Effect of Schedule Compression on Project Effort

Ye Yang, Zhihao Chen, Ricardo Valerdi, Barry Boehm
Center for Software Engineering, University of Southern California (USC-CSE)
Los Angeles, CA 90089-0781, USA
{yangy, zhihaoch, rvalerdi, boehm}@cse.usc.edu

Abstract

Schedule pressure is often faced by project managers and software developers who want to quickly deploy information systems. Typical strategies to compress project time scales might include adding more staff/personnel, investing in development tools, improving hardware, or improving development methods. The tradeoff between cost, schedule, and performance is one of the most important analyses performed during the planning stages of software development projects. In order to adequately compare the effects of these three constraints on the project it is essential to understand their individual influence on the project's outcome.

In this paper, we present an investigation into the effect of schedule compression on software project development effort and cost and show that people are generally optimistic when estimating the amount of schedule compression. This paper is divided into three sections. First, we follow the Ideal Effort Multiplier (IEM) analysis on the SCED cost driver of the COCOMO II model. Second, compare the real schedule compression ratio exhibited by 161 industry projects and the ratio represented by the SCED cost driver. Finally, based on the above analysis, a set of newly proposed SCED driver ratings for COCOMO II are introduced which show an improvement of 6% in the model estimating accuracy.

Keywords: Schedule compression, COCOMO II SCED Driver, software cost/effort estimation

I. Introduction

There are many cases in which software developers and project managers want to compress their project schedules. Numerous proactive and reactive project management methods for achieving schedule compression have been documented. Some examples are, hiring extra staff, investing on development tools, improving hardware, and improving development methods.

Software cost estimation models should be able to provide more reliable estimates of the number of staff needed and additional cost/resource needed in order to achieve schedule compression goals. Figure 1 shows the impact of schedule compression and stretch-out on development cost modeled in COCOMO II 2000 (CII), which is calibrated on 161 industry projects. In CII, a compression of 75% of the most efficient schedule is considered Very Low schedule compression level, and it will increase the development cost by 43%.

The estimated schedule significantly influences what actually happens in a project. If the schedule is under estimated, planning inefficiencies are introduced to the project. Invariably this can lead to delays in the project and increase the actual schedule. If the schedule is over estimated, Parkinson's Law can come into effect. The law claims that work expands to fill the time available for its completion. Allowing for extra time may also endanger the project with unexpected functions and unnecessary gold plating, potentially leading to increased schedules. Our work here is to help people improve their understanding of the effects of schedule compression and expansion on project effort estimates.

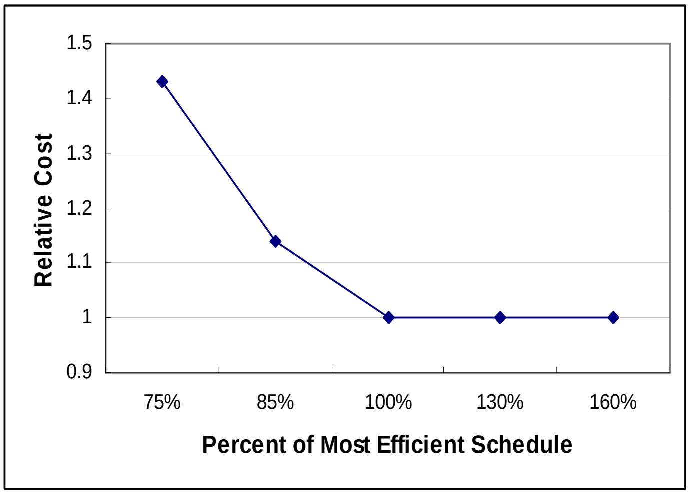


Figure 1 Relative Cost of Schedule Compression/ Expansion in COCOMO II

According to the 2001 CHAOS reports [1], 45% of projects exhibited cost overruns while 63% experienced schedule overruns. Even with the sophisticated development processes of today there needs to be better cost and resource control of software projects. Current software cost estimation models such as PRICE-S[1], SLIM[2], SEER-SEM[3], and COCOMO II[4] provide the necessary inputs for capturing the schedule implications of software development. Each model employs a slightly different philosophy towards schedule estimation. Understanding these philosophies can help software estimators select which approach best meets their development environment.

II. Overview of Schedule Compression Approaches in Cost Estimation Models

Schedule compression is recognized as a key cost element in many of the software estimation models in use today. Current models provide appropriate inputs for meeting particular scheduling needs but each treats schedule compression in its own unique way:

- PRICE-S tool has a built in cost penalty for deviations from the reference schedule
- SLIM accounts for schedule compression and lengthening in the range of impossible region and impractical region
- SEER-SEM calculates the optimal schedule and considers any schedule that is less than the optimal schedule impossible
- COCOMO II only includes penalties for schedule compression

[1] PRICE-S is a product of PRICE Systems, LLC. http://www.pricesystems.com/

[2] SLIM is a product of QSM, Inc. http://www.qsm.com/

[3] SEER-SEM is a product of Galorath, Inc. http://www.galorath.com

[4] COCOMO II is a product of the Center for Software Engineering at USC http://sunset.usc.edu

Aside from the different approaches all four aforementioned models agree with the fact that any acceleration during the design phase will increase the cost of software development.

