International Journal of Emerging Science and Engineering (IJESE)
ISSN: 2319–6378, Volume-1 Issue-6 April 2013

Comparison of SDLC-2013 Model with Other SDLC Models by using COCOMO

Naresh Kumar, Pinky Chandwal

***Abstract*— There exist a large number of Models to develop software. Each model has its own characteristics, limitations and working environment. According to the requirements, software industry people use different models to develop different software. Waterfall model is generally used for development of software that is small with clear and stable requirements. While prototype model is used for the development of that software whose requirements are unclear and unstable, Incremental model is similar to the waterfall model but the software is developed in increments. Due to different architecture of SDLC models, each of them leads to different LOC provided that the same software is being developed. Simply we can put this discussion as different SDLC if used for developing same software then the amount of LOC that would be coded will be different. In this study we compare software build by different SDLC models in terms of cost schedule and effort estimated by using COCOMO.**

***Index Terms*— SDLC, Software Development, SDLC Phases, LOC, COCOMO Model.**

I. INTRODUCTION

Software development life cycle is the most important element in software development. Software Development Life Cycle (SDLC) is a process of building software [1]. Typically, it includes various phases and every SDLC models describe these phases and the order in which they are to be executed so as to develop software. The common phases are Requirement Analysis, Designing, Coding, Testing and Maintenance, etc. A software application is designed to perform a particular set of tasks [2]. Often, this set of tasks that the system will perform provides well-defined results, which involve complex computation and processing. Thus, a systematic development process which is able to emphasize on the understanding of the scope and complexity of the total development process is essential [3]. Now-a-days a large number of life cycle models are available for the systematic development of software such as waterfall model, prototyping model, incremental model and spiral model etc. These models have their own unique characteristics and are suited to a particular situation of software development and software types [4]. Choosing the right SDLC is very important because choosing the wrong SDLC will add time to the development cycle. Adding extra time to the development cycle will automatically increase estimated budget and effort required to build the software [5]. One software life cycle model may prove to be more efficient than the other one depending upon the development environment. Due to different architecture of SDLC models, each of them leads to different LOC provided that the same software is being developed, that is, different SDLC if used for developing same software then the amount of LOC that would be coded will be different [6].

Manuscript received on April, 2013.

Naresh Kumar, M. Phil (IT) Scholar, Information Technology Department, Dr. C. V. Raman university, Bilaspur, India.
Pinky Chandwal, M. Phil (IT) Scholar, Information Technology Department, Dr. C. V. Raman university, Bilaspur, India.

II. METHODOLOGY

COCOMO

COCOMO was first published in 1981 Barry W. Boehm as a model for estimating effort, cost, and schedule for software projects. COCOMO is a hierarchy of software cost estimation model, which include basic, intermediate and detailed sub models [7].

COCOMO Models are:

Basic	The Basic COCOMO model which computes software development effort and cost as a function of program size expressed in LOC.
Intermediate	The Intermediate COCOMO model which computes software development effort and cost as a function of program size and a set of cost drivers that include subjective assessments of product, hardware, personnel, and Project attributes.
Advance	The Advanced COCOMO model which incorporates all the characteristics of the intermediate version with an assessment of all the cost drivers' impact on each step (analysis, design, etc.) of the software engineering Process.

BASIC COCOMO

Basic COCOMO is a static, single-valued model that computes software development effort and cost as a function of program size expressed in estimated lines of code. COCOMO applies to three classes of software projects:

1. Organic projects - are relatively small, simple software projects in which small teams with good application experience work to a set of less than rigid requirements.
2. Semi-detached projects - are intermediate (in size and complexity) software projects in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements.
3. Embedded projects - are software projects that must be developed within a set of tight hardware, software, and operational constraints.

The basic COCOMO equations take the form

$$E = a_1 (KLOC)^{b_1}$$

$$D = c_1 (E)^{d_1}$$

$$P = E / D$$

Where E is the effort applied in person-months, D is the development time in chronological Months, KLOC is the estimated number of delivered lines of code for the project (expressed in thousands), and P is the number of people required. The coefficients a_1 b_1 c_1 d_1 are given in the following table [8].

