

*CUSTOMER*

*DSDC*

***PARTNERS***  
*in Quality Solutions*

**DSDC**  
DLA'S Central Design Activity

*INDUSTRY*

*Requirements*

*Engineering*

Presented by: DSDC

For more info, send requests to: [sepg@dsdc.dla.mil](mailto:sepg@dsdc.dla.mil)

# Description and Objectives

**Description:** This course provides an overview of the requirements activities (identification, analysis and management) necessary to produce software-intensive systems that meet the need of the end-user.

## Objectives:

1. Identify the requirements activities that occur throughout the systems development life cycle
2. Identify characteristics and types of good system/software requirements and ways to document them
3. Discuss the effects of requirements on project cost and schedule

*CUSTOMER*

*DSDC*

***PARTNERS***  
*in Quality Solutions*

# ***Requirements Activities***

*INDUSTRY*

- ✓ **Systems Requirements Analysis**
- ✓ **Software Requirements Analysis**
- ✓ **Requirements Baselineing**
- ✓ **Requirements Management**

# Attributes of a Well-Written Requirements Document

**IF** the perfect document exists, it would be:

- ✓ Correct
- ✓ Unambiguous
- ✓ Complete
- ✓ Verifiable (Testable)
- ✓ Consistent
- ✓ Understandable by non-computer specialists
- ✓ Modifiable
- ✓ Traceable
- ✓ Annotated

# **Correct** **IF and Only IF . . .**

✓ **Every requirement stated within it represents something required of the system to be built**

# Unambiguous IF and Only IF . . .

- ✓ Every requirement has only one interpretation
  - ✓ Terms with multiple meaning must appear in a glossary
  - ✓ Potentially ambiguous words should be avoided or clarified
  - ✓ Using formal notation reduces ambiguity but may be difficult to understand
  - ✓ Diagramming may reveal ambiguity

# Complete IF and Only IF . . .

- ✓ Possesses the following four qualities
  - ✓ Contains everything that the software is supposed to do
  - ✓ Definitions of the responses of the software to all realizable classes of input data in all realizable classes of situations are included
  - ✓ All pages are numbered; all figures and tables are numbered, named and referenced; all terms and units of measure are provided; all referenced materials are present
  - ✓ No sections are marked “To Be Determined” (TBD)

CUSTOMER

DSDC

**PARTNERS**  
in Quality Solutions

INDUSTRY

# Verifiable IF and Only IF . . .

✓ Every requirement is verifiable; i.e., there exists some finite cost effective process with which a person or machine can check that the actual as-built software meets the requirement (testable)

DSDC

CUSTOMER

**PARTNERS**  
*in Quality Solutions*

INDUSTRY

# Consistent IF and Only IF . . .

✓ No subsets of individual requirements stated within it  
conflict

# **Understandable IF and Only IF . . .**

- ✓ Characteristics for Understanding the requirements document are present:
  - ✓ Logically organized
  - ✓ Clear, precise wording
  - ✓ No extraneous phrases
  - ✓ Terms and abbreviations defined in glossary
  - ✓ Use of graphics clarify rather than confuse
  - ✓ Redundancy may exist to improve readability

DSDC

CUSTOMER

**PARTNERS**  
*in Quality Solutions*

INDUSTRY

# **Traceable IF and Only IF . . .**

- ✓ The origin of each requirement is clear
- ✓ It facilitates the referencing of each requirement in future development or enhancement document

DSDC

CUSTOMER

**PARTNERS**  
*in Quality Solutions*

INDUSTRY

# **Annotated IF and Only IF . . .**

- ✓ The document provides guidance to the development organization
  - ✓ Relative necessity (priority)
  - ✓ Relative stability
  - ✓ Notes section

# Types of Requirements

- ✓ System capability (functional) requirements
- ✓ External interface requirements
- ✓ Internal interface requirements
- ✓ System internal data requirements
- ✓ Adaptation requirements
- ✓ Safety requirements
- ✓ Security and privacy requirements
- ✓ System environment requirements
- ✓ Computer resource requirements