The PRICE-S tool, developed by PRICE Systems, has a built in cost penalty for any deviations from the reference schedule. As shown in Figure 2 accelerations of this schedule will result in more people being thrown at the problem; increasing communication problems adding errors, and causing inefficiencies within the team. Schedule extensions will allow the people on the team to over-engineer the product, adding features and enhancements that are not necessary and often adding integration and test time to the cycle [2]. Several important things should be noted from the plot in Figure 2. First, there is a ten percent margin of error in the application of effects on labor hours, hence the flat line hovering between 90 percent and 110 percent. Within this range, staffing profiles can change without cost penalties. Second, acceleration in schedule has a much greater cost penalty than an extension in the schedule as indicated by the slope of the line.

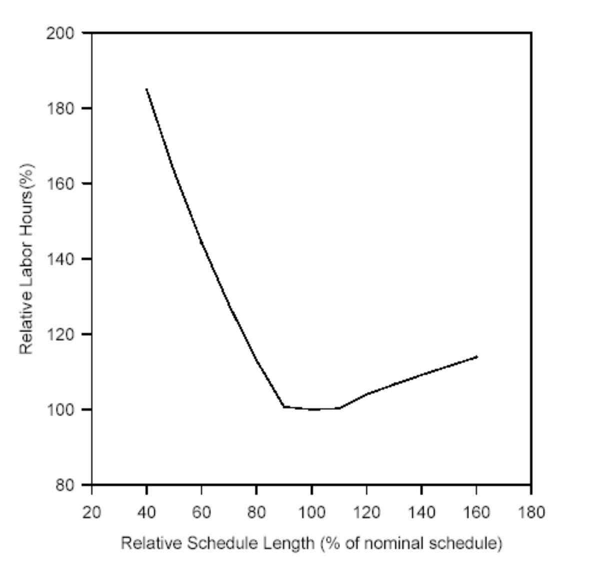


Figure 2. PRICE-S Schedule [2]

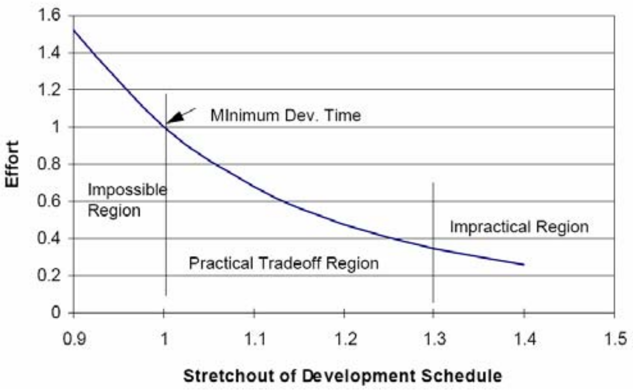


Figure 3. SLIM Schedule Effects [3]

SLIM, developed by Larry Putnam, treats schedule compression and expansion in terms of an impossible region and impractical region [3]. Theoretically the estimated schedule will fall in the practical tradeoff region [4]. Within the practical tradeoff region, if the schedule is compressed, the effort will increase exponentially; if the schedule is lengthened, further gains in reduced Effort trail off while “fast cycle time” is lost as shown in Figure 3.

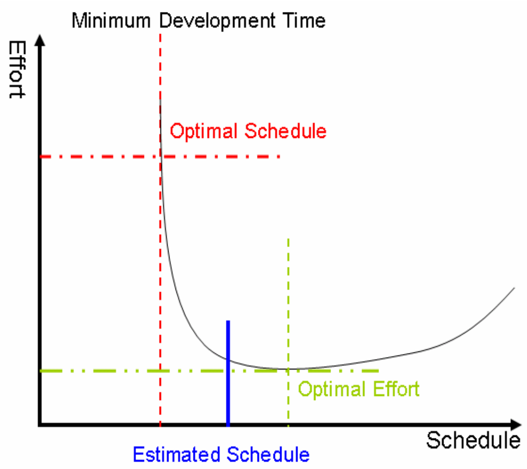


Figure 4. SEER-SEM Schedule Effects [5]

SEER-SEM, developed by Galorath Inc., takes effort as entropy of size which can vary based on user selected options. It takes schedule in a similar manner and less sensitive to size [5]. Its staffing profile shows the estimated effort spread over the estimated schedule. The model calculates the minimum development time based on an optimal schedule and assumes that the schedule that is less than optimal is impossible as illustrated in Figure 4.

III. Schedule Compression (SCED) in COCOMO

The COCOMO team at the University of Southern California continued to develop COCOMO 81 into COCOMO II during the mid 1990s to reflect the rapid changes in software development technologies and processes. The first version of COCOMO II was released in late 1995 [6], which described its initial definition and rationale. The current version of the COCOMO II model was calibrated on a 161 industry project database and released in 2000.

The COCOMO II Post-Architecture equation is shown in Equation 1. COCOMO II measures effort in calendar months where one month is equal to 152 hours (including development and management hours). The core intuition behind COCOMO-based estimation is that as systems grow in size, the effort required to create them grows somewhat exponentially. The scale factors in the model such as PREC (Precendentedness), FLEX (Development Flexibility), RESL (Architecture and Risk Resolution), TEAM (Team Cohesion) and PMAT (Process Maturity) are believed to represent the diseconomies of scale experienced in software development [7]. SCED is the effort multipliers used to estimate the additional effort resulting from schedule compress or expansion from the estimated nominal project schedule. This multiplier is one of 17 parameters that have a multiplicative effect on the effort estimation.

$$PM = A * (KSLOC^{(B+1.01*\sum_{i=1}^{5} SF_i}) * (\prod_{j=1}^{17} EM_j) \qquad (1)$$

Where

A = baseline Multiplicative Constant

B = baseline Exponential Constant

Size = Size of the software project measured in terms of KSLOC (thousands of Source Lines of Code) or Function Points and programming language.

