Corad Agile Method for Agile Software Cost Estimation

Govind Singh Rajput, Ratnesh Litoriya

Department of Computer Science & Engineering, Jaypee University of Engineering & Technology, Guna, India
Email: Govind.singh.nsn@gmail.com, Ratnesh.litoriar@juet.ac.in

Received 16 April 2014; revised 19 May 2014; accepted 22 June 2014

Abstract

In this paper we are introducing new method of software cost estimation for Agile based software projects (Web based agile projects) and also like to introduce how this method is improving performance and efficiency of Agile based Software cost estimation. Rapid application development model CORADMO provided cost estimation in the form of rapid person productivity count by using person-month (Effort) and month (Schedule), but CORADMO followed waterfall model, so in agile case we do not require waterfall model fully and in agile software development all the stage of development having rapidly change with customer requirements, and COCOMO-II have no longer support for this, we needed a new method to improve rapid cost estimation performances, and also relate with agile methodology and provide more matching relation with agile methods: DSDM, ASD, AUP and SCRUM.

Keywords

Agile Methods (DSDM, ASD, SCRUM, AUP), COCOMO-II Extension CORADMO Method, COCOMO-II Scale Factors

Subject Areas: **Applications of Communication Systems, Network Modeling and Simulation**

1. Introduction

In present scenario most of the software projects are dealing with Agile Principles and using agile methodology in any state in the project for fast and efficient project outcomes. The emergence of agile methods has been phenomenal during the past few years and is not showing any signs of ceasing. However, managing these agile projects has presented difficulties for many project managers who have been trained in the use of traditional development approaches. However, we have found that many projects experience slower schedules by jumping into agile methods without awareness of their pitfalls [1]. These include making easiest first, hard-to-refractor architectural commitments, choosing unsalable or incompatible off-the shelf products, accepting unsuitable on-site customer representatives, teambuilding insufficiently, or assuming low personnel turnover. The Constructive

Rapid Application Development Model (CORADMO) attempts to quantify both the positive and the negative effects of key schedule drivers, and thus enable planners to estimate the relative schedule that will result from varying these parameters. CORADMO is a derivative of the revised Constructive Cost Model (COCOMO II) [2], this was calibrated against larger projects that were typically optimized to reduce cost. In contrast, the goal of projects using agile/lean techniques is often to compress schedule. But in this sequence still we do not have any desire cost estimation technique for Rapid application development for agile methods. For dealing we construct new method for agile name is Corad-Agile (Constructive rapid application development method for agile), for this concept we are using agile methods (SCRUM, DSDM, ASD and AUP) and using scale factors of COCOMO-II with COCOMO-II extension CORADMO cost drivers.

Scrum: Scrum is an iterative and incremental agile software development framework for managing software projects and product or application development. Its focus is on “a flexible, holistic product development strategy where a development team works as a unit to reach a common goal” as opposed to a “traditional, sequential approach”. Scrum enables the formation of self-organizing teams by heartening co-location of all team members, and verbal communication between all team members and disciplines in the project. A key principle of Scrum is its gratitude that during a project the customers can change their minds about what they want and need (often called requirements churn), and that surprising challenges cannot be easily addressed in a traditional predictive or planned manner.

As such, Scrum adopts an empirical approach accepting that the difficulty cannot be fully understood or defined, focusing instead on maximizing the team’s ability to deliver quickly and respond to emerging requirements. Like other agile development methodologies, Scrum can be implemented through a wide range of tools. Many companies use universal tools, such as spreadsheets to build and maintain artifacts such as the sprint backlog [3].

Dynamic Systems development Method (DSDM): DSDM is an agile project delivery framework, primarily used as a software development method. First released in 1994, DSDM originally sought to provide some discipline to the rapid application development (RAD) method.

In 2007, DSDM became a generic approach to project management and solution delivery. DSDM is an incremental approach that embraces principles of agile development, including incessant user/customer participation. DSDM fixes cost, quality and time at the outset and uses.

Focus on the Business, Collaborate, Never negotiation quality, Build incrementally from firm foundations, Develop iteratively, Communicate continuously and clearly, Demonstrate control [3].

Agile Unified Process (AUP): AUP is a simplified version of the IBM Rational Unified Process (RUP) developed by Scott Ambler. It describes a simple, easy to understand approach to developing business application software using agile techniques and concepts yet still remaining true to the RUP. The AUP applies agile techniques including test driven development (TDD), Agile Modeling, agile change management, and database refactoring to improve productivity [3].

1) Disciplines: Unlike the RUP, the AUP has only seven disciplines.

2) Model: Understand the business of the organization, the problem domain being addressed by the project, and identify a viable solution to address the problem domain.

