Volume 3, Issue 6, June 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Cost Estimation Model for Reuse Software

Mayank Mandloi
Scholar –M.Tech(IT)
Dept. of Information Technology
PCST, Indore India

Prof. Sachin Patel
Head of Dept. IT
Dept. Of Information Technology
PCST, Indore India

Prof. Rakesh Pandit
Proffesor
Dept. of Information Technology
PCST, Indore India

Abstract : Software cost estimation is the way of predicting the effort required to develop a software system. In this process we include software reuse concept which provide the the amount of code reused in a product and estimated the cost to develop the software. In this concept we estimate cost of reuse code which is reduce the cost of software during development and maintenance. To measure the costs involved in reuse programs and to evaluate the impacts of reuse in software development.

Index Terms—Software Reuse, Reuse Survey, Reuse metrics, Reuse cost

I. INTRODUCTION

"Actually it is hard to see how much time or budget u have save during the project developing. it is also quite impossible to set up the goal and timetable about the reuse work in the beginning of the project"[1].Cost savings is the most promoted benefit for reuse, but benefits also exist in risk, schedule, and performance [2]. Software reuse have the same cost and risk features as any financial investment [3].

1.1 Prerequisites for Cost

An organization have to analyze following factors related to cost before implementing a reuse program.

- Cost to implement a reuse program.
- Investment approach
- Cost to sustain the reuse program.
- Time to take to break even reuse investment.
- Expected benefits

1.2 Investment approaches

There are three investment approaches for software reuse process:

- Proactive Investment

This approach tends to require a large initial investment and is preferred in stable domain [5]. Returns on investment can only be seen when products are developed and maintained .

- Reactive Investment

This is an incremental approach to build reusable components generally used in reengineering and is preferred in unstable domain [4] .

- Extractive Investment

Combination of Proactive and Reactive approach .

1.3 Cost Factors for Reuse

- Management

This cost is spent to manage a reuse program. Reuse is as much of a technical issue as it is an organizational one [5].

- Development and implementation

This cost is spent to develop and implement a reusable program.

- Operation

This cost is spent to reuse a component.

- Investment for Support

This cost is spent to support a reuse program such as training and tool acquisition.

II. Accurate software cost estimates

Accurate software cost estimates are critical to both developers and customers[6]. They can be used for creating request for proposals, contract negotiations, scheduling, monitoring and control. Underestimating the costs may result in management approving proposed systems that then exceed their budgets, with underdeveloped functions and poor quality, and failure to

complete on time. Overestimating may result in too many resources committed to the project, or, during contract bidding, result in not winning the contract, which can lead to loss of jobs.

Accurate cost estimation is important because[6]:

- It can help to classify and prioritize development projects with respect to an overall business plan.
- It can be used to determine what resources to commit to the project and how well these resources will be used.
- It can be used to assess the impact of changes and support replanning.
- Projects can be easier to manage and control when resources are better matched to real needs.
- Customers expect actual development costs to be in line with estimated costs.

Software cost estimation involves the determination of one or more of the following estimates:

- Effort (usually in person-months)
- Project duration (in calendar time)
- Cost (in dollars)

Most cost estimation models attempt to generate an effort estimate, which can then be converted into the project duration and cost.[6] Although effort and cost are closely related, they are not necessarily related by a simple transformation function. Effort is often measured in person-months of the programmers, analysts and project managers. This effort estimate can be converted into a dollar cost figure by calculating an average salary per unit time of the staff involved, and then multiplying this by the estimated effort required. Practitioners have struggled with three fundamental issues [6]:

- Which software cost estimation model to use?
- Which software size measurement to use – lines of code (LOC), function points (FP), or feature point?
- What is a good estimate?

In the last three decades, many quantitative software cost estimation models have been developed. They range from empirical models such as Boehm's COCOMO models [7] to *analytical* models such as those in [10, 9, 8]. An *empirical model* uses data from previous projects to evaluate the current project and derives the basic formulae from analysis of the particular database available. An *analytical model*, on the other hand, uses formulae based on global assumptions, such as the rate at which developer solve problems and the number of problems available. The accuracy of size estimation directly impacts the accuracy of cost estimation.

Although common size measurements have their own drawbacks, an organization can make good use of any one, as long as a consistent counting method is used.