Software Projects	a_1	b_1	c_1	d_1
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and other project attributes known to have a significant influence on software costs, which limits its accuracy.

INTERMEDIATE COCOMO

Intermediate COCOMO computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four "cost drivers", each with a number of subsidiary attributes [9].

1) Product attributes
   a) Required software reliability
   b) Size of application database
   c) Complexity of the product
2) Hardware attributes
   a) Run-time performance constraints
   b) Memory constraints
   c) Volatility of the virtual machine environment
   d) Required turnabout time
3) Personnel attributes
   a) Analyst capability
   b) Software engineering capability
   c) Applications experience
   d) Virtual machine experience
   e) Programming language experience
4) Project attributes
   a) Use of software tools
   b) Application of software engineering methods
   c) Required development schedule

Each of the 15 attributes receives a rating on a six-point scale that ranges from "very low" to "extra high" (in importance or value). An effort multiplier from the table below applies to the Rating. The product of all effort multipliers results in an EFFORT ADJUSTMENT FACTOR (EAF). Typical values for EAF range from 0.9 to 1.4.

The Intermediate COCOMO formula now takes the form:

$$E = a_1 (KLOC)^{b_1} . EAF$$

Where E is the effort applied in person-months, **KLOC** is the estimated number of thousands of delivered lines of code for the project, and **EAF** is the factor calculated above. The coefficient a_2 and the exponent b_2 are given in the next table.

Software Projects	a_2	b_2
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

The Development time D calculation uses E in the same way as in the Basic COCOMO.

ADVANCED COCOMO

Advanced COCOMO - incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process [10].

III. DEPLOYING SOFTWARE

Software is developed for automating the work of a doctor's clinic. There are various Software Development Models for developing software but we choose Waterfall model, Prototype Model, Incremental Model and SDLC-2013 Model for developing software (DCA, stand for doctor's clinic automation) in order to compare the working of existing Models with the SDLC-2013. Software developed by traditional SDLC Models:

A. Development of software by Waterfall model

As we know waterfall is a linear sequential flow model. We analysed the requirements and freeze them and moved toward the designing phase followed by the Coding and testing phases for developing software named as DCA-I. But the DCA-1 was not accepted by the client (doctor) because client was not satisfied, as the client want to change it in terms of graphics, functionality and features. As, waterfall model does not allow changes after freezing the requirements so, it fails to deliver the software product.

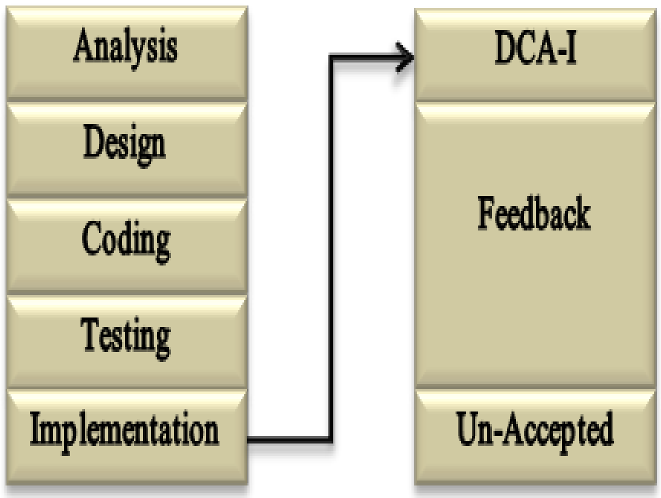

Figure1. Employing waterfall model for software development

B. Development of software by Prototype model

We know that prototype model build prototype to give feel of the proposed software to the client. As we already have doctor's requirement so, we build prototype and showed it to the client.

After client's feedback, we changed it and again showed it to the client. After building and showing three prototypes, doctor finalized the requirements and we passed these final requirements to next phases to develop the software and named it as DCA-II. Finally DCA-II was delivered to the client. But building prototype affects cost, schedule and effort which get exceeded.