3) Implementation: Transform model(s) into executable code and perform a basic level of testing, in particular unit testing.

4) Test: Perform an objective evaluation to ensure quality. This includes finding defects, validating that the system works as designed, and verifying that the requirements are met.

5) Deployment: Plan for the delivery of the system and to execute the plan to make the system available to end users.

6) Configuration Management: Manage access to project artifacts. This includes not only tracking artifact versions over time but also controlling and managing changes to them.

7) Project Management: Direct the activities that take place within the project. This includes managing risks, directing people (conveying tasks, tracking progress, etc.), and coordinating with people and systems outside the scope of the project to be sure that it is delivered on time and within budget.

8) Environment: Support the rest of the effort by ensuring that the proper process, guidance (standards and guidelines), and tools (hardware, software, etc.) are available for the team as needed.

Adaptive Software Development (ASD): ASD is a software development process that grew out of rapid application development ASD replaces the traditional waterfall cycle with a repeating series of speculates, collaborate, and learn cycles. This dynamic cycle provides for continuous learning and adaptation to the emergent state of the project. The characteristics of an ASD life cycle are that it is mission focused, feature based, iterative, time boxed, risk driven, and change tolerant. The word speculates refers to the paradox of planning—it is more likely to assume that all stakeholders are comparably wrong for certain aspects of the project's mission, while trying to define it. Collaboration refers to the efforts for balancing the work based on predictable parts of the environment (planning and guiding them) and adapting to the uncertain surrounding mix of changes caused by various factors—technology, requirements, stakeholders, software vendors, etc. The learning cycles, challenging all stakeholders, are based on the short iterations with design, build and testing. During these iterations the knowledge is gathered by making small mistakes based on false assumptions and correcting those mistakes, thus leading to greater experience and eventually mastery in the problem domain [4].

1.1. Cost Drivers

A cost driver is the unit of an activity that causes the change inactivity's cost. Examples: In marketing, cost drivers are Number of advertisements, Number of sales personnel etc. In Customer service, cost drivers are Number of service calls attended, number of staff in service department, number of warranties handled, Hours spent on servicing etc.

The Activity Based Costing (ABC) approach relates indirect cost to the activities that drive them to be incurred. Activity Based Costing is based on the belief that activities cause costs and therefore a link should be established between activities and product. The cost drivers thus are the link between the activities and the cost.

In traditional costing the cost driver to allocate indirect cost to cost objects was volume of output. With the change in business structures, technology and thereby cost structures it was found that the volume of output was not the only cost driver.

RAD-Agile with New Cost Drivers

Our required model is to address mainly Rapid Application model for Agile Methods (DSDM, AUP, SCRUM, FDD, and ASD) for RAD-Agile method estimation here we replaced three cost drivers RAD capability of Personnel (RCAP), Collaboration Support (CLAB), and Prepositioning Assets (PPOS) with new cost drivers; 1) Degree of Collaboration Support (DCLAB), 2) Multi Site Development (MULSITE), 3) Daily Basis Customer Interaction with vendor team (DBCI).

These all multipliers support agile methods on target functionality and conveyance for agile methodology based software development here we have used three multipliers of existing CORADMO model these are also relate accurately and conventionally with agile methods and followed all functionalities of agile methods in different stages. They are such as 4) Rapid Prototyping (RPRO), 5) Development Process Reengineering and streamlining (DPRS), 6) Architecture, Risk Resolution (RESL).

1) Multi Site Development (MULSITE): Multi Site development provides different-different sites (location) team employees interaction for particular work module completion with help of this drivers more and more team interaction possible for particularly software work module completion within time line with great efficiency and effort, this is also define different location work creation with utilizing resources if resources are available to do things. So the multi site development provides good utilization of resources. Here **Table 1(a)** and **Table 1(b)** demonstrate multisite development cost driver values for all four phases with different levels (Nominal, High, Very High, Extra High) of cost driver adjustments.

2) Degree of Collaboration Support (DCLAB): Degree of Collaboration Support Provides smooth relationship between teams and team members who can collaborate effectively can reduce both effort and schedule, those that don't collaborate effectively have increased schedule and effort (due to wasted time).With this multiplier, staff level does not change based on collaboration support. But this cost driver provides better interaction in-between team members which is good for Multi Site development. Here **Table 2(a)** and **Table 2(b)** demonstrate Degree of Collaboration Support cost driver values for all four phases (Inception, Elaboration, Construction, and Transition) with different levels (Very Low, Low, Nominal, High, Very High, Extra High) of cost driver adjustments.

3) Daily Basis Customer Interaction with Vendor Teams (DBCI): In agile methodology customer interaction with vendor teams members like planning team feasibility team, designing, coding and testing team throughout customer interaction will be very much needed for completion project rapidly. Here daily basis customer

Table 1. Multisite development cost driver values.