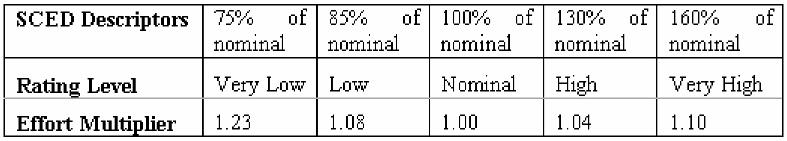
SF = Scale Factors including PREC, FLEX, RESL, TEAM, and PMAT

EM = Effort Multipliers including SCED

3.1. SCED in COCOMO 81

When the first version of the COCOMO model was developed, COCOMO 81, the effort multiplier Required Development Schedule (SCED) was used as a measure of the schedule constraint imposed on the project team developing software [8]. The life cycle phase used for the software development effort was divided into four phases: Requirements, Design, Code, and Integration & Test. The magnitude and phase distribution of the schedule constraint effects defined in COCOMO 81 are shown in Table 1.

Table 1. COCOMO 81 SCED Rating Scale



SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal
Rating Level	Very Low	Low	Nominal	High	Very High
Effort Multiplier	1.23	1.08	1.00	1.04	1.10

The nominal rating is always represented with a multiplier of 1.00 because it has neither a cost savings nor cost penalty on the project. Ratings above and below nominal correspond to a cost penalty due to the effects of schedule compression or expansion mentioned earlier.

3.2. SCED in COCOMO II.2000

In the COCOMO II model, the philosophy of the SCED driver underwent two significant changes. For one, the cost impact of schedule compression was almost doubled. The penalty for compressing the project schedule by 15% was increased from 8% to 14%. Similarly, the penalty for compressing the project schedule by 25% was increased from 23% to 43%. The message was clear: schedule compression has been observed to have a much greater impact than initially suspected. The second change in the SCED driver ratings was the reduction of the schedule expansion multipliers to 1.0. It was believed that, while schedule expansion would increase the amount of development effort, representation of this phenomenon could be captured in other drivers such as increased SLOC or Function Points. These changes are reflected in Table 2.

Table 2. COCOMO II SCED Rating Scale

SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal
Rating Level	Very Low	Low	Nominal	High	Very High
Effort Multiplier	1.43	1.14	1.00	1.00	1.00

Moreover, consider that schedule expansion usually leads to a reduction in team size. This can balance the need to carry project administrative functions over a longer period of time and thus not have a significant effect on overall cost.

3.3. Discussion

COCOMO II is calibrated to end-of-project actual size, actual effort in person-months, and ratings for the 17 cost drivers and 5 scale factors. Unlike the actual size and effort that can be accurately tracked by projects, ratings for the cost drivers and scale factors reflect the particular data reporter's subjective judgments. Such subjective judgments unavoidably carry the common bias of the counting conventions in a particular organization. As one of the 17 cost drivers in COCOMO II model, SCED is often rated inaccurately due to such common bias. For example, organizations where people are accustomed to compressed schedules and take them for granted will inaccurately report SCED ratings.

However, actual schedule compression can be computed for the projects in the COCOMO repository and compared to the reported SCED ratings shown in Figure 5. In the following sections, two experimental studies are described that provide insights about the real effects of SCED driver, and show the differences between the actual SCED versus reported SCED ratings.

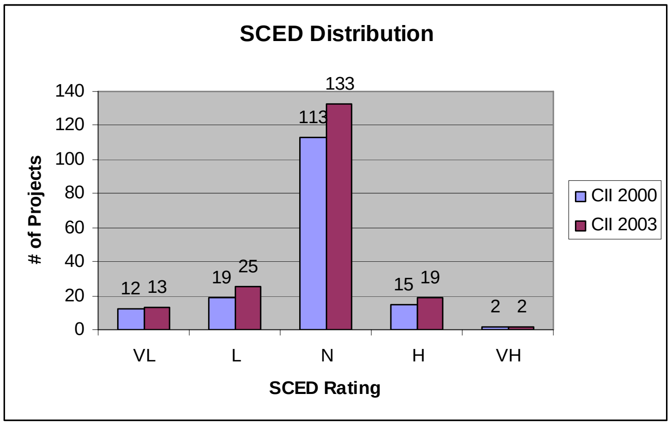


Figure 5. SCED on COCOMO II Database

VI. Two kinds of SCED Experiments

In order to arrive at a rating scale that could better capture the effects of schedule compression and expansion two datasets were analyzed: the COCOMO II 2000 dataset which consists of 161 projects and the COCOMO II 2004 dataset which consists of 192 projects. The "CII 2003" data shown in Figure 5 corresponds to the COCOMO II 2004 dataset since the data was collected in 2003 and analyzed in 2004. Almost half of the 192 projects currently in the COCOMO II 2004 database are from COCOMO II 2000 dataset. These two data sets were treated separately in two different experiments. The first involved a calculation of SCED quality and the second a derivation of the ideal effort multiplier. The result of these experiments helped determine a new SCED rating scale that improves the model accuracy by as much as 6%.

