Lecture X:

Software Metrics

General Views

ENGINEER

Nutritional Facts

Serving Size: 1 Highly Efficient Engineer*

Amount Per Serving: Multiple Revisions

	%Daily Value**
Critical Thinking	1000%
Unrivaled Skill	500%
Determination	100%
Pride	300%
Wrong Answers	0%
Caffeine	110%

*Not a Significant Source of Correct Grammar.
**Percent Daily Values are Based on Your Unique Diet

Topics to be Covered

- Process and Project Metrics
- Software Measurement
- Metrics for Software Quality
- COCOMO

Measurement

- Provides a mechanism for objective evaluation
- Assists in
  - Estimation
  - Quality control
  - Productivity assessment
  - Project Control
  - Tactical decision-making
- Acts as management tool
- The Major Software Project Dimensions involve:
  - **People**: Dev productivity, Team numbers
  - **Process**: Dev basics, risk management, quality assurance, lifecycle planning, customer orientation
  - **Product**: Most tangible dimension
  - **Technology**

Metrics in the Process and Project Domains

- Process metrics are collected across all projects and over long periods of time.

- Project metrics enable a software project manager to
  - Assess the status of an ongoing project
  - Track potential risks
  - Uncover problem areas before they go “critical”
  - Adjust work flow or tasks
  - Evaluate the project team’s ability to control quality of software work products

Process Metrics and Software Process Improvement cont'd

- The efficacy of a software process is measured indirectly, based