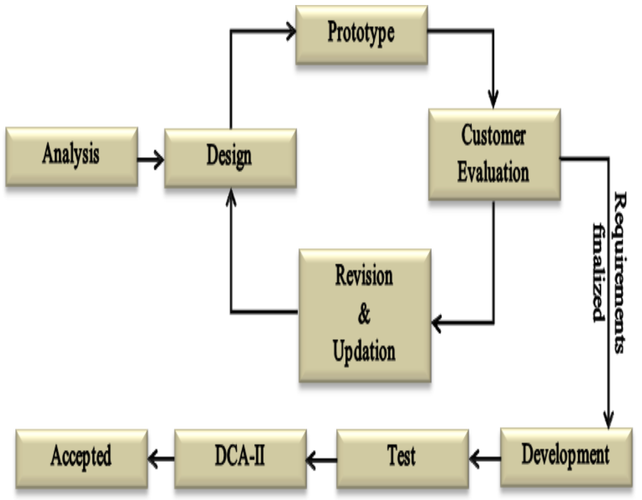

Figure2. Employing prototype model for software development

C. Development of software by Incremental model

Incremental model is an evolution of waterfall model which has number of iterations and after each iteration, we get a working product. Initially we analysed the requirements and go through the designing, coding and testing phases and released the first iteration. The first iterations working product was given to the client and after getting clients feedback we changed it and released the product of the second iteration. With each iteration functionality and feature of the product get enhanced and after three iterations we got DCA-III which was finally delivered to the client. Incremental model reduce the cost of building prototype because instead of building prototype it accommodate the changes into the working product but due to iterations, schedule get exceeded which in turn effect the cost and effort.

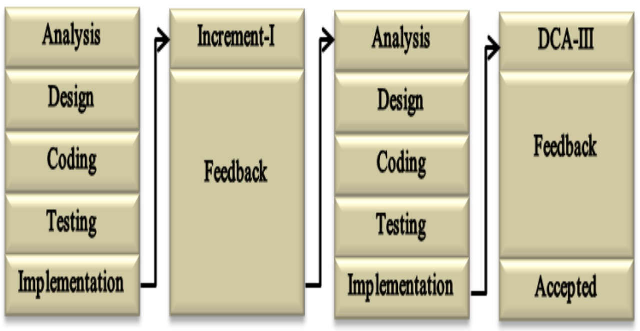

Figure3. Employing incremental model for software development

D. Development of software by SDLC-2013 model

SDLC-2013 is an Advance Model for the software development. The striking feature of this model is the client satisfaction.