Experiment I: SCED Rating Quality

This experiment was performed on COCOMO II 2000 database and COCOMO II 2004 to determine the rating of SCED quality. Since it is recognized that the SCED rating in every data point comes from a subjective judgment, we try to logically derive a more accurate SCED rating by analyzing the data. To calculate the *Derived SCED*, we computed the estimated effort without *Rated SCED* using Equation 2 and use that effort to calculate the estimated schedule *TDEV_est* in Equation 3 then we calculate the schedule compression ratios using Equation 4 to determine the *Derived SCED.* We can obtain a quality of the *SCED* rating for each project by comparing the *Derived SCED* and the *Rated SCED*. The steps being performed in this experiment are show in the figure 6.

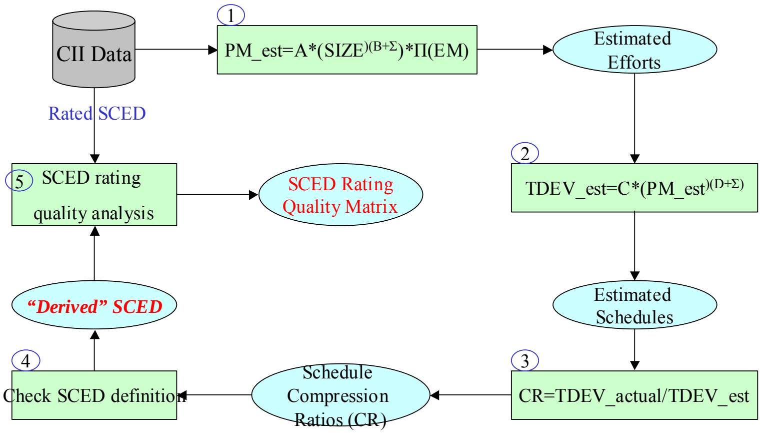

CII Data
1
PM_est=A*(SIZE)^(B+Σ)*Π(EM)
Estimated Efforts
Rated SCED
2
TDEV_est=C*(PM_est)^(D+Σ)
5
SCED rating quality analysis
SCED Rating Quality Matrix
Estimated Schedules
"Derived" SCED
4
Check SCED definition
Schedule Compression Ratios (CR)
3
CR=TDEV_actual/TDEV_est

Figure 6. SCED Rating Quality Study Steps

Step 1: Compute estimated effort with assuming schedule is nominal

$$Estimated\_effort = A*(KSLOC)^{(B+0.01*(\sum_{i=1}^{5} SF_i))} *(\prod_{j=1}^{16} EM\_But\_SCED_j) \qquad (2)$$

$$TDEV\_est = C*(PM_{est})^{(D+0.2*(E-B))} \qquad (3)$$

$$CR = TDEV_{actual} / TDEV_{est} \qquad (4)$$

where

- A, B are model constants, calibrated for each different version of COCOMO model.
- C is schedule coefficient that can be calibrated

- D is scaling base-exponent for schedule that can be calibrated
- E is the scaling exponent for the effort equation
- SF_i are five scale factors including PMAT, PREC, TEAM, FLEX, and RESL;
- $EM\_But\_SCED_j$ are effort multipliers except SCED, including RELY, DATA, CPLX, RUSE, DOCU, TIME, STOR, PVOL, ACAP, PCAP, PCON, APEX, PLEX, LTEX, TOOL, and SITE.
- A nominal Schedule is under no pressure, which means no schedule compression or expansion; initially set to 1.0.

Step 2: Compute estimated schedule *TDEV_est*

Nominal Schedule. Based on COCOMO II post-architecture model's effort and schedule estimation, a nominal schedule can be estimated based on the cost driver ratings and estimated effort (in person-months) according equations 3 and 4.

Step 3: Compute Actual Schedule Compression/Stretch-out Ratio (SCR)

Actual Schedule. Every data point comes with an actual schedule. For example, in COCOMO II, it is named TDEV (time to development).

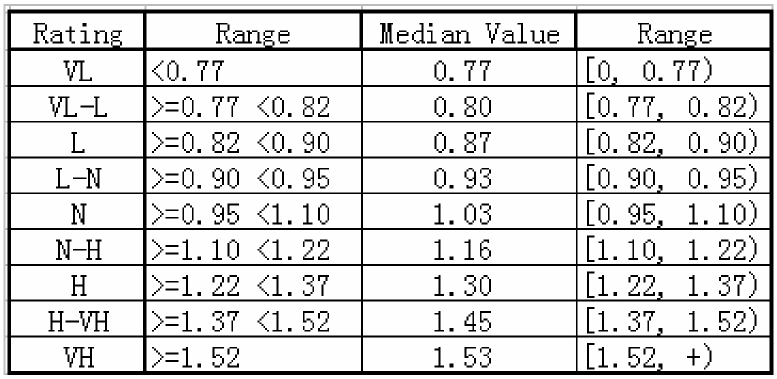
Actual Schedule Compression/Stretch-out Ratio (SCR). The SCR can be easily derived through the following equation:

$$SCR = Actual\quad Schedule / Derived\quad SCED \qquad (5)$$

For example, if a project's TDEV is 6 month, and the estimated nominal schedule is about 12 month, then we consider the actual schedule compression as 50% (= 6 / 12).