(a)

MLSITE						
Level	Inception			Elaboration		
	PM	M	P	PM	M	P
N	0.91	0.91	1.00	1.00	1.00	1.00
H	0.93	0.93	1.11	1.03	0.93	1.11
VH	0.95	0.95	1.23	1.06	0.86	1.23
EH	0.97	0.97	1.38	1.10	0.80	1.38

(b)

MLSITE						
Level	Construction			Transition		
	PM	M	P	PM	M	P
N	1.00	1.00	1.00	0.99	0.99	1.00
H	1.03	0.93	1.11	0.93	0.93	1.00
VH	1.06	0.94	1.13	0.96	0.96	1.00
EH	1.10	0.97	1.13	0.97	0.97	1.00

Table 2. DCLAB cost driver values.

(a)

DCLAB						
Level	Inception			Elaboration		
	PM	M	P	PM	M	P
VL	1.21	1.21	1.00	1.15	1.15	1.00
L	1.10	1.10	1.00	1.07	1.07	1.00
N	1.00	1.00	1.00	1.00	1.00	1.00
H	0.93	0.93	1.00	0.95	0.95	1.00
VH	0.86	0.84	1.02	0.90	0.90	1.00
EH	0.83	0.80	1.04	0.86	0.86	1.00

(b)

DCLAB						
Level	Construction			Transition		
	PM	M	P	PM	M	P
VL	1.10	1.10	1.00	0.98	0.98	1.00
L	1.05	1.05	1.00	0.99	0.96	1.03
N	1.00	1.00	1.00	1.00	1.00	1.00
H	0.98	0.98	1.00	0.97	0.97	1.00
VH	0.95	0.95	1.00	0.96	0.95	1.01
EH	0.93	0.93	1.00	0.95	0.93	1.02

interaction with vendor teams will provide fast and iterative things to complete software projects within provided schedule and efforts. Here Table 3(a) and Table 3(b) demonstrate Daily basis customer interaction cost driver values for all four phases (Inception, Elaboration, Construction, Transition) with different levels (VL, L, N, H, VH, EH) of cost driver adjustments.

4) Rapid prototyping (RPRO): The first driver for CORADMO is RPRO (Rapid Prototyping) that expresses the degree to which the personnel have experience in Rapid Prototyping. This driver reflects schedule compression in Inception and Elaboration stages due to faster prototyping or option exploration. For this driver, the effort compression is hypothesized to be the same as the schedule compression; that is, the team size would stay the same over a shorter period. The rating for this driver depends on the amount of Rapid Prototyping Experience the development team has had in the domain of the project being evaluated. Here Table 4(a) and Table 4(b) demonstrate and provide Rapid Prototyping cost driver values for all four phases (Inception, Elaboration, Construction, Transition) with different levels (VL, L, N, H, VH, EH) of cost driver adjustments.

5) Development Process Reengineering and streamlining (DPRS): The degree to which the project and organization allow and encourage streamlined or re-engineered development processes: the current level of bureaucracy is a clear indicator. The schedule compression or expansion, because of this driver, doesn't alter staff level (P). Here Table 5(a) and Table 5(b) demonstrate and provide DPRS cost driver values for all four phases (Inception, Elaboration, Construction, Transition) with different levels (VL, L, N, H, VH, EH) of cost driver adjustments.

6) Architecture, Risk Resolution (RESL): This rating is exactly the same as the COCOMO II RESL rating. The architecture portion enables parallel construction, thus reducing schedule during the construction phase assuming that staff level increases during construction while applying the same effort. Good risk resolution in a schedule driven development effort applying RAD strategies increases the probability of the strategies' success.

Table 3. DBCI cost driver values.

(a)

DBCI						
Level	Inception			Elaboration		
	PM	M	P	PM	M	P
VL	1.04	1.04	1.00	1.02	1.02	1.00
L	1.00	1.00	1.00	1.00	1.00	1.00
N	0.98	0.98	1.00	0.99	0.99	1.00
H	0.97	0.94	1.03	0.97	0.95	1.02
VH	0.93	0.92	1.01	0.95	0.94	1.01
EH	0.90	0.90	1.00	0.95	0.95	1.00

(b)

DBCI						
Level	Construction			Transition		
	PM	M	P	PM	M	P
VL	1.20	1.20	1.00	0.89	0.89	1.00
L	1.03	1.03	1.00	0.88	0.88	1.00
N	1.00	1.00	1.00	1.00	1.00	1.00
H	0.98	0.95	1.03	0.87	0.87	1.00
VH	0.98	0.94	1.04	0.86	0.85	1.00
EH	0.97	0.96	1.01	0.85	0.85	1.00

Table 4. RPRO cost driver values.

