COQUALMO[1] Data Collection Questionnaire

1. Introduction

The Center for Software Engineering at the University of Southern California is conducting research to update the software development cost estimation model called COCOMO. The project name is COCOMO II and is led by Dr. Barry W. Boehm.

A fundamental requirement for such research is real-world software development project data. This data will be used to test hypotheses and verify the model's postulations. In return the model will be open and made available to the public. The contribution of your data will ensure the final model is useful.

The data that is contributed is important to us. We will safeguard your contribution so as not to compromise company proprietary information. Some Affiliates have an active collection program and the data from past projects is available for the COCOMO II data collection effort. This questionnaire can be used to extract relevant COCOMO II data. A rosetta-stone that converts COCOMO '81 data to COCOMO II data is also available. Please contact us if you would like to get a copy.

This questionnaire attempts to address two different levels of data granularity: project level and component level. The project level of granularity is data that is applicable for the whole project. This includes things like application type and development activity being reported. Component level data are things like size, cost, and component cost drivers. If the data being submitted is on a project that has multiple components then fill out the project data once, and the component data for each of the identifiable component. If the data being submitted is for the whole project fill out the form once. The data collection activity for the COCOMO II research effort started in November 1994. The first calibration was published in 1997 based on 83 datapoints collected. It became popular as COCOMO II.1997 and produced estimates within 30% of the actuals 52% of the time for effort. The second calibration was published in 1998 based on 161 datapoints. It is known as COCOMO II.1998 and produces estimates within 30% of the actuals 71% of the time for effort. The aim of the COCOMO II research team is to continually update the existing COCOMO II database and to publish annual calibrations of the COCOMO II model. Hence by submitting your data to us, you play a significant role in the model calibration.

COCOMO II Points of Contact

For questions on the COCOMO II Model and its extensions, data definitions, or project data collection and management, contact:

A. Winsor Brown (Research Scientist)	Voice: (213) 740-6599, Fax: (213) 740-4927
Barry Boehm (Project Leader)	Voice: (213) 740-8163, Fax: (213) 740-4927
Internet Electronic-Mail	*cocomo-info@sunset.usc.edu*

COCOMO II Data Submission Address:

COQUALMO Data Submission
Center for Software Engineering
Department of Computer Science
Henry Salvatori Room 330
University of Southern California
941 W. 37th Place
Los Angeles, CA 90089-0781
U.S.A.

[1] COnstructive QUALity MOdel (COQUALMO) is the quality model extension to COCOMOII.

2. Project Level Information

As described in the Introduction section of this questionnaire, project level information is applicable for the whole project. This includes things like application type and development activity being reported.

2.A General Information

2.1 Affiliate Identification Number. Each separate software project contributing data will have a separate file identification number of the form XXX. XXX will be one of a random set of three-digit organization identification numbers, provided by USC Center for Software Engineering to the Affiliate.

_____

2.2 Project Identification Number. The project identification is a three digit number assigned by the organization. Only the Affiliate knows the correspondence between YYY and the actual project. The same project identification must be used with each data submission.

_____

2.3 Date prepared. This is the date the data elements were collected for submission.

_____

2.4 Application Type. This field captures a broad description of the type of activity this software application is attempting to perform.

Circle One:	Command and Control,	MIS,	Simulation,
	Communication,	Operating Systems,	Software Dev. Tools,
	Diagnostics,	Process Control,	Testing,
	Engineering and Science	Signal processing,	Utilities

Other:_____

2.5 Development Type.

Is the development a new software product or an upgrade of an existing product?

Circle One: New Product Upgrade

2.6 Development Process. This is a description of the software process used to control the software development, e.g. waterfall, spiral, etc.

_____

2.7 Step in Process. This field captures information about the project's position in its development process. The answers depend on the process model being followed.

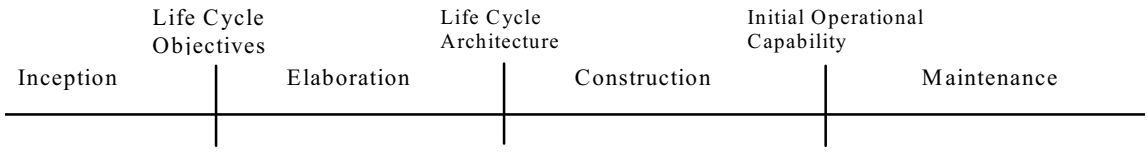
2.7.1 Activity. This field captures the waterfall phase of development that the project is in. For one-time reporting the activity is `completed'. It is assumed that data for completed projects includes data from software requirements through integration/test. Please report the correct phasing if this is not the case.