A good software cost estimate should have the following attributes [11]:

- It is conceived and supported by the project manager and the development team.
- It is accepted by all stakeholders as realizable.
- It is based on a well-defined software cost model with a credible basis.
- It is based on a database of relevant project experience (similar processes, similar technologies, similar environments, similar people and similar requirements).
- It is defined in enough detail so that its key risk areas are understood and the probability of success is objectively assessed.

Software cost estimation historically has been a major difficulty in software development.

Several reasons for the difficulty have been identified:

- Lack of a historical database of cost measurement.
- Software development involving many interrelated factors, which affect development effort and productivity, and whose relationships are not well understood.
- Lack of trained estimators and estimators with the necessary expertise.
- Little penalty is often associated with a poor estimate.

III. Process of estimation

Cost estimation is an important part of the planning process. For example, in the top-down planning approach, the cost estimate is used to derive the project plan:

1. The project manager develops a characterization of the overall functionality, size, process, environment, people, and quality required for the project.
2. A macro-level estimate of the total effort and schedule is developed using a software cost estimation model.
3. The project manager partitions the effort estimate into a top-level work breakdown structure. He also partitions the schedule into major milestone dates and determines a staffing profile, which together forms a project plan.

The actual cost estimation process involves seven steps [7]:

1. Establish cost-estimating objectives.
2. Generate a project plan for required data and resources.
3. Pin down software requirements.

4. Work out as much detail about the software system as feasible.
5. Use several independent cost estimation techniques to capitalize on their combined strengths.
6. Compare different estimates and iterate the estimation process.
7. After the project has started, monitor its actual cost and progress, and feedback results to project management

No matter which estimation model is selected, users must pay attention to the following to get best results:

- Coverage of the estimate (some models generate effort for the full life-cycle, while others do not include effort for the requirement stage).
- Calibration and assumptions of the model
- Sensitivity of the estimates to the different model parameters
- Deviation of the estimate with respect to the actual cost.

IV. Software reuse cost factors

During development, the cost factors of reusable components are divided into seven categories, described below.

1. **Identification and acquisition costs:** Before searching for reusable components, the producer must develop a complete description of the product and identify the requirements and environment characteristics from the consumer[12]. With these requirements, the producer develops components generically in order to allow for future reuse.[12].

   Additional costs for mining and acquiring reusable assets are: 1) the cost of technical staff and consultants in identifying needed components for an application; 2) any costs incurred in making a reusable component or system function properly in order to evaluate its potential reuse in the new application (including media conversions, implementation differences, non-current documentation, and cost of existing functionality evaluation for potential reuse); 3) the creation of a design document for planning and implementation of components reuse; and 4) the purchase price and maintenance fee acquired from outside the company [13,14,15,16,17]. Assets may be transitioned for reuse in the following ways: 1) buying a copy of a repository component for a certain product and making changes to it via white-box reuse (also known as cataloged asset acquisition); 2) buying a copy of a repository component and using it without making changes (known as black-box reuse); 3) buying an existing component from another product (known as mining and cataloging); 4) using a copy of a component that is not cataloged in the repository but that an employee has knowledge of (copy and paste); and 5) purchasing a component from another organization and adding it to the repository (external acquisition)[18].

2. Modification costs: During modification for reuse, there are two methods that may be used in the transformation of repository assets: 1) adaptation for reuse, which is the modification of an existing repository asset; and 2) white-box reuse, which involves the revision of one asset into another asset inside the same application. Asset modification costs include: 1) additional effort required to modify other artifacts in order to integrate a reused component into the new product when white-box, black-box, or commercial off-the-shelf software is reused.; 2) the cost of interface design between the application and the repository asset - all functions of the reusable component and all attributes that make up the interface between the application and the reusable component must be identified and specified properly in order to achieve the desired functionality; and 3) the cost of reformatting data when an application is migrated from one platform, database management system, or operating system to another. For instance, a reusable component developed in C# and uses Microsoft Access database. The new application is developed in Java using an Oracle database, so the changes required for component reuse may require a great deal of effort [13,16].

3. **New development costs:** Development costs of new assets are also involved in software reuse. These costs fall into two categories: producer and consumer. Producers construct a new repository asset from scratch in a manner in which the asset will conform to specified standards that allow for reusability, and consumers incorporate the reusable assets developed by producers into software components. Consumers may develop new components as needed in order to integrate reusable software into their application [19]. When finding needed components in the reuse repository, the consumer must decide if any additional components are needed, if it is viable to build the product from the reusable components, the code needed for component integration, and the additional code needed to satisfy the requirements. If reuse is economical, the consumer chooses the necessary reusable components, codes additional required components, assimilates the retrieved components as black box reuse or white box reuse, and develops integration code to make the components work together [20].