(a)

Level	RPRO					
	Inception			Elaboration		
	PM	M	P	PM	M	P
VL	1.04	1.04	1.00	1.02	1.02	1.00
L	1.00	1.00	1.00	1.00	1.00	1.00
N	0.98	0.98	1.00	0.99	0.99	1.00
H	0.94	0.94	1.00	0.97	0.97	1.00
VH	0.90	0.90	1.00	0.95	0.95	1.00
EH	0.90	0.90	1.00	0.95	0.95	1.00

(b)

Level	RPRO					
	Construction			Transition		
	PM	M	P	PM	M	P
VL	1.00	1.00	1.00	0.99	0.99	1.00
L	1.00	1.00	1.00	0.88	0.88	1.00
N	1.00	1.00	1.00	1.00	1.00	1.00
H	1.00	1.00	1.00	0.87	0.87	1.00
VH	1.00	1.00	1.00	0.86	0.86	1.00
EH	1.00	1.00	1.00	0.85	0.85	1.00

Table 5. DPRS cost driver values.

(a)

Level	DPRS					
	Inception			Elaboration		
	PM	M	P	PM	M	P
VL	1.20	1.20	1.00	1.15	1.15	1.00
L	1.08	1.08	1.00	1.06	1.06	1.00
N	1.00	1.00	1.00	1.00	1.00	1.00
H	0.96	0.96	1.00	0.98	0.98	1.00
VH	0.90	0.90	1.00	0.95	0.95	1.00
EH	1.20	1.20	1.00	1.15	1.15	1.00

(b)

Level	DPRS					
	Construction			Transition		
	PM	M	P	PM	M	P
VL	1.15	1.15	1.00	1.02	1.02	1.00
L	1.06	1.06	1.00	0.87	0.87	1.00
N	1.00	1.00	1.00	0.92	0.92	1.00
H	0.98	0.98	1.00	0.85	0.85	1.00
VH	0.95	0.95	1.00	0.84	0.84	1.00
EH	1.15	1.15	1.00	1.02	1.02	1.00

Here **Table 6(a)** and **Table 6(b)** demonstrate Architecture, Risk Resolution cost driver values for all four phases (Inception, Elaboration, Construction, Transition) of software development life cycles with different levels (VL, L, N, H, VH, EH) of cost driver adjustments.

2.1. Scale Factors

A scale factor is a number which scales, or multiplies, some quantity. In the equation $y = Cx$, C is the scale factor for x. C is also the coefficient of x, and may be called the constant of proportionality of y to x. For example, doubling distances corresponds to a scale factor of 2 for distance, while cutting a cake in half results in pieces with a scale factor of $1/2$. The basic equation for it is image over pre image. In the field of measurements, the scale factor of an instrument is sometimes referred to as sensitivity. The ratio of any two corresponding lengths in two similar geometric figures is called as Scale Factor [5].

COCOMO-II Scale Factors

1) Precedentedss. Reflects the previous experience of the organization with this type of project. Very low means no previous experience. Extra high means that the organization is completely familiar with this application domain.

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

3) Architecture/risk resolution. Reflects the extent of risk analysis carried out. Very low means little analysis, Extra high means a complete a thorough risk analysis.

4) Team cohesion. Reflects how well the development teams know each other and work together. Very low

Table 6. RESL cost driver values.

(a)

	RESL					
Level	Inception			Elaboration		
	PM	M	P	PM	M	P
VL	1.00	1.00	1.00	1.00	1.00	1.00
L	1.00	1.00	1.00	1.00	1.00	1.00
N	1.00	1.00	1.00	1.00	1.00	1.00
H	0.97	0.97	1.00	1.00	1.00	1.00
VH	0.96	0.96	1.00	1.00	1.00	1.00
EH	0.95	0.95	1.00	1.00	1.00	1.00

(b)

	RESL					
Level	Construction			Transition		
	PM	M	P	PM	M	P
VL	1.00	1.00	1.00	0.98	0.98	1.00
L	1.00	1.00	1.00	0.95	0.95	1.00
N	1.00	1.00	1.00	1.00	1.00	1.00
H	1.00	0.91	1.10	0.94	0.94	1.00
VH	1.00	0.83	1.20	0.93	0.93	1.00
EH	1.00	0.75	1.33	0.92	0.92	1.00

means very difficult interactions, Extra high means an integrated and effective team with no communication problems.

5) Process maturity. Reflects the process maturity of the organization. The computation of this value depends on the CMM Maturity Questionnaire but an estimate can be achieved by subtracting the CMM process maturity level.