Circle One: Requirements, Design, Code,
Unit Test, Integration/Test, Maintenance,
Completed

Other:_____

2.7.2 Stage. "Stage" refers to the aggregate of activities between the life cycle anchor points[2]. The four stages, based on Rational's Objectory Process[3], and the anchor points are shown on the timeline below. Please see section 2 of the "CORADMO[4] Extensions of COCOMO II Schedule Estimation Questionnaire" for more details.

Please circle the most advance anchor point (milestone) the project has achieved.



See the Appendix A for definitions of the LCO, LCA, and IOC milestones. The COCOMO II model covers the effort required from the completion of the LCO to IOC. If you are using a waterfall model, the corresponding milestones are the Software Requirements Review, Preliminary Design Review, and Software Acceptance Test.

2.8 Development Process Iteration. If the process is iterative, e.g. spiral, which iteration is this?

_____

2.9 COCOMO Model. This specifies which COCOMO II model is being used in this data submission. If this is a "historical" data submission, select the Post-Architecture model or the Applications Composition model.

- Application Composition: This model involves prototyping efforts to resolve potential high risk issues such as user interfaces, software/system interaction, performance, or technology maturity.
- Early Design: This model involves exploration of alternative software/system architectures and concepts of operations. At this stage of development, not enough is known to support fine-grain cost estimation.
- Post-Architecture: This model involves the actual development and maintenance of a software product. This stage of development proceeds most cost-effectively if a software life-cycle architecture has been developed; validated with respect to the system's mission, concept of operation, and risk; and established as the framework for the product.

Circle One: Application Composition, Early Design, Post-Architecture

[2] Barry W. Boehm, "Anchoring the Software Process," *IEEE Software,* 13, 4, July 1996, pp. 73-82. An unabridged version, dated November 1995, is in Appendix A.
[3] Rational Corp., "Rational Objectory Process 4.1 – Your UML Process", available at http://www.rational.com/support/techpapers/toratobjprcs/.
[4] COnstructive RAD schedule and effort MOdel

2.10 Success Rating for Project. This specifies the degree of success for the project.

- Very successful; did almost everything right
- Successful; did the big things right
- OK; stayed out of trouble
- Some Problems; took some effort to keep viable
- Major Problems; would not do this project again

Circle One: Very Successful Successful OK Some Problems Major Problems

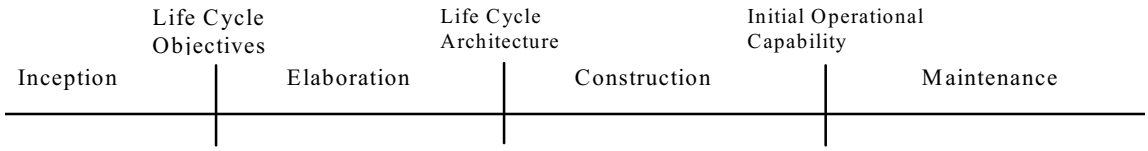
Schedule

2.11 Year of development. For reporting of historical data, please provide the year in which the software development was completed. For periodic reporting put the year of this submission or leave blank.

_____

2.12 Schedule Months. For reporting of historical data, provide the number of calendar months from the time the development began through the time it completed. For periodic reporting, provide the number of months in this development activity.

Circle the life-cycle stages that the schedule covers:



Schedule in months: _____

2.B COQUALMO (COCOMO QUALITY MODEL)

This subsection has additional questions related to the **CO**COMO **QUAL**ITY **MO**DEL. Please fill this out if you have data on quality metrics.

2.18 Severity of Defects. Categorize the several defects based on their severity using the following classification[5] information:

- **Critical**

  Causes a system crash or unrecoverable data loss or jeopardizes personnel.

  The product is unusable (and in mission/safety software would prevent the completion of the mission).

- **High**

  Causes impairment of critical system functions and no workaround solution exists.

  Some aspects of the product do not work (and the defect adversely affects successful completion of mission in mission/safety software), but some attributes do work in its current situation.