4. **Integration and testing costs:** Product integration costs include: 1) the cost of partial and full integrations; 2) the cost of data transfer from a previous application to a new application in order to verify and test a component in preparation for reuse; and 3) the cost of design reviews which require effort to review a document and prepare an abstract. Integration costs also include verification and validation. activities (i.e., technical design reviews, formal code walkthroughs, and unit test plans) that are involved in software reuse coding costs of new components as well

as the same costs performed directly on the reused assets[18]. Product testing costs include: 1) the development of a test environment; 2) unit testing and debugging; 3) acceptance testing; 4) subsystem and system testing; and 5) testing of functionality by a quality control engineer [6,9]. The interaction of the reused component with the system must be thoroughly tested to guarantee that the functionality is performing as expected. After development of the entire product, both the producer and the consumer perform tests, potentially reusing test cases and test data, to determine whether the functionality was built according to specification and validate the software to determine if the system accomplishes the intended goal [20].

5. **Infrastructure costs**: Prior to instating a reuse program, a new development process implementing reuse and a reuse repository must be established [2]. Costs to establish and maintain the repository include: 1) database analysis and design; 2) the cost of tool development or purchase, which requires textbook or online training in the new technology; and 3) the cost of database administration. The cost to store and catalog repository artifacts includes 1) the cost of the time required for the approval of artifacts for the repository; 2) the cost of analyzing the metadata needed in order to employ efficient searches of artifacts in the catalog; and 3) the cost of a mechanism for the retrieval of assets from the catalog. In the storage phase, the producer must classify and store assets that will be maintained in a repository for consumer retrieval [20,13,15]. The actual development cost of the software reuse cannot be determined precisely. Some costs may not apply since the catalog rarely provides a complete product; however, multiple assets may be constructed into a complete product in which several assets have been reused. The cost of development of such an asset may be the sum of its integration, verification, and validation costs[18].

V. Reuse cost estimation

When the development of a new software project begins, it is necessary to compare the cost of developing new components versus integrating reusable components. To determine the cost of reuse, several cost estimation models have been developed. In this work, we focus on six common models, briefly described below[18].

1. **COCOMO-II model (Boehm):** The original COCOMO model predicts the length and effort of a project by drawing an association between the size of the system and various cost factors. The factors were weighed based on the project's domain, environment, and limitations in order to convert the estimated project size into estimated person-months which could be broken down into staff size and project length. In addition, the equations include fifteen cost drivers for each phase of a project (product design, detailed design, coding/unit test, and integration test). In 2000, the model was extended to COCOMO-II, which presented three sizing. options: object points (used during the application composition model), function points (used during the early design model), and lines of source code (used for the post-architectural model) [16,21,22].

2. **Reuse-Based model (Jasmine/Vasantha):** The model is based on reuse cost metrics. Metrics are considered important in deciding whether to modify a reusable component, and they can be used to better measure, manage, and plan software application development. Such metrics may give engineers the information they need to make decisions in technical areas, and they may give management the information needed in making decisions regarding project planning. The two categories of metrics are: 1) product metrics that establish component characteristics and 2) process metrics that involve measurements of cost and time among other things. The model considers many cost factors associated with reusable components, including domain analysis, explicit documentation to increase the ease of implementing a reusable component, maintenance of and revisions to of reuse documents and components, costs in the form of licenses and royalties for artifacts acquired from other organizations, purchase, installation, and operation of a reuse repository, and costs associated with training personnel in reuse design and incorporation. Management should also consider the number of times a component is expected to be reused when building reusable components [4].

3. **Basic-Reuse model (SPC):** This basic reuse costing model [23] was developed by the Software Productivity Consortium (SPC). The model takes into account the following csot factors: the cost of software development, the relative cost to reuse software, the proportion of reused code in the software, the relative cost of developing a reusable asset, and the number of reuses over which the asset development costs will be amortized.

4. **Rate-Based model (Intermetrics):** This ad-hoc method is based on the experience of Intermetrics, Inc. Under this model, much of the reuse cost has to do with the human energy and time spent in searching for code and retrieving enough of the necessary reusable components required to work together to produce the desired functionality as well as determining the condition in which the reusable software components are found. From its experience in the development of a reusable software system, Intermetrics identified the following cost factors: data transfer, data reformatting, document review, abstract preparation, facet and keyword preparation, configuration management, reusable component testing, environment testing, outside resource personnel, and consulting[18].