2.2. Agile Manifesto

1) Individuals and interactions over processes and tools.
2) Working software over comprehensive documentation.
3) Customer collaboration over contract negotiation.
4) Responding to change over following a plan.

2.3. Existing CORADMO Effort, Schedule

The Constructive Rapid Application Development model has its roots in the results of a 1997 CSE Focused Workshop on Rapid Application Development. RAD refers to an application of any of a number of techniques or strategies to reduce software development cycle time. The six classes of strategies whose degree of implementation can be used to parameterize a schedule estimate given an effort estimate produced by COCOMO II. 2000 are the following: Development Process Re-engineering (DPRS), Rapid Prototyping (RPRO), Collaboration efficiency (CLAB), Architecture and risk resolution (RESL), Pre-Positioning of assets (PPOS), RAD Capability of Personnel (RCAP) [6].

The intent of the Constructive Application Development Model is to calculate/predict the schedule (months, M), personnel (P), and adjusted effort (person-months, PM) based on the distribution of effort and schedule to the various stages, and impacts of the selected schedule driver ratings on the M, P, and PM of each stage.

The Constructive Application Development Model utilizes a new COCOMO II extension that allocates effort and schedule to the stages, which are anchored at points in a development life cycle. The anchor points are Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operational Capability (IOC).

A phased schedule and effort distribution is needed because the effects of the RAD strategies identified above are different for the different stages. Also, a new mathematical function is used to calculate (predict) the calendar months for a given amount of effort: the function is only radically different in low (under 16) person-month's efforts where it seems more normal have an equal number of people and months to accomplish the task. At the higher (greater than 120) person-month's efforts, the traditional COCOMO II. 2000 function is used which is based on the traditional cube-root-like function of effort. A smooth curve is fit within these ranges. CORADMO also allows the specification of the number of work hours per person-month [7] [8].

2.4. Proposed Model

Here we are using COCOMO-II baseline value calculation and put in to constructive phase schedule and effort model (COPSEMO), it is pre-processor of constructive rapid application development model (CORADMO). COPSEMO is based on the lifecycle anchoring concepts discussed by Boehm. The anchor points are defined as Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operational Capability (IOC). COPSEMO divide these anchor point in to four phases Inception, Elaboration, Construction, and Transition, COCOMO-II provide Scale factor with low, very low, high, very high levels these level value calculate in base line value calculation this provides PM_BS (Person Month Base) value and M_BS (Month Base) value and after that we apply these value in COPSEMO and it provides M = f(PM) to PM_BS; and P_BS from PM_BS/M_BS. After that COPSEMO output put in to cost drivers of new method then multiplication occurring and finally we find RAD schedule and effort for agile based projects A phased schedule and effort distribution is needed because the effects of the RAD strategies identified above are different for the different stages [9] [10]. Also, a new mathematical function is used to calculate (predict) the calendar months for a given amount of effort: the function is only radically different in low (under 16) person-month's efforts where it seems more normal have an equal number of people and months to accomplish the task. At the higher (greater than 120) person-month's efforts, the traditional COCOMO II-2000 function is used which is based on the traditional cube-root-like function of effort. A smooth curve is fit within these ranges [11].

2.4.1. Baseline Values Calculation

Effort Equation

$$\text{PM\_BS} = 2.97 * \text{EAF} * (\text{KSLOC})^\wedge \text{E}$$

where EAF = Is the Effort Adjustment Factor derived from the cost drivers; E = Is an exponent derived from the five scale drivers.

Schedule Equation

$$\text{M\_BS} = 3.67 * (\text{Effort})^\wedge \text{SE}$$

where EAF = Is the Effort from COCOMO-II Effort equation; E = Is the schedule equation exponent derived from the five scale drivers.

Person Productivity

$$\text{P\_BS} = \text{PM\_BS}/\text{M\_BS}$$

Now we calculate RAD (rapid application based development) PM_BS, M_BS for that we need to adjust PM_BS and M_BS by below given adjustments-PM_BS by adjusting work hours per months; M_BS by applying COPSEMO's M = f(PM) to PM_BS; and P_BS from PM_BS/M_BS [2]. Calculate Months_Base-Schedule (M_BS) as function of COCOMO II.2000 calculated effort (PM_CnoSCED).

$$\text{PM\_CnoSCED} = 2.97 * \text{EAF\_C} * \text{Power}(\text{Size}, \text{B\_C})/1000$$

where $\text{B\_C} = 0.91 + 0.01 * (\text{PREC\_R} + \text{FLEX\_R} + \text{RESL\_R} + \text{TEAM\_R})$.

Here-PREC_R (Precedentedss), FLEX_R (Development flexibility), RESL_R (Architecture/risk resolution), TEAM_R (Team cohesion, PMAT_R (Process maturity) are describing Scaling factors.