- **Medium**

  Causes impairment of critical system function, though a workaround solution does exist.

  The product can be used, but a workaround (from a customer's preferred method of operation) must be used to achieve some capabilities. The presence of medium priority defects usually degrades the work.

- **Low**

  Causes a low level of inconvenience or annoyance.

  The product meets its requirements and can be used with just a little inconvenience. Typos in displays such as spelling, punctuation, and grammar which generally do not cause operational problems are usually categorized as low severity.

- **None**

  Concerns a duplicate or completely trivial problem, such as a minor typo in supporting documentation.

COQUALMO accounts for only Critical, High and Medium severity defects.

2.19 Defect Introduction by Artifact, Stage and Severity. The software development process can be viewed as introducing a certain number of defects into each software product artifact. Stage refers to the aggregate of activities between the life cycle anchor points. The four stages, based on Rational's UML Process, and the anchor points are shown on the timeline below. Please see section 2 of the "CORADMO Extensions of COCOMO II Schedule Estimation Questionnaire" for more details. Enter the number of defects introduced in the several artifacts involved in the software development process.

A Requirements Defect is a defect introduced in the Requirements Activity and a Design Defect is a defect introduced in the Design activity and so on and so forth.

LCO **LCA** **IOC**

Stage / Artifact	Inception	Elaboration	Construction
Requirements			
Design			
Code			

[5] Adapted from IEEE Std 1044.1-1995

2.19.1 Requirements Defects

Severity	Urgent	High	Medium	Low	None
No. of Requirements Defects					

2.19.2 Design Defects

Severity	Urgent	High	Medium	Low	None
No. of Design Defects					

2.19.3 Code Defects

Severity	Urgent	High	Medium	Low	None
No. of Code Defects					

2.20 Defect Removal by Artifact, Stage and Capability

Throughout the development life cycle, defect removal techniques are incorporated to eliminate defects before the product is delivered. Enter the number of defects removed in each stage of the software development process.

LCO LCA IOC

Stage / Artifact	Inception	Elaboration	Construction
Requirements			
Design			
Code			

COQUALMO models defect removal by classifying defect removal capabilities into three relatively orthogonal profiles with each profile having 5 levels of increasing defect removal capability, namely 'Very Low', 'Low', 'Nominal', 'High' and 'Very High' with 'Very Low' being the least effective and 'Very High' being the most effective in defect removal.

2.20.1 Automated Analysis

	Very Low	Low	Nominal	High	Very High	Don't know
Rating Scale	Simple compiler extensions for static module-level code analysis, syntax, type-checking.	Simple compiler extensions for static module-level code analysis, syntax, type-checking. Basic requirements and design consistency, traceability checking	Intermediate-level code syntax and semantic analysis. Simple requirements/ design view consistency checking	More elaborate requirements/design view consistency checking. Basic distributed-processing and temporal analysis, model checking, symbolic execution	Formal specification and verification. Advanced distributed processing and temporal analysis, model checking, symbolic execution	
Your rating						

2.20.2 People Reviews

	Very Low	Low	Nominal	High	Very High	Don't know
Rating Scale	Ad-hoc informal walkthroughs Minimal preparation, follow-up	All to the left, plus well-defined sequence of preparation, review, follow-up. Informal review roles and procedures	All to the left, plus formal review roles and procedures applied to detailed design and code reviews	All to the left, plus formal review roles and procedures applied to specification, design code, test, documentation artifacts. Basic review checklists, root cause analysis	All to the left, plus formal review roles and procedures for fixes, change control. Extensive review checklists, root cause analysis. Continuous review process improvement	
Your rating						

2.20.3 Execution Testing and Tools

	Very Low	Low	Nominal	High	Very High	Don't know
Rating Scale	Ad-hoc testing and debugging. Basic text-based debugger	Basic unit test, integration test, system test process Basic test data management, problem tracking support	In Well-defined test sequence tailored to organization (acceptance / alpha / beta / flight / etc.) test. Basic test coverage tools, test support system. Basic test process management	More advanced test tools, test data preparation, basic test oracle support, distributed monitoring and analysis, assertion checking Metrics-based test process management.	Highly advanced tools for test oracles, distributed monitoring and analysis, assertion checking Integration of automated analysis and test tools Model-based test process management	
Your rating						

2.C Distribution of Effort and Schedule By Stage

This subsection has additional metrics that are required to calibrate the distribution of effort and schedule by stage. Please fill this out if the necessary information is available.