5. **Risk-Based model (Lim):** This model determines reuse value by adding total of reduced and avoided consumer costs to the greater profit generated by software reuse and subtracting out the producer costs. This number is multiplied by a probability that factors in risk and adjusts the result in order to consider the value of money over time. The model utilizes the following cost factors: cost to create product without reuse, cost to create product with reuse, cost to create an asset for reuse, profit from increased revenues enabled by reuse, probability of receiving cash flow, interest rate by which cash flows are discounted, number of time periods under consideration, and net Present Value [19].

6. **Product-Line model (Tomer):** A software product line is a group of software applications using a shared set of components that meet specific requirements of a certain application domain. The product line approach achieves considerable savings and is economical due to the reuse and integration of related components from the common asset library [24]. In this approach, transformation and transition are involved in the cost of a reusable component. The transformation operation's cost consists of the cost required to modify the reusable component; while the transition operation's cost is made up of the cost incurred in copying code from one component into another in the application or black-box reuse [13]. The model uses these cost factors: cost of adaptation for reuse, cost of development from scratch, cost of white box reuse, cost of constructing the target private artifact from scratch as new development, cost of catalog acquisition (replicating a copy of repository), cost of copy and paste, cost of mining, and cost of externally acquiring. In addition, the calculations of reuse costs depend upon the reuse scenario put into place, such as Systematic Reuse Cost for adapting assets from artifacts mined during domain analysis; Systematic Reuse Cost for new assets developed from scratch; Controlled Reuse Cost; Opportunistic Reuse Cost; and Pure Development Cost. In order to determine the most cost-effective reuse scenario that should be used in the production of the desired functionality from the original source assets, one must be able to compare the costs of other possible reuse scenarios. Reuse scenarios are any sequence of transition and transformation operations performed while practicing reuse [23,24].

VI. Compare of estimation models

Here we compare the models analyzed in this work, Table-1 depicts a list of common cost factors. The cost factors from each model are linked with the common cost factors in Table-2 in order to portray the similarities and differences in each model. To be clear, a producer is a programmer who creates reusable components from scratch while a consumer is a programmer who uses reusable components to create other applications.

Table-1: Cost Factors of Reuse Cost Estimation Models (adapted from [19]).

QUANTITIES		**LINES OF CODE**	
NR	Number of reuses	ASLOC	Adapted source lines of code in project
CM	Percentage of code modified	RLOC	Reused lines of code in product
DM	Percentage of design modified	RLOCL	Reusable lines of code in library
IM	Percentage of integration effort required	PLOC	Lines of code in product
CONSUMER		**PROFIT**	
Cc,wrp	Cost to create product without reuse	P	Profit from increased revenues enabled by reuse
Cc,rp	Cost to create product with reuse	**RISK**	

Cc,mwr	Cost to maintain product created without reuse	p	Probability of receiving cash flow
Cc,mr	Cost to maintain product created with reuse	pl	Probability of asset being found in library
Cc,ra	Cost to consumer to reuse asset	**TIME VALUE**	
Cc,r1	Cost of royalties and licenses for external assets	i	Interest rate by which cash flows are discounted
Cc,r2	Cost to adapt asset	M	Number of time periods under consideration
Cc,r3	Cost to acquire, generalize, search, and retrieve	**PRODUCER**	
Cc,r4	Cost to utilize assets: instantiation training	Cp,r	Cost to producer to create asset for reuse
Cc,r5	Cost to reuse: identify, retrieve, understand, validate, integrate, and test an asset	Cp,lm	Cost to producer/maintain reusable assets in the library; configuration and change management
Cc,r6	Cost of incentive paid to component developers	Cp,wr	Cost to create non-reusable version of asset
DOCUMENTATION		**OVERHEAD**	
DD	Documentation development	O1	Library overhead costs
DU	Documentation maintenance	O2	Cost of Domain Engineering
OUTPUT		O3	Infrastructure costs -

			repository mechanisms
E	Effort in programmer months	O4	Infrastructure costs - domain analysis, architecture, training
C		Cost of effort	
NPV		Net present value	
SZ		Project size – LOC, object, or function points	

Table-2: Comparison of Software Reuse Cost Models by Cost Factor.