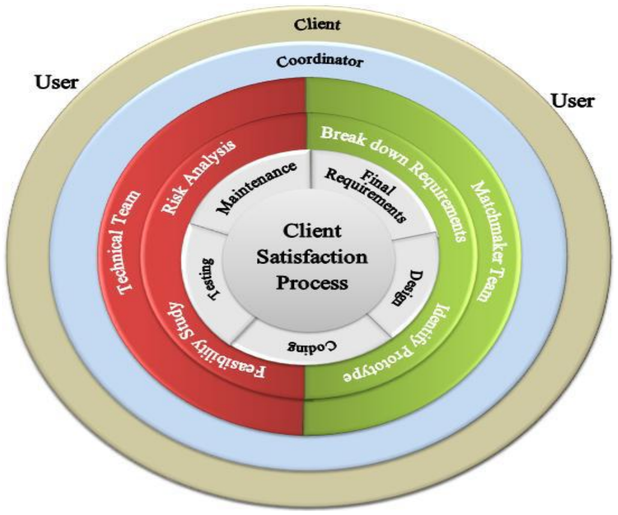

Figure4. SDLC-2013 Model

Firstly, Coordinator deal with the client (doctor) to discover the requirements and then he passed these requirements to the matchmaker team. Matchmaker team analysed the available requirements for the proposed system and searched the most matching software for them. He found two such software whose requirements matched with the proposed software's requirements. Accordingly, he has to breakdown the available requirements into implemented and non-implemented requirements but in this case there was no non implemented requirement. Implemented requirements along with their matching software were given back to the coordinator. Coordinator showed the software to the client so that the client got the feel of proposed software and also identifies the undiscovered requirements and gave his suggestion and feedback to the coordinator. Coordinator again passed these suggestions to the matchmaker team and the process goes on until the client finalized the requirements. Coordinator passed final requirements to the technical team for the risk analysis and requirement validation. After validation and resolving various risk associated with the final requirements, these requirements were passed to designing, coding and testing phases followed by the validation process to develop the final product named as DCA. DCA was accepted by the client because it satisfied the client's requirements within budget and schedule because budget and schedule were not disturbed or affected due to various increments or by building prototypes.

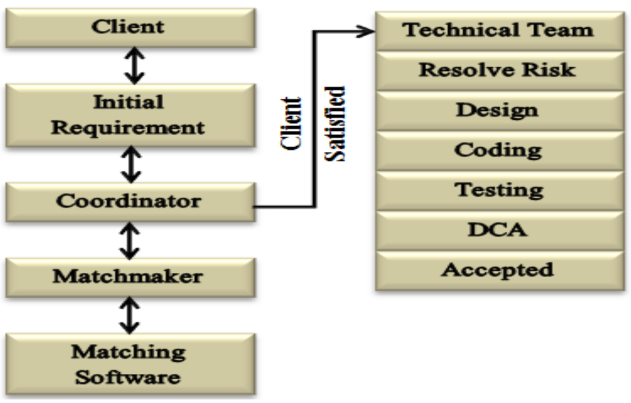

Figure5. Employing SDLC-2013 model for software development

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Module	Forms	Software	Total Design LOC	Total App. LOC	Module Design LOC	Module App. LOC	Total Module LOC	Total Test	Successful Test	Failed test
Security	Login				123	32	155	12	8	4
Entry	Welcome				97	19	116	3	2	1
Data manage	Patient Registration Patient Record Case Record Contact list Doctor Profile	DCA-I	1205	418	642	280	922	57	45	12
Report	Report				343	87	430	15	13	2
					1205	418		87	68	19
Total LOC			1623		1623		1623			

Table1. Conclusive Result of Waterfall Model product DCA-I

Module	Forms	Software	Total Design LOC		Total App. LOC		Module Design LOC		Module App. LOC		Total Module LOC	Total Test	Successful Test	Failed test
			Re-use	New	Re-use	New	Re-use	New	Re-use	New				
Security	Login						40	90	24	36	190	9	8	1
Entry	Welcome						18	78	11	24	131	7	5	2
Data manage	Patient Registration Patient Record Case Record Contact list Doctor Profile	DCA-II	703	912	328	810	465	491	226	655	1837	51	47	4
Report	Report						180	253	67	95	595	19	15	4
			1615		1138		703	912	328	810		86	75	11
							1615		1138					
Total LOC			2753				2753				2753			

Table2. Conclusive Result of Prototype Model product DCA-II

Module	Forms	Software	Total Design LOC		Total App. LOC		Module Design LOC		Module App. LOC		Total Module LOC	Total Test	Successful Test	Failed test
			Re-use	New	Re-use	New	Re-use	New	Re-use	New				
Security	Login						36	83	20	36	175	14	12	2
Entry	Welcome						14	70	9	24	117	11	9	2
Data manage	Patient Registration Patient Record Case Record	DCA-III	570	710	270	570	410	372	174	415	1371	35	34	1

	Contact list												
	Doctor Profile												
Report	Report						110	185	67	95	457	23	21	2
			1280		840		570	710	270	570		83	76	7
							1280		840					
Total LOC			2120				2120				2120			