Step 4: Obtain "derived" SCED rating

Table 3. SCED Rating table



Rating	Range	Median Value	Range
VL	<0.77	0.77	[0, 0.77)
VL-L	>=0.77 <0.82	0.80	[0.77, 0.82)
L	>=0.82 <0.90	0.87	[0.82, 0.90)
L-N	>=0.90 <0.95	0.93	[0.90, 0.95)
N	>=0.95 <1.10	1.03	[0.95, 1.10)
N-H	>=1.10 <1.22	1.16	[1.10, 1.22)
H	>=1.22 <1.37	1.30	[1.22, 1.37)
H-VH	>=1.37 <1.52	1.45	[1.37, 1.52)
VH	>=1.52	1.53	[1.52, +)

Using equation 4, we compute the actual schedule compressions/stretch-outs ratios, look up in the SCED driver definition shown in Table 3 and check for the closest matched SCED ratings. Then a new set of *SCED* ratings is produced that more accurately reflect the project's schedule compression.

Step 5: Compare "derived" and "rated" SCED to analyze SCED Rating Quality

Table 4. SCED Rating Quality Analysis in COCOMO II 2000 database

161 Projects

SCED Derived from the experiment

SCED Reported In Data

SCR	[0, 0.77)	[0.77, 0.82)	[0.82, 0.90)	[0.90, 0.95)	[0.95, 1.10)	[1.10, 1.22)	[1.22, 1.37)	[1.37, 1.52)	[1.52, +)
	VL	VL-L	L	L-N	N	N-H	H	H-VH	VH
VL	7		1		2	1			
VL-L	2			1	3				
L	5	3	2	1	4	1	1		
L-N			3	1	4	1			1
N	19	2	11	11	14	12	17	4	9
N-H	1								1
H	2			1	4	2	2		3
H-VH									
VH	1							1	

Table 5. SCED Rating Quality Analysis in COCOMO II 2004 database

119 Projects

SCED Derived from the experiment

SCED Reported In Data

SCR	[0, 0.77)	[0.77, 0.82)	[0.82, 0.90)	[0.90, 0.95)	[0.95, 1.10)	[1.10, 1.22)	[1.22, 1.37)	[1.37, 1.52)	[1.52, +)
	VL	VL-L	L	L-N	N	N-H	H	H-VH	VH
VL	2		2				1		
VL-L	1		3		1				
L	3	2	1	1	1		1		2
L-N			3	2	3	1			
N	7	4	12	7	14	10	12	2	9
N-H	1						1		
H	1					2	3	2	
H-VH									
VH	1							1	

The comparison of derived SCED and rated SCED is shown in Table 4 and 5 for the two data sets. We use the term of SCED Rating Quality to measure how close the subjective ratings are to the new ratings. From Table 4, it is evident that only 26 projects out of 161 rated SCED the same as the derived ones. From Table 5, only 22 out 119 projects are rated the same level. SCED was reported as nominal in 99 projects in Table 4 and 77 projects in Table 5. When people had no idea about the estimated schedule, they probably reported it as nominal. It also shows that the estimated schedule by people's intuition is very likely wrong, and most of them are too optimistic as the number of Very Low derived SCED are much bigger than that of rated SCED from two tables. Later we show that higher values of SCED improve the model accuracy.

The current rationale in COCOMO II is that stretch-outs do not add or decrease effort shown Figure 1. But our experiments show the current rationale in COCOMO II is not exactly right as schedule stretch-outs do bring additional effort. If schedule stretch-outs do bring additional effort, what are the values for different levels of schedule stretch-outs? To find the answer and to validate our conclusion, we conduct another experiment.

Experiment II: Ideal Effort Multiplier (IEM) Analysis on SCED

Methods exist to normalize out contaminating effects of individual cost driver attributes in order to get a clear picture of the contribution of that driver (in our case, the SCED) on development productivity [9]. We slightly modified the original definition as our working definition*:*

> *For the given project P, compute the estimated development effort using the COCOMO estimation procedure, with one exception: do not include the effort multiplier for the cost driver attribute (CDA) being analyzed. Call this estimate PM(P,CDA). Then the ideal effort multiplier, IEM(P, CDA), for this project/cost-driver combination is defined as the multiplier which, if used in COCOMO, would make the estimated development effort for the project equal to its actual development effort PM(P, actual). That is*

$$IEM(P, SCED) = PM(P, actual) / PM(P, SCED) \qquad (5)$$

Where

- IEM(P, SCED): the ideal effort multiplier for project P
- PM(P, actual): project P's actual development effort
- PM(P, SCED): CII estimate excluding the SCED driver
- PM: Person-Months

Steps for IEM-SCED analysis

The following steps were performed to complete the IEM-SCED analysis on the COCOMO II database.

1) Compute the PM(P, CDA), using the following formula

$$PM(P, CDA) = A * (KSLOC)^{(B+0.01*(\sum_{i=1}^{5} SF_i))} * (\prod_{j=1}^{16} EM\_But\_SCED_j) \qquad (6)$$

where

- A, B are model constants, calibrated for each different version of COCOMO model.
- SF_i are five scale factors including PMAT, PREC, TEAM, FLEX, and RESL;
- $EM\_But\_SCED_j$ are effort multipliers except SCED, including RELY, DATA, CPLX, RUSE, DOCU, TIME, STOR, PVOL, ACAP, PCAP, PCON, APEX, PLEX, LTEX, TOOL, and SITE.