Now Calculation done on Person Month Base (PM_BS).

$$\mathbf{PM\_BS} = (\mathbf{152}/\mathbf{WHperMon}) * \mathbf{PM\_C}$$

where

WHperMon = Work hours per months (COCOMO-II.1998 default = 152)

Now Calculating Month Base (M_BS)

If PM_C < 16 then M_BS = Sqrt(PM_CnoSCED); and therefore P = M

If 16<=PM_C<64 then M_BS = then M_BS=(Mof64-4)/48*PMnoSCED)+(4-16*(Mof64-4)/48)

If PM=>64 then M_BS

=3.67*(PM_C^(0.28+0.2*(B_C-0.91)))*SCEDph/100

Finally get M_BS

```
M_BS = IF(PMnoSCED<16.001,SQRT(PMnoSCED),
IF(PMnoSCED<64,((Mof64-4)/48*PMnoSCED)
+(4-16*(Mof64-4)/48),3.67*POWER(PMnoSCED,
(0.28+0.2*(B- 0.91)))*SCEDph/100))
```

Here **Figure 1** shows that, COCOMO-II provide base line effort (PM), schedule (M) and also scale factors values, we applied these five scale factors values by adding all five values with different levels (L, VH, N, H, VH, EH) of adjustments on COPSEMO (Construction phase schedule effort model) and COPSEMO distribute COCOMO-II effort and schedule values in to four phases (Inception, Elaboration, Construction, Transition). And then applied above Base line calculation and find rapid application development based effort (PM), schedule (m) and by dividing both of them, we find person/staff (p) after this we multiplies newly added cost drivers (multipliers) values with different levels of cost drivers adjustments and find person month (Effort), Month (Schedule), Persons (P) for Agile based software development projects.

COCOMO-II Scale factor and SCED Baseline SCED, Effort

Scale Factor Inputs

Cost Drivers Input

User

Client

Using COPSEMO determine schedule and effort

$$Mphase = \prod_{driver=1}^{5} (Smult)\, driver, phase * Sphase$$

$$PMphase = \prod_{driver=1}^{5} (Emult)\, driver, phase * Ephase$$

Total schedule and effort become on inception Phase

$$Mcorad - agile \sum_{phase=inception}^{cos\,ntruction} Mphase$$

$$Mcorad - agile \sum_{phase=inception}^{cos\,ntruction} PMphase$$

Total schedule and effort become on Elaboration Phase

$$Mcorad - agile \sum_{phase=Elaboration}^{cos\,ntruction} Mphase$$

$$Mcorad - agile \sum_{phase=Elaboration}^{cos\,ntruction} pMphase$$

Total schedule and effort become on Construction Phase

$$Mcorad - agile \sum_{phase=Construction}^{cos\,ntruction} Mphase$$

$$Mcorad - agile \sum_{phase=Construction}^{cos\,ntruction} pMphase$$

Phase distribution (Inception, Elaboration, Construction, Transition) for calculate Person -Month, Month , and Person (Staff) values are given below.

$$\mathbf{PSE(Phase)distributed = (PM\_BS * Effort\%/100)/(M\_BS * Schedule\,\%\,100)}$$

1) For finding PM (Person-Month), M (Month) and P (Person) in Inception Phase-
   a) Inception: PM = PSE distributed*∏, M= PSE distributed*∏
   b) Inception: PM/M = P (Person)

where ∏ is the product of cost drivers values with respect to PM and M.

2) For finding PM (Person-Month), M (Month) and P (Person) in Elaboration Phase-
   a) Elaboration: PM = PSE distributed*∏, M= PSE distributed*∏
   b) Elaboration: PM/M = P (Person)

3) For finding PM (Person-Month), M (Month) and P (Person) in Construction Phase-
   a) Construction: PM = PSE distributed*∏, M = PSE distributed*∏
   b) Construction: PM/M = P (Person)

4) For finding PM (Person-Month), M (Month) and P (Person) in Transition Phase-
   a) Transition: PM = PSE distributed*∏, M = PSE distributed*∏
   b) Transition: PM/M = P (Person)

2.5. Experimental Analysis

Experimental analysis done over Mean Magnitude Relative Error (MMRE), here we have taken four actual projects input parameters from Promise Software Engineering project data web site and find MMRE by comparing both MRE of existing CORADMO method (effort. schedule) and Proposed method (effort, schedule), for that we need to calculate MRE by given equation [12].

$$\mathbf{MRE(effort)} = \frac{\mathbf{(Actual\ effort - predicted\ effort)}}{\mathbf{Actual\ effort}}$$

$$\mathbf{MRE(Schedule)} = \frac{\mathbf{(Actual\ schedule - predicted\ schedule)}}{\mathbf{Actual\ schedule}}$$

Also, experimental comparisons done in between CORAD_AGILE and COCOMO-II Ext. CORADMO with different projects inputs for cost drivers, scale factor line of codes these are all depends with different type of projects scenario.