Table3. Conclusive Result of Incremental Model product DCA-III

Module	Forms	Software	Total Design LOC		Total App. LOC		Module Design LOC		Module App. LOC		Total Module LOC	Total Test	Successful Test	Failed test
			Re-use	New	Re-use	New	Re-use	New	Re-use	New				
Security	**Login**						**76**	**46**	**45**	**30**	**197**	**11**	**11**	**0**
Entry	**Welcome**						**65**	**45**	**34**	**24**	**168**	**9**	**9**	**0**
	Patient Registration													
	Patient Record													
Data manage	**Case Record**	**DCA**	**515**	**412**	**330**	**236**	**199**	**176**	**164**	**116**	**655**	**20**	**19**	**1**
	Contact list													
	Doctor Profile													
Report	**Report**						**175**	**145**	**87**	**66**	**473**	**23**	**23**	**0**
			927		**566**		**515**	**412**	**330**	**236**		**63**	**62**	**1**
							927		**566**					
Total LOC			**1493**				**1493**				**1493**			

Table4. Conclusive Result of SDLC-2013 Model product DCA

Module	Forms	Total Test				Successful Test				Failed Test			
		DCA-I	DCA-II	DCA-III	DCA	DCA-I	DCA-II	DCA-III	DCA	DCA-I	DCA-II	DCA-III	DCA
Security	**Login**	**12**	**9**	**14**	**11**	**8**	**8**	**12**	**11**	**4**	**1**	**2**	**0**
Entry	**Welcome**	**3**	**7**	**11**	**9**	**2**	**5**	**9**	**9**	**1**	**2**	**2**	**0**
	Patient Registration												
	Patient Record												
Data manage	**Case Record**	**57**	**51**	**35**	**20**	**45**	**47**	**34**	**19**	**12**	**4**	**1**	**1**
	Contact list												
	Doctor Profile												

Report	**Report**	**15**	**19**	**23**	**23**	**13**	**15**	**21**	**23**	**2**	**4**	**2**	**0**
Total		**87**	**86**	**83**	**45**	**68**	**71**	**76**	**62**	**19**	**11**	**7**	**1**

Table5. Conclusive Result of Test Cases

A. Overall Cost and Schedule Estimation for waterfall Model product DCA-I

Here, we will take the project to be organics.

Total lines of codes = 0.155k + 0.116k + 0.922k + 0.430 = <u>1.623k.</u>

$E = 2.4\,(1.623)^{1.05}$ = <u>**3.99**</u> **Persons-Months**

$D = 2.5\,(3.99)^{0.38}$ = <u>**4.22**</u> **Months**

$P = 3.99 \,/\, 4.2297$ = <u>**0.94**</u> <u>**= 1**</u> **Persons**

Assume salary = 12000, then cost of project = E * 12000 = 3.99 * 12000 = 47880

B. Cost Estimation for Different Modules of DCA-I

Security Module = 0.115k	**Entry Module = 0.116k**	**Data Module = 0.922k**	**Report Module = 0.430k**
$E = 2.4\,(0.155)^{1.05}$ = <u>**0.33**</u> **Persons-Months**	$E = 2.4\,(1.16)^{1.05}$ = <u>**2.80**</u> **Persons-Months**	$E = 2.4\,(0.922)^{1.05}$ = <u>**2.20**</u> **Persons-Months**	$E = 2.4\,(0.430)^{1.05}$ = **0.<u>98</u>** **Persons-Months**
$D = 2.5\,(0.33)^{0.38}$ = <u>**1.64**</u> **Months**	$D = 2.5\,(2.80)^{0.38}$ = <u>**3.69**</u> **Months**	$D = 2.5\,(2.20)^{0.38}$ = <u>**3.37**</u> **Months**	$D = 2.5\,(0.98)^{0.38}$ = <u>**2.48**</u> **Months**
$P = 0.33 \,/\, 1.64$ = <u>**0.20**</u> <u>**= 1**</u> **Persons**	$P = 2.80 \,/\, 3.69$ = <u>**0.75**</u> <u>**= 1**</u> **Persons**	$P = 2.20 \,/\, 3.37$ = <u>**0.65**</u> <u>**= 1**</u> **Persons**	$P = 0.98 \,/\, 2.48$ = <u>**0.39**</u> <u>**= 1**</u> **Persons**

A. Overall Cost and Schedule Estimation for Prototype Model product DCA-II
Here, we will take the project to be organics.
Total lines of codes = 0.19k + 0.131k + 1.837k + 0.595 = <u>2.753k.</u>