2) Compute the IEM(P,CDA) using equation 5

3) Group IEM(P, CDA) by the same SCED rating (i.e. VL, L, N, H, VH)

These groupings are shown in Figures 6 and 7.

4) Compute the median value for each group as IEM-SCED value for that rating.

This step involves the computation of the median value of IEM-SCED for each rating level. These are summarized in Table 6 and grouped by COCOMO II 2000 and 2004 databases.

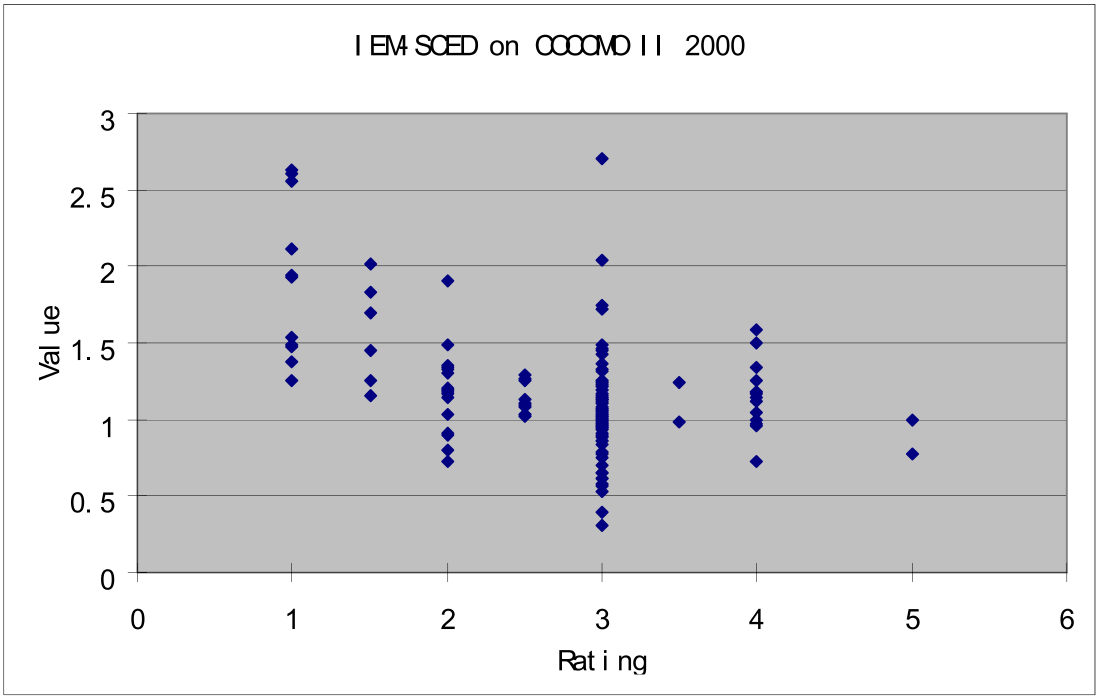


Figure 6. IEM-SCED Group Distributions for CII2000

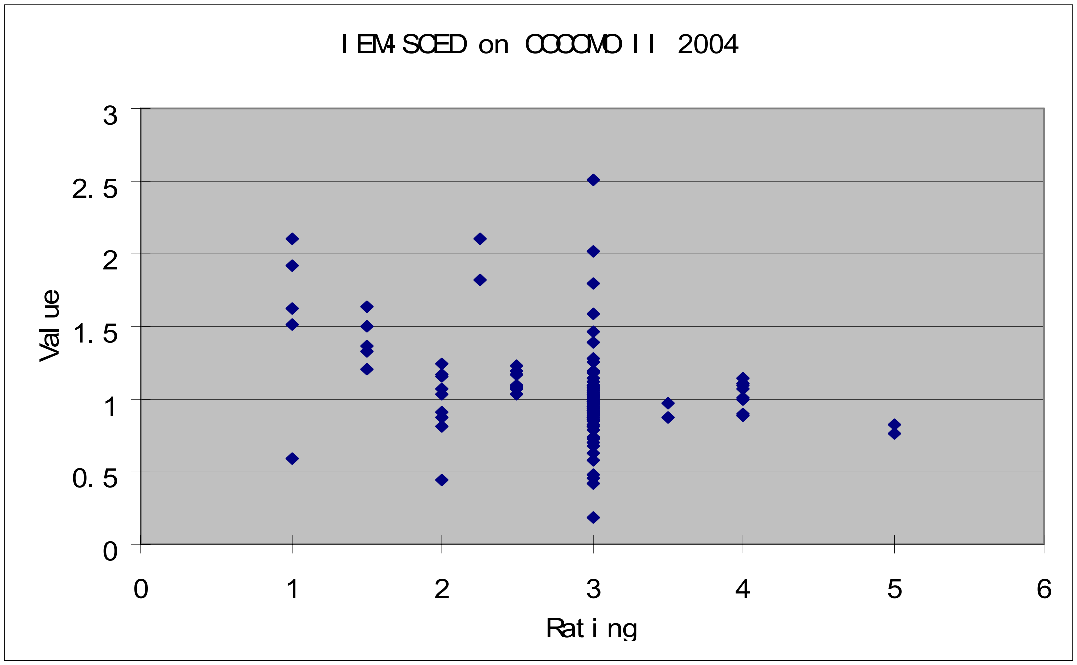


Figure 7. IEM-SCED Group Distributions for CII2004

Given that extreme values (outliers) exist in our databases. Those outliers could give great impact to the mean values. To avoid that, the median value is used since it is not as sensitive to outliers.