2.21 Total Effort (Person Months). Divide the total effort required for the project into effort (in Person Months) required for each of the following three stages: Inception, Elaboration and Construction.

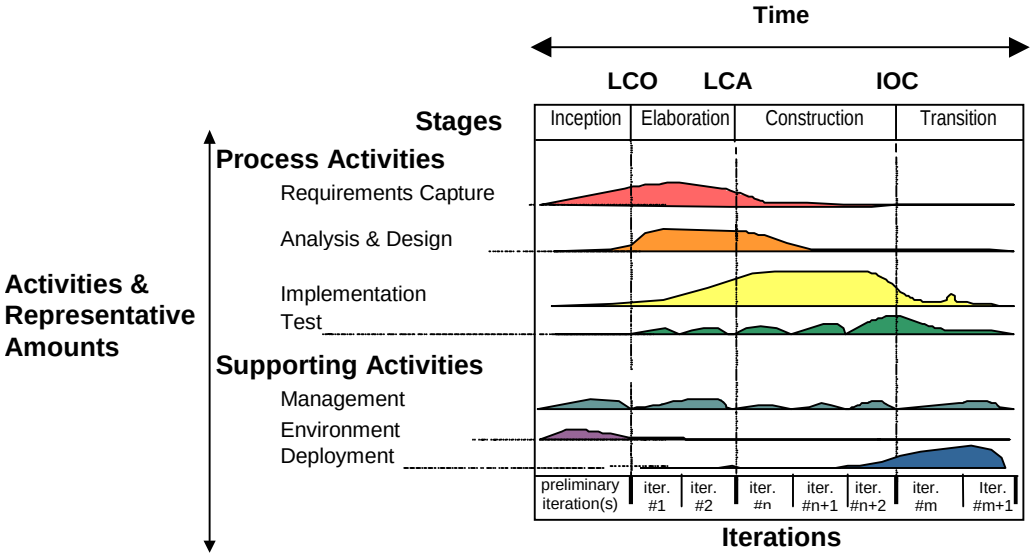
	LCO	LCA	IOC
	Inception	**Elaboration**	**Construction**
Effort Distribution			

2.22 Schedule Months. Divide the total time for development (schedule) required for the project into schedule (in Calendar Months) required for each of the following three stages: Inception, Elaboration and Construction.

	LCO	LCA	IOC
	Inception	**Elaboration**	**Construction**
Schedule Distribution			

Appendix A

The lifecycle anchoring concepts are discussed by Boehm[6]. The anchor points are defined as Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operational Capability (IOC). An enhanced version of an illustration from Rational Corporation[7] showing the phases around the anchor points is shown below.



The correspondence between phases, COCOMOII's submodels and the life cycle anchor points is shown in the following table for the MBASE spiral lifecycle model. The table also has indications of the relative amounts of the different activities.

COCOMO II Submodel Usage	**Early Design**	**Post-Architecture**		**Maintenance**
	LCO	**LCA**	**IOC**	
Activities \ Phase	**Inception**	**Elaboration**	**Construction**	**Transition**
Requirements Capture	Some usually	Most, peaks here	Minor	None
Analysis & Design	A little	Majority, mostly constant effort	Some	Some, for repair during ODT&E
Implementation	Practically none	Some, usually for risk reduction	Bulk; mostly constant effort	Some, for repair during ODT&E
Test	None	Some, for prototypes	Most for unit, integration and qualification test.	Some, for repaired code.

COCOMOII's effort and schedule estimates are focused on Elaboration and Construction (the phases between LCO and IOC. Inception corresponds to the COCOMO's "Requirements" activity in a waterfall process model. COCOMO's effort for the "Requirements" activity is an additional, fixed percentage of the effort calculated by COCOMO for the development activities. The table also indicates the areas in which the COCOMO II Early Design and Post-Architecture submodels are normally used.

[6] Barry W. Boehm, "Anchoring the Software Process," *IEEE Software,* 13, 4, July 1996, pp. 73-82

[7] Rational Corp., "Rational Objectory Process 4.1 – Your UML Process", available at http://www.rational.com/support/techpapers/toratobjprcs/.