$E = 2.4\,(2.753)^{1.05}$ = <u>**6.95**</u> **Persons-Months**

$D = 2.5(6.95)^{0.38}$ = <u>**5.22**</u> **Months**

$P = 6.95 \,/\, 5.22$ = <u>**1.33**</u> <u>**= 2**</u> **Persons**

Assume salary = 12000, then cost of project = E * 12000 = 6.95 * 12000 = 83400

B. Cost Estimation for Different Modules of DCA-II

Security Module = 0.19k	**Entry Module = 0.131k**	**Data Module = 1.837k**	**Report Module = 0.595k**
$E = 2.4\,(0.19)^{1.05}$ = **0.41** **Persons-Months**	$E = 2.4\,(0.131)^{1.05}$ = **0.28** **Persons-Months**	$E = 2.4\,(1.837)^{1.05}$ = **4.54** **Persons-Months**	$E = 2.4\,(0.595)^{1.05}$ = **1.39** **Persons-Months**
$D = 2.5\,(0.41)^{0.38}$ = **1.78** **Months**	$D = 2.5\,(0.28)^{0.38}$ = **1.54** **Months**	$D = 2.5\,(4.54)^{0.38}$ = **4.44** **Months**	$D = 2.5\,(1.39)^{0.38}$ = **2.83** **Months**
$P = 0.41 / 1.78$ = **0.23** = **1** **Persons**	$P = 0.28 / 1.54$ = **0.18** = **1** **Persons**	$P = 4.54 / 4.44$ = **1.02** = **2** **Persons**	$P = 1.39 / 2.83$ = **0.49** = **1** **Persons**

A. Overall Cost and Schedule Estimation for Incremental Model product DCA-III

Here, we will take the project to be organics.
Total lines of codes = 0.175k + 0.117k + 1.371k + 0.457 = 2.12k.

$E = 2.4\,(2.12)^{1.05}$ = **5.28** **Persons-Months**

$D = 2.5(5.28)^{0.38}$ = **4.70** **Months**

$P = 5.28 / 4.70$ = **1.12** = **2** **Persons**

Assume salary = 12000, then cost of project = E * 12000 = 5.28 * 12000 = 63360

B. Cost Estimation for Different Modules of DCA-III

Security Module = 0.175k	**Entry Module = 0.117k**	**Data Module = 1.371k**	**Report Module = 0.457k**
$E = 2.4\,(0.175)^{1.05}$ = **0.38** **Persons-Months**	$E = 2.4\,(1.17)^{1.05}$ = **2.83** **Persons-Months**	$E = 2.4\,(1.371)^{1.05}$ = **3.34** **Persons-Months**	$E = 2.4\,(0.457)^{1.05}$ = **1.05** **Persons-Months**
$D = 2.5\,(0.38)^{0.38}$ = **1.73** **Months**	$D = 2.5\,(2.83)^{0.38}$ = **3.71** **Months**	$D = 2.5\,(3.34)^{0.38}$ = **3.95** **Months**	$D = 2.5\,(1.05)^{0.38}$ = **2.54** **Months**

$P = 0.38 / 1.73$ = <u>0.21</u> <u>= 1</u> **Persons**	$P = 2.83 / 3.71$ = <u>0.76</u> <u>= 1</u> **Persons**	$P = 3.34 / 3.95$ = <u>0.84</u> <u>= 1</u> **Persons**	$P = 1.05 / 2.54$ = <u>0.41</u> <u>= 1</u> **Persons**

A. Overall Cost Schedule Estimation for SDLC-2013 product DCA

Here, we will take the project to be organics.