# Types of Requirements Cont'd

- ✓ System quality factors
- ✓ Design and construction constraints
- ✓ Personnel-related requirements
- ✓ Training-related requirements
- ✓ Logistics-related requirements
- ✓ Other requirements
- ✓ Packaging requirements
- ✓ Precedence/criticality requirements

# Requirements

*DSDC*

*CUSTOMER*

***PARTNERS***  
*in Quality Solutions*

*INDUSTRY*

**Why are better requirements needed?**

**Better requirements lead to and result in better estimates**

# Estimation Goals

DSDC

CUSTOMER

**PARTNERS**  
*in Quality Solutions*

INDUSTRY

- ✓ Consistency
- ✓ Documentation and History of Estimates
- ✓ Continuing effort to bring estimates closer to actuals

DSDC

CUSTOMER

**PARTNERS**  
*in Quality Solutions*

INDUSTRY

# When Does DSDC Give Estimates?

- ✓ Preliminary Cost Estimate (Quick Look)
- ✓ Proposal Estimate
- ✓ Software Development Plan Estimate
- ✓ Rebaseline Estimate

CUSTOMER

DSDC

**PARTNERS**  
in Quality Solutions

# How Much Will It Cost?

INDUSTRY

## HOW BIG IS IT?

- ✓ Lines of Code
- ✓ Function Points
- ✓ Pages of Documentation



## HOW MUCH EFFORT WILL IT TAKE?

- ✓ FP/workhour
- ✓ COCOMO



DSDC

CUSTOMER

**PARTNERS**  
in Quality Solutions

# Basic Estimating Methods

INDUSTRY

## ✓ Professional Judgment

“My experience and training tell me...”

## ✓ Analogy

“This job is similar to another one...”

## ✓ Automated Tools

“The model results show that...”

*CUSTOMER*

*DSDC*

***PARTNERS***  
*in Quality Solutions*

# ***Estimation Techniques We Have Used***

*INDUSTRY*

- ✓ **System Experts**
- ✓ **Task breakdown and effort estimate**
- ✓ **Tools (Swan)**
- ✓ **Schedule Driven**
- ✓ **The Hat Method**
- ✓ **“And then Add xx%”**

DSDC

CUSTOMER

**PARTNERS**  
in Quality Solutions

# DSDC Estimation Guidelines

INDUSTRY

- ✓ Estimate using at least two techniques
  - ✓ Validate models/productivity rates against knowledge, experience, common sense
- ✓ Get estimates from independent sources
- ✓ Adjust for the people doing the work
- ✓ Do it as a team, and document your assumptions and rationale

CUSTOMER

DSDC

**PARTNERS**  
in Quality Solutions

# Closer to the Mark

INDUSTRY

- ✓ Requirement Definition
- ✓ Requirements Control
- ✓ Process Models
- ✓ Mil-Std-498
- ✓ FP Analysis
- ✓ COCOMO



DSDC

CUSTOMER

**PARTNERS**  
in Quality Solutions

INDUSTRY

# COCOMO

## (COnstructive COst MOdel)

- ✓ B.W. Boehm, “*Software Engineering Economics*”, 1981
- ✓ An accepted Industry Standard
- ✓ Based upon extensive research of previous information system projects
- ✓ Based upon software size (new and adapted) and 15 environmental cost drivers
- ✓ An Exponential Model - “Twice as big means more than twice as hard”

# What is COCOMO?

*The COCOMO model is really a rigorous extension of the “Analogy” method of estimation*

- ✓ The model allows comparison of the current project to an extensive calibration database of completed projects
- ✓ It is as if we had completed these projects ourselves and had metrics for them
- ✓ It is repeatable - thus, it fosters consistency
- ✓ It is recognized world-wide as an industry standard mechanism for software estimation

# The Forms of COCOMO

**BASIC:** Considers only size of the software system

**INTERMEDIATE:** Based upon software size (measured in Lines of Code [LOC]) plus 15 general “cost drivers” that relate to the Product, Project, Personnel, and Environment complexity

**DETAILED:** Intermediate COCOMO taken to the software component level. Complex and labor intensive.

*DSDC has chosen Intermediate COCOMO as a good compromise between accuracy and effort and time required to obtain estimations*