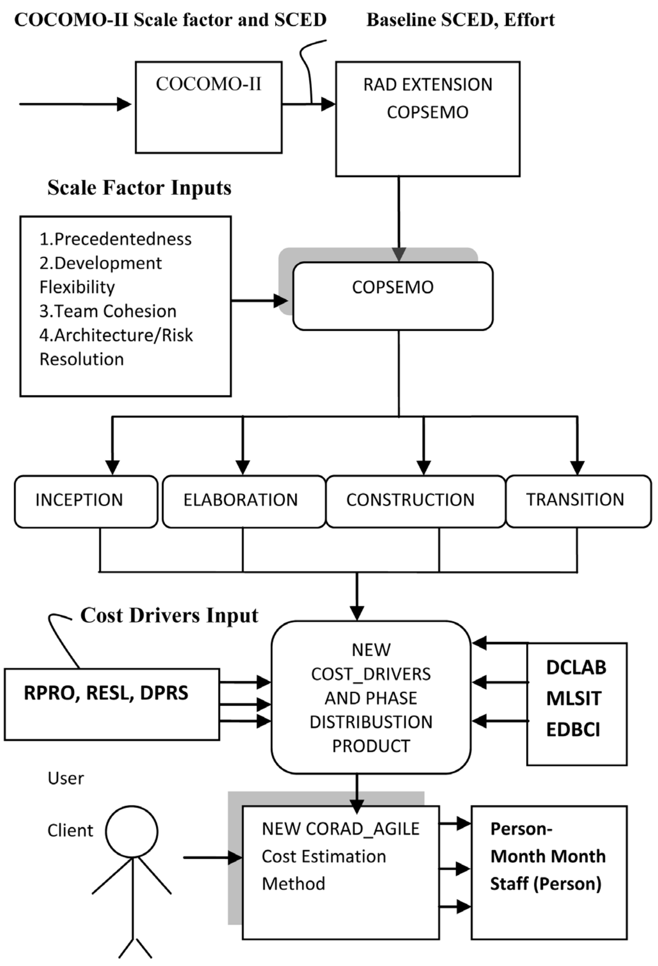

Figure 1. Proposed Model of RAD_AGILE Method.

First comparison done when Size = 1000 KSLOC, Scale Factor (SF) levels = High, Cost Drivers levels = Low, low, nominal, high, high, nominal after that 2nd comparison done when Size = 2000 KSLOC, Scale Factor (SF) levels = High, Cost Drivers levels = Low, high, nominal, nominal, low then 3rd comparison done on the basis of when Size = 2000 KSLOC, Scale Factor (SF) levels = High, Cost Drivers levels = High, extra high, nominal, nominal and Last comparison done when Size = 5000 KSLOC, Scale Factor (SF) levels = High, Cost Drivers levels = Nominal, Nominal, Nominal, high, high.

In Table 7 we find compassion results and if we look into the table and we can conclude Proposed Method Corad_Agile **having less Magnitude of Relative Error with respect Effort and Schedule** in compassion to existing COCOMO-II Extension CORADMO.

3. Conclusions

In our work we mainly focus on construction RAD-Agile model for rapid application development based software projects, and also we are focusing how to use COCOMO-II extension (CORADMO) model for design and develop an efficient RAD cost estimation model for Agile method (SCRUM, DSDM, ASD, AUP).

In conclusion, research on estimation has been conducted for decades with immense quantities of models and tools produced. This study has looked at the estimation process in the emerging field of agile development, and examined the causes of inaccurate estimates and steps to improve the process.

From the four case studies, a number of recommendations can be summarized as follows: fixed price budgets may be the best option for both developers and customers; multisite development very necessary; daily basis customer interaction with vendor employees is good for complete workloads fast and efficiently because daily

Table 7. Magnitude of relative error results.