	COCOMO-II (Boehm)	**Reuse-Based (Jasmine/ Vasantha)**	**Basic-Reuse (SPC)**	**Rate-Based (Intermetrics)**	**Risk-Based (Lim)**	**Product-Line (Tomer)**	
Cost Factor							**Total**
Quantities							
NR			**Y**				**1**
CM	**Y**						**1**
DM	**Y**						**1**
IM	**Y**						**1**
Consumer							
Cc,wrp		**Y**			**Y**		**2**
Cc,rp					**Y**		**2**
Cc,mwr		**Y**					**0**
Cc,mr							**1**
Cc,ra			**Y**				**2**
Cc,r1							**0**
Cc,r2	**Y**	**Y**					**3**
Cc,r3							**1**
Cc,r4							**0**
Cc,r5	**Y**	**Y**		**Y**			**4**
Cc,r6							**0**
Documentation							

DD				**Y**			***1***
DU				**Y**			***1***
Lines of Code							
ASLOC	**Y**						**1**
RLOC	**Y**		**Y**				**2**
RLOCL							**0**
PLOC	**Y**		**Y**				**2**
Profit							
P					**Y**		***1***
Risk							
P					**Y**		***1***
pl		**Y**					***1***
Time Value							
i					**Y**		***1***
M					**Y**		***1***
Producer							
Cp,r			**Y**		**Y**	**Y**	***3***
Cp,lm				**Y**			***1***
Cp,wr							***0***
Overhead							
O1							
O2							
O3							
O4							
Total	**8**	**5**	**5**	**4**	**7**	**6**	***35***
	Comparison of Reuse Cost Estimation Model by Output						
E	**Y**	**Y**		**Y**			***3***
C			**Y**	**Y**		**Y**	***3***
NPV					**Y**		***1***
SZ	**Y**						***1***

VII. Analysis of Similarities

Thirty-five cost factors were considered in this study. Table-2 reveals the following commonly used cost factors:

- Cost to reuse: cost to identify, retrieve, understand, validate, integrate, and test an asset (4 models)
- Cost to producer to create asset for reuse (3 models)
- Reused lines of code in product (2 models)
- Cost to adapt asset (2 models)
- Cost to create product/system without reuse (2 models)
- Cost to create product/system with reuse (2 models)
- Lines of code in product (2 models)
- Cost to consumer to reuse asset (2 models)

The most common cost factors make up fifteen of the total of thirty-five instances resulting from the comparison or 41% similarity, and they make up eight of the thirty-three or 23% of the cost factors used for the comparison. The greatest similarity among the models involves the costs to create or adapt components for reuse. As far as costs to the consumer, Jasmine/Vasantha, Tomer, and Boehm models all include the cost to adapt an asset and the cost to acquire reusable assets in their calculations.

Both Boehm and SPC include lines of code in the product (PLOC) as well as reused lines of code in the product (RLOC) in their calculations. For Boehm, PLOC is similar to his equivalent source lines of code, and for SPC, RLOC and PLOC are used to determine the proportion of reused code in the product. All of the identified costs in the list of similarities specified above occur in the design, development, and integration/test phases. When reuse is implemented, requirements, code, and test cases are affected. Documentation of the reused artifact as well as the developed/modified artifact becomes very important in order for the reused components to be implemented more easily and accurately. The overhead costs of reuse are not included in the formulas of any models; however, Jasmine/Vasantha's approach considers additional overhead reuse costs, specifically domain analysis, creation and operation of a reuse library, and training of personnel in design for and with reuse. None of the six models explicitly considers the time required to create or revise a particular component. Such measurements include the start and end times of an assignment or project duration.

VIII. Analysis of differences

Not one of these models indicates whether the producer or consumer should be responsible for the cost of a reusable asset. The producer and consumer information found in this paper comes from Lim's 1996 paper [19] on Reuse Economics. Both Jasmine/Vasantha and Lim include the cost to create a product/system without reuse in their calculations of the cost with reuse. Lim and Tomer's calculations of the cost of development with reuse include the cost to the consumer to create a product with reuse. Only Jasmine/Vasantha considers the cost to the consumer to operate and maintain a product/system created for reuse, but they do not include this information in their calculations. They simply suggest that this information be considered when determining the cost of reuse. Only SPC and Tomer include the cost to the consumer to reuse an asset in their calculations. SPC uses the relative cost to reuse software, and Tomer uses the cost of black box reuse. Only Tomer's Controlled Reuse Cost formula uses the cost to the consumer of external acquisition, searching, and retrieval. The importance of documentation to the reuse effort is only mentioned in one model (Jasmine/Vasantha) - increased documentation facilitates reuse, and maintenance and enhancement of reuse documents is suggested; However, even in this model, the cost of documentation is not specified in any formula.