SCED Rating	VL	L	N	H	VH
IEM-SCED in CII 2000	1.94	1.2	1.04	1.16	0.88
IEM-SCED in CII 2004	1.62	1.17	0.98	1.04	0.78

Table 6. IEM-SCED Analysis Results of CII 2000 and 2004 databases

Comparison of IEM results and COCOMO II 2000

To compare the SCED cost driver's effects in different databases, the IEM-SCED values from Table 6 and the SCED values in COCOMO 81 and COCOMO II 2000 are plotted in Figure 7.

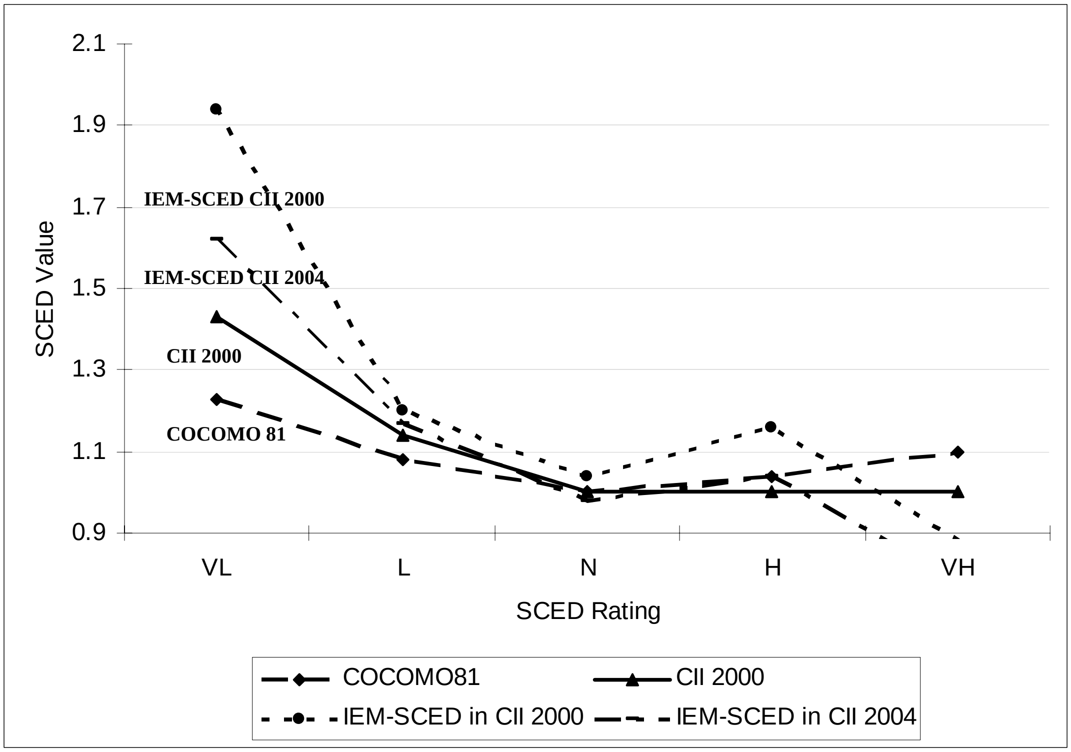


Figure 7. SCED Comparison

The diamond-dashed line shows the SCED driver values used in COCOMO 81. Its V-shape implies that either schedule compression or expansion will cause increase on project development effort. The solid-triangle line shows current SCED driver values in COCOMO II 2000, where the VL rating for SCED is increased from 1.23 to 1.43 as discussed earlier. However, for ratings above Nominal, the line remains flat indicating that schedule expansion does not add or decrease effort. One underlying explanation might be that the savings due to small team size are generally balanced by the need to carry project administrative functions over a longer period of time. The remaining lines show the SCED ratings derived from the IEM-SCED analysis using the COCOMO II 2000 and COCOMO II 2004 datasets. In both cases, the lines have steeper slopes from Very Low to Low and Low to Nominal rating levels, meaning that the effect of schedule compression on increasing development cost is enhanced. Another observation is that the levels above Nominal demonstrate a relatively different shape than accustomed to. The projects with a

High IEM-SCED value exhibited some cost penalty while the projects with a Very High IEM-SCED value exhibited cost savings. Further investigation is needed to determine real-life reasons why this could happen on software development projects.

Model accuracy with IEM results

Efforts to calibrate COCOMO II with the 2004 data set following the Bayesian Calibration approach are still ongoing [10]. In the meantime, we have applied the derived IEM-SCED values back to the well-calibrated CII 2000 database and have seen an improvement in the model's accuracy. The increased accuracy is shown in Table 7.

Table 7. Accuracy Analysis Results of COCOMO II 2000

Database		Pred(20)	Pred(25)	Pred(30)
CII 2000	W/O IEM	58%	65%	72%
	With IEM	61%	71%	76%

The table shows that by applying the IEM-SCED values into the CII model, all three accuracy levels - Pred(20), Pred(25), and Prec(30) - increase by 3%, 6%, and 4%.