MRE- when Size = 1000 KSLOC, SF = High, Drivers Level = Low, low, nominal, high, high, nominal										
	Actual Project Data		CORADMO-Effort, Schedule		MRE		Proposed Method_Effort, Schedule		MRE	
Phase	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)
Inception	0.40	0.52	0.24	0.31	0.39	0.41	0.28	0.35	0.30	0.33
Elaboration	0.78	0.68	0.47	0.38	0.40	0.44	0.52	0.45	0.33	0.34
Construction	8.88	6.66	5.27	4.07	0.41	0.39	5.88	4.45	0.34	0.33
Transition	3.68	2.02	2.15	1.10	0.42	0.46	2.34	1.34	0.36	0.34
MRE- when Size = 2000 KSLOC, SF = High, Drivers Level = Low, high, nominal, nominal, low										
	Actual Project Data		CORADMO-Effort , Schedule		MRE		Proposed Method_Effort, Schedule		MRE	
Phase	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)
Inception	1.32	0.59	0.88	0.38	0.33	0.36	0.90	0.40	0.32	0.33
Elaboration	3.23	1.45	2.20	0.94	0.32	0.35	2.00	0.99	0.38	0.32
Construction	9.60	6.98	7.01	4.54	0.27	0.35	6.23	4.58	0.35	0.34
Transition	5.55	1.34	3.53	0.78	0.36	0.42	4.05	0.79	0.27	0.41
MRE- when Size = 2000 KSLOC, SF = High, Drivers Level = High, extra high, nominal, nominal										
	Actual Project Data		CORADMO-Effort, Schedule		MRE		Proposed Method_Effort, Schedule		MRE	
Phase	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)
Inception	1.77	0.35	0.99	0.25	0.44	0.29	1.02	0.25	0.42	0.29
Elaboration	4.88	0.98	2.96	0.56	0.39	0.43	3.00	0.57	0.39	0.42
Construction	10.99	3.55	8.20	2.71	0.25	0.24	8.56	2.71	0.22	0.24
Transition	2.68	0.24	1.46	0.16	0.45	0.34	1.39	0.14	0.48	0.42
MRE- when Size = 5000 KSLOC, SF = High, Drivers Level = Nominal, Nominal, Nominal, high, high										
	Actual Project Data		CORADMO-Effort, Schedule		MRE		Proposed Method_Effort, Schedule		MRE	
Phase	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)	Effort (PM)	Schedule (M)
Inception	2.25	1.78	1.18	0.89	0.47	0.50	1.12	0.89	0.50	0.50
Elaboration	4.45	2.11	2.76	1.22	0.38	0.42	2.88	1.23	0.35	0.42
Construction	6.22	5.68	4.24	3.98	0.32	0.30	4.22	3.99	0.32	0.30
Transition	3.06	0.88	2.06	0.47	0.33	0.46	2.02	0.48	0.34	0.45

basis back locks (Sprint) can be remove by using this constraint; Collaboration support with all team member is good for rapidly work completion; and finally the most critical success factors for agile cost estimation is that experience and past project data should be documented and used to aid the estimation of subsequent projects.

References

[1] Cowan, S., Watson, P. and Cowan, S. (2011) An Agile Cost Estimating Methodology for Aerospace Procurement Operations: Genetic Causal Cost CENTRE-ing. www.intechopen.com

[2] Winsor Brown, A. (2002) Cyrus Fakharzadeh, CORADMO Extensions of COCOMO II.

[3] Boehm, B., Chulani, S. and Eyed, A. (1997) Knowledge Summary: USC-CSE Focused Workshop on Rapid Application Development. USC-CSE Technical Report.

[4] Boehm, B.W. (1981) Schedule Estimation Questionnaire. Prentice-Hall, Englewood Cliffs.

[5] Scale Factor Description with COCOMO-II. http://en.wikipedia.org/wiki/Scale_factor

[6] Khan Zia, Z., Kamal Tipu, S. and Khan Zia, S. (2012) An Effort Estimation Model for Agile Software Development. www.worldsciencepublisher.org

[7] Abbas, S.A., Liao, X.F., Ur Rehman, A., Azam, A. and Abdullah, M.I. (2012) Cost Estimation: A Survey of Well-Known Historic Cost Estimation Techniques. *Journal of Emerging Trends in Computing and Information Sciences*, **3**, 612-636. http://www.cisjournal.org

[8] Lang, M., Conboy, K. and Keaveney, S. (2012) Cost Estimation in Agile Software Development Projects. http://agilemanifesto.org/history.html

[9] Barbosa, C.M., Loubach, D.S. and Marques, A. (2010) An Estimation Model to Measure Computer Systems Development Based on Hardware and Software. http://mdp.ivv.nasa.gov

[10] Clark, B., Devnani-Chulani, S. and Boehm, B. (1998) Calibrating the COCOMO II Post-Architecture Model. http://sunset.usc.edu/Research_Group/Sunita/down/calpap.pdf

[11] Talby, D. and Dubinsky, Y. (2005) Governance of an Agile Software Project. http://domino.research.ibm.com/library/cyberdig.nsf/

[12] Attarzadeh, I. and Hock Ow, S. (2009) Software Development Effort Estimation Based on a New Fuzzy Logic Model. *International Journal of Computer Theory and Engineering*, **1**, 1793-8201.