DSDC

CUSTOMER

**PARTNERS**  
in Quality Solutions

INDUSTRY

# Intermediate COCOMO

## Covers:

### DEVELOPMENT PHASES

- ✓ Plans/Requirements Mgmt
- ✓ Product Design
- ✓ Programming
- ✓ Developmental Integration/Test

### Does not cover:

- ✓ Requirements Development
- ✓ OT & E

### ACTIVITIES

- ✓ Requirements Analysis
- ✓ Product Design
- ✓ Programming
- ✓ Developmental Test Planning
- ✓ Verification/Validation
- ✓ Project Office
- ✓ CM/QA
- ✓ Manuals/Documentation

# The Software Estimation Paradox

*All software estimation models exploit the correlation between software “size” and the resultant effort/schedule required to produce it.*

- ✓ “Size” can be measured in just about any units, so long as one is consistent. (# listing pages, weight of listing, etc)
- ✓ The common size unit is Source Lines of Code (or equivalently, Delivered Source Instructions)
- ✓ Arguments rage over lines of code, but as long as one is consistent, they are pretty much academic

*The Paradox: How do we estimate software size when it isn't built yet and all we have are functional requirements?*

# Enter Function Point Analysis

*Function Point Analysis (FPA) attempts to solve this problem by modeling ultimate software size as a function of the characteristics and attributes of the requirements*

- ✓ Counts of inputs, outputs, files, interfaces, and inquiries, ranked by complexity (simple, average, complex)
- ✓ Rankings on 14 “Influence Factors” which describe the overall software product
- ✓ A pre-calibrated mapping of function points to Lines of Code (LOC) for a variety of programming languages in common usage (and some not-so-common ones!)

# Function Point Analysis

## Definition:

An objective, quantitative measurement of the size and complexity of a system based upon the user point of view.

Number of:  
Inputs    Outputs  
Inquiries Data Files  
Interfaces

Adjusted  
By

- Data Communications
- Distributed Data Processing
- Performance
- Heavily Used Configuration
- Transaction Rate
- Online Data Entry
- End-User Efficiency
- Online Update
- Complex Processing
- Reusability
- Installation Ease
- Operational Ease
- Multiple Sites
- Facilitate Change

Equals

Total  
Function  
Points

**Total Function Points**  
**X Language Conversion Factor**  


---

**Estimated Lines of Code**

DSDC

CUSTOMER

**PARTNERS**  
in Quality Solutions

INDUSTRY

# COCOMO - Cost Drivers

Line of Code Estimate

Adjusted  
By



- Required Software Reliability
- Data Base Size
- Product Complexity
- Execution Time Constraint
- Main Storage Constraint
- Virtual Machine Volatility
- Computer Turnaround Time
- Analyst Capability
- Applications Experience
- Programmer Capability
- Virtual Machine Experience
- Programming Language Experience
- Modern Programming Practices
- Use of Software Tools
- Required Development Schedule

Producing



Person Hours &  
Duration

# “What if I need it sooner?”

## *Two well established software engineering principles:*

- If you compress a software development schedule, the work will require more effort (staff hours)
- If you have “all the time in the world,” you will take it

- ✓ Compression of the schedule leads to higher risks
- ✓ Compression to less than about 70% of nominal is generally unrealistic.
- ✓ One effective mechanism for schedule compression is to compress only the critical delivery tasks.

*COCOMO does not lower the intrinsic risk of schedule compression, BUT it does make it quantitatively visible*

# COCOMO Output

## COCOMO Produces:

- ✓ An estimate of Effort (staff-months) for the project
- ✓ An estimate of the nominal Schedule (months)
- ✓ A distribution by development phase and activity
- ✓ An estimated average staff size, based on productivity

*COCOMO assumes “commercial documentation,” and certain other characteristics about the software engineering methodologies. The raw results of the model can be scaled to accommodate variations from these assumptions*

# How Accurate is COCOMO for DSDC?

*DSDC provided data for a number of completed projects. These data were used to create estimates from various models.*

- ✓ Intermediate COCOMO estimates effort for Organic Mode projects with very low errors (average < 1%)
- ✓ Schedule estimates tends to be a little inflated, which is highly typical
- ✓ Large project data provided an extremely good fit, however number of projects was insufficient for statistical significance