V. Conclusions

Software development has changed dramatically in the last decade as a result of new applications that have enabled faster development. Managing the development schedule remains a key aspect of reliable and timely development. The current cost models available, including COCOMO II, provide ways to quantify the impact of schedule compression or expansion. All of these models agree with the idea that acceleration of schedules will increase cost. But the problem is that reported SCED ratings often differ greatly from what actually happened on the project. We have shown the current ratings for the COCOMO II SCED driver do not adequately reflect the impact of schedule fluctuation. As such we have developed new ratings that better reflect the cost impact of schedule changes. The new ratings resulted in improvements in CII 2000 model accuracies, i.e. by 3% for Pred(20), 6% for Pred(25), and 4% for Pred(30). While this may seem insignificant, these improvements are powerful considering that COCOMO II has 22 parameters. Additionally, the SCED Rating Quality metric illustrates a significant difference between the reported schedule and the actual schedule; confirming that subjective assessments of schedule are often incorrect.

A number of opportunities exist for future work in the area of schedule estimation using COCOMO II. For one, the dynamic range of the rating scale could be expanded to cover projects which are compressed by more than 25 percent or expanded by more than 60 percent. Secondly, a new method of collecting schedule information needs to be developed to improve the reliability of the SCED driver. Currently there is a significant difference between the reported schedule and the actual schedule – introducing measurement error in the model. In order to overcome this, project schedule could be determined by the start and end date and compared to the original baseline schedule. Thirdly, the effect of local calibration should be accommodated through some new methods or metrics when it comes for a general model like COCOMO to do a “full” calibration. This is because differences in local calibrated model parameters might cause bias when trying to understand and study on one single driver. These metrics can be used to derive the actual schedule compression, if any, and improve the reliability of schedule estimation.

VI. References

[1] CHAOS 2001, http://standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf

[2] Your Guide to PRICE-S: Estimating Cost and Schedule of Software Development and Support, PRICE Systems, LLC, Mt. Laurel, NJ, 1998.

[3] Lawrence H. Putnam, Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers, New York: The Institute of Electrical and Electronics Engineers, Inc., 1980.

[4] Lawrence H. Putnam, MEASURES FOR EXCELLENCE Reliable Software on Time, within Budget, Englewood Cliffs: Yourdon Press., 1992.

[5] SEER-SEM, http://www.galorath.com

[6] Boehm, B., B. Clark, E. Horowitz, C. Westland, R. Madachy, R. Selby. “Cost Models for Future Software Life Cycle Processes: COCOMO 2.0”, Annals of Software Engineering Special Volume on Software Process and Product Measurement, J.D. Arthur and S.M. Henry, Eds., J.C. Baltzer AG, Science Publishers, Amsterdam, The Netherlands, Vol. 1, pp. 45 - 60, 1995

[7] Devnani-Chulani, S. “Bayesian Analysis of Software Cost and Quality Models", unpublished Ph.D. Dissertation, University of Southern California, May 1999.

[8] Boehm, B. W., Software Engineering Economics. Prentice-Hall, 1981.

[9] Boehm, B., Abts, C., Brown, A., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D., and Steece, B., Software Cost Estimation with COCOMO II, Prentice Hall, 2000.

[10] Sunita Chulani, Barry W. Boehm, Bert Steece: Bayesian Analysis of Empirical Software Engineering Cost Models. IEEE Trans. Software Eng. 25(4): 573-583, 1999.

Biographies:

Ye Yang

Ye is a Research Assistant at the Center for Software Engineering and a PhD student of Computer Science Dept. at the University of Southern California. Her research interests include software cost estimation modeling for COTS-based systems and Product Line Investment based on COCOMO II model, and process modeling and risk management for COTS-based application development. She received her major bachelor degree in Computer Science and minor bachelor degree in Economics from Peking University, China in 1998, and her Master degree in Software Engineering from Chinese Academy of Sciences in 2001.

Zhihao Chen

Zhihao is a PhD student at USC doing research in Software Engineering under Professor Barry W. Boehm. His research is focused on empirically based Software Engineering – empirical methods and model integration, which support the generation of an empirically based software development process covering high level lifecycle models to low level techniques, provide validated guidelines/knowledge for selecting techniques and models and serves, and help people better understand such issues as what variables affect cost, reliability, and schedule, and integrating existing data and models from the participants and all collaborators. He also focuses on software project management and cost estimation. Previously, he got his bachelor and master of computer science from South China University of Technology. He previously worked for HP and EMC.

Ricardo Valerdi

Ricardo is a Research Assistant at the Center for Software Engineering and a PhD student at the University of Southern California in the Industrial and Systems Engineering department. His research is focused on the cost modeling of systems engineering work. While completing a Masters degree in Systems Architecting & Engineering at USC he collaborated in the creation of COSYSMO (Constructive Systems Engineering Model). He earned his bachelor's degree in Electrical Engineering from the University of San Diego. Ricardo is currently a Member of the Technical Staff at the Aerospace Corporation in the Economic & Market Analysis Center. Previously, Ricardo worked as a Systems Engineer at Motorola and at General Instrument Corporation.

Barry Boehm

Barry is the TRW professor of software engineering and director of the Center for Software Engineering at the University of Southern California. He was previously in technical and management positions at General Dynamics, Rand Corp., TRW, and the Defense Advanced Research Projects Agency, where he managed the acquisition of more than $1 billion worth of advanced information technology systems. He originated the spiral model, the Constructive Cost Model, and the stakeholder win-win approach to software management and requirements negotiation.