References

[1]. Henrik, a software development professor in Sweden.

[2]. Jared Fortune1, Ricardo Valerdi2, Barry W.Boehm3 and F. Stan Settles4 .Estimating Systems Engineering Reuse, 7th Annual Conference on Systems Engineering Research 2009 (CSER 2009).

[3]. Barnes, B. and T. Bollinger, Making Reuse Cost Effective, IEEE Software, Jan. 1991, 8(l): 13-24.

[4]. Jasmine K.S and Dr. R. Vasantha ,Cost Estimation Model For Reuse Based Software Products, Proceedings of the International Multi Conference of Engineers and Computer Scientists 2008 Vol I IMECS 2008, 19-21 March, 2008, Hong Kong

[5] Jared Fortune and Ricardo Valerdi Los Angeles, Considerations for Successful Reuse in Systems Engineering, University of Southern California , CA, Massachusetts Institute of Technology, Cambridge, MA.

[6]. Hareton Leung Zhang Fan. Software Cost Estimation Department of Computing .The Hong Kong Polytechnic University.

[7]. B. W. Boehm, *Software engineering economics*, Englewood Cliffs, NJ: Prentice-Hall, 1981.

[8]. P. Strassman (1997), The Squandered Computer: Evaluating the Business Alignment of Information Technologies, Information Economics Press, New Canaan, CT.

[9]. N. A. Parr, "An alternative to the Raleigh Curve Model for Software development effort", IEEE on Software Eng. May 1980.

[10]. L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem", IEEE Trans. Soft. Eng., July 1978, pp. 345-361.
[11]. W. Royce, *Software project management: a unified framework*, Addison Wesley, 1998.
[12]. Hisham M. Haddad, Nancy R. Ross, and Woranuch Kaensaksiri. Software Reuse Cost Factors .Department of Computer Science, Kennesaw State University, Kennesaw, GA, USA.
[13]. Tomer, A., Goldin, L., Kuflik, T., Kimchi, E., and Schach, S. R. (2004). Evaluating software reuse alternatives: A model and its application to an industrial case study. *IEEE Transactions on Software Engineering*, *30*(9), 601-61
[14]. Ravichandran, T., and Rothenberger, M. A. (2003). Software reuse strategies and component markets. *Communications of the ACM*. *46*(8), 109-114.
[15]. Krutz, W. K., Allen, K., and Olivier, D. P. (1991). The costs related to making software reusable: Experience from a real project. *Proceedings of the Conference on TRI-Ada' 91: Today's Accomplishments, Tomorrow's Expectations*, 437-443.
[16]. Boehm, B., Abts, C., and Chulani, S. (2000). Software development cost estimation approaches - A survey. *Annals of Software Engineering*, *10*(1-4). Springer, Netherlands, November 2000.
[17]. Nasir, M. (2006). A survey of software estimation techniques and project planning practices. Proceedings of the ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing.
[18]. Hisham M. Haddad, Nancy R. Ross, and Woranuch Kaensaksiri. Software Reuse Cost Factors. Department of Computer Science, Kennesaw State University, Kennesaw, GA, USA.
[19]. Lim, W. C. (1996). Reuse economics: A comparison of seventeen models and directions for future research. *Proceedings of the Fourth International Conference on Software Reuse 1996*, 41-50.
[20]. Chmiel, S. F. (2000). An integrated cost model for software reuse. (Doctoral Dissertation). Retrieved from the ACM Digital Library database.
[21]. Kemerer, C. F. (1987). An empirical validation of software cost estimation models. *Communication of the ACM, 30*(5), 416-429.
[22]. Hari, CH., Reddy, P., Kumar, S., and Ganesh, G. (2009). Identifying the importance of software reuse in COCOMO81, COCOMOII. *International Journal on Computer Science and Engineering, 1*(3), 142-147.
[23]. Data & Analysis Center for Software (DACS). (2010). Assess reuse risks and costs. *Software Tech 13*(3). Retrieved on April 29, 2012 from the World Wide Web: http://www.goldpractices.com/practices/arrc/index.php .
[24]. Software Engineering Institute (SEI) (2009). A framework for software product line practice, version 5.0: What is a software product line? Retrieved April 29, 2012, Wide Web: http://www.sei.cmu.edu/productlines/frame_report/what.is.a.PL.htm.