Total lines of codes = 0.197k + 0.168k + 0.655k + 0.473 = <u>1.493k.</u>

$E = 2.4\,(1.493)^{1.05}$ = <u>3.65</u> **Persons-Months**

$D = 2.5(3.65)^{0.38}$ = <u>4.08</u> **Months**

$P = 3.65 / 4.08$ = <u>0.89</u> <u>= 1</u> **Persons**

Assume salary = 12000, then cost of project = E * 12000 = 3.65 * 12000 = 43800

B. Cost Estimation for Different Modules of DCA

Security Module = 0.197k	**Entry Module = 0.168k**	**Data Module = 0.655k**	**Report Module = 0.473k**
$E = 2.4\,(0.197)^{1.05}$ = <u>0.43</u> **Persons-Months**	$E = 2.4\,(0.168)^{1.05}$ = <u>0.36</u> **Persons-Months**	$E = 2.4\,(0.655)^{1.05}$ = <u>1.53</u> **Persons-Months**	$E = 2.4\,(0.473)^{1.05}$ = <u>1.09</u> **Persons-Months**
$D = 2.5\,(0.43)^{0.38}$ = <u>1.81</u> **Months**	$D = 2.5\,(0.36)^{0.38}$ = <u>1.69</u> **Months**	$D = 2.5\,(1.53)^{0.38}$ = <u>2.93</u> **Months**	$D = 2.5\,(1.09)^{0.38}$ = <u>2.58</u> **Months**
$P = 0.43 / 1.81$ = <u>0.23</u> <u>= 1</u> **Persons**	$P = 0.36 / 1.69$ = <u>0.21</u> <u>= 1</u> **Persons**	$P = 1.53 / 2.93$ = <u>0.52</u> <u>= 1</u> **Persons**	$P = 1.09 / 2.58$ = 0.<u>42</u> <u>= 1</u> **Persons**

Table16. Overall Comparison Result of Different SDLC Models

	DCA-I	**DCA-II**	**DCA-III**	**DCA**
Persons-Months	**3.99**	**6.95**	**5.28**	**3.65**
Months	**4.22**	**5.22**	**4.70**	**4.08**
Persons	**1**	**2**	**2**	**1**

Cost	**47880**	**83400**	**63360**	**43800**

IV. CONCLUSION

The proposed work is summarized as the development of software by different SDLC models and compares them in terms of cost, schedule and effort. For this comparison, we choose COCOMO model which is well known and widely accepted for the early estimation of cost, schedule and effort. After following COCOMO equations and by performing various calculations of COCOMO model, the comparison result is shown in tabular form which is easy to understand and analyses. According to the comparison result, it is clear that in this case SDLC-2013 is more efficient than other Models such as Waterfall Model, Prototype Model and Incremental Model in terms Cost, Schedule and Effort.

REFERENCES

1. Naresh Kumar , A. S. Zadgaonkar, Abhinav Shukla - Estimation of software Quality by Using fuzzy (FIS) : volume 2, issue-1 IJSCE.
2. K. K. Aggarwal, Yogesh Singh Software Engineering 3rd Edition.
3. Software Development Life Cycle (SDLC) – the five common principles.htm
4. Software Methodologies Advantages & disadvantages of various SDLC models.mht
5. Comparative analysis of different types of models in Software Development Life Cycle, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Volume 2, May 2012, Ms. Shikhamaheshwari, Prof. Dinesh Ch. Jain
6. Comparing various SDLC models and the new proposed model on the basis of available methodology, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), volume 2, April 2012,Vishwas Massey, Prof. K. J Satao.
7. Roger Pressman titled Software Engineering - a practitioner's approach
8. Seminar on Software Cost Estimation by Requirements Engineering Research Group, Department of Computer Science, University of Zurich, Switzerland. Prof. Dr. Martin Glinz, Arun Mukhija.
9. www.en.wikipedia.org/wiki/COCOMOMODEL
10. www.en.wikipedia.org/wiki/COCOMO Different Models

AUTHORS PROFILE

Naresh Kumar did his MSC-IT from Baba Ghulam Shah Badshah University, Rajouri, J&K., India and Currently pursing M.Phil-IT from Dr. C. V. Raman University, Bilaspur, Chhattisgarh, India He has Published 3 Papers in International Journals.

Pinky Chandwal did her MSC-IT from Baba Ghulam Shah Badshah University, Rajouri, J&K., India and Currently pursing M.Phil-IT from Dr. C. V. Raman University, Bilaspur, Chhattisgarh, India. She has Published 3 Papers in International Journals.