like Rational Rose, Magic Draw UML, etc.
- The tool can be further upgraded for estimation of Assessment cost and tailoring cost for component-based systems.
- Other metrics related to Component-based systems can be included in enhanced version of the tool proposed.

References

Abts, C.: 'Extending the COCOMO II software cost model to estimate effort and schedule for software systems using commercial-off-the-shelf (COTS) software components: the COCOTS model'. PhD thesis, University of Southern California, 2004

Abts, C., Boehm, B.W.: 'COTS software integration cost modeling study' (Centre for Systems and Software Engineering, University of Southern California), http://sunset.usc.edu/csse/TECHRPTS/1998/usccse98-520/usccse98-520.pdf, accessed August 2008

Abts, C., Boehm, B.W., Clark, E.B.: 'COCOTS: a COTS software integration lifecycle cost model - model overview and preliminary data collection findings'. Proc. 11th European Software Control and Metrics Conf. and Third Software Certification Programme in Europe,

Abts, C., Boehm, B.W., Clark, E.B.: 'Empirical observations on COTS software integration effort based on the initial COCOTS calibration database'. Proc. Second Workshop on Commercial Off-The-Shelf Software - in Conjunction with 22nd Int. Conf. on Software Engineering, (ICSE 2000), Limerick, Ireland, 2000, pp. 99-104

Albrecht, A.J., Gaffney, J.E.: 'Software function, source lines of code and development effort prediction: a software science validation', IEEE Trans. Softw. Eng., 1983, 9, (6), pp. 639-648.

Aoyama, M.: 'A component-based software development methodology ', IPSJ SIG Notes, 1996, 96, (84), pp. 33-40

Aoyama, M.: 'Process and economic model of component-based software development: a study from software CALS next generation software engineering program'. Proc. Fifth Int. Symp. on Assessment of Software Tools, (SAST '97), Pittsburgh, PA, June 1997, pp. 100-103

Boehm, B.W., Clark, B., Horowitz, E., Westland, J.C., Madachy, R.J., Selby, R.W.: 'Cost models for future software life cycle processes: COCOMO 2.0', Ann. Softw. Eng., 1995, 1, (1), pp. 57-94

Dagnino, A., Srikanth, H., Naedele, M., Brantly, D.: 'A model to evaluate the economic benefits of software components development'. Proc. Int. Conf. on Systems, Man and Cybernetics, October 2003,pp. 3792-3797

Ellis, T.: 'COTS integration in Software solutions - a cost model'. Proc. Fifth Int. Symp. Int. Council on Systems Engineering (INCOSE) -Systems Engineering in the Global Marketplace, St. Louis, MO, 1995,pp. 170-177

Fenton, N.E., Pfleeger, S.L.: 'Software metrics: a rigorous and practical approach' (PWS Publishing, 1998)

Hastings, T.E., Sajeev, A.S.M.: 'A vector-based approach to software size measurement and effort estimation', IEEE Trans. Softw. Eng.,2001, 27, (4), pp. 337-350

Jawwad W. Shareef, Pandey Rajesh Kumar, "Dependency Analysis using UML for Component Based Software System: an XMI Approach", International Journal of Computer Applications (0975-888), Vol. 47, No. 18, June 2012

Karpowich, M., Sanders, T., Verge, R.: 'An economic analysis model for determining the custom versus commercial software tradeoffs,' in Gulledge, T.R., Hutzler, W.P. (Eds): 'Analytical methods software engineering economics', (Springer-Verlag, 1993), pp. 237-252

Lai, R., Huang, S.J.: 'A model for estimating the size of a formal communication protocol specification and its implementation', IEEE Trans. Softw. Eng., 2003, 29, (1), pp. 46-62

Mahmood, S., Lai, R., Kim, Y.S., Kim, J.H., Park, S.C., Oh, H.S.: 'A survey of component based System quality assurance and assessment', Inf. Softw. Technol., 2005, 47, (10), pp. 693-707

Minkiewicz, A.F.:'Are software COTS solutions an Affordable alternative'.Proc. Aerospace Conf Piscataway, NJ, March 2004, pp. 4073-4082.

Naunchan, P., Sutivong, D.: 'Adjustable cost estimation model for COTS-based development'.Proc. 18th Australian Software Engineering Conf., (ASWEC 2007), Melbourne, Victoria, April 2007,pp. 341-348.

Reifer, D.J., Basili, V.R., Boehm, B.W., Clark, B.: 'Eight lessons learned during COTS-based systems maintenance', IEEE Softw., 2003,20, (5), pp. 94-96

Stutzke, R.: 'Costs impact of COTS volatility'. Proc. Knowledge Summary: Focused Workshop on COCOMO 2.0, Los Angeles, CA, 1995.

T. Wijayasiriwardhane, R Lai, K.C. Kang, "Effort estimation of component-based software development - a survey", IET Software, 2011, Vol. 5, Issue 2, pp. 216-228 doi: 10.1049/ietsen. 2009.0051

Verner, J.M., Tate, G.:'A software size model', IEEE Trans. Softw. Eng., 1992, 18, (4), pp. 265-278.

Yakimovich, D., Bieman, J.M., Basili, V.R.: 'Software Architecture classification for estimating the cost of COTS integration'. Proc.21st Int. Conf. on Software Engineering, Los Angeles, CA, May 1999,pp. 296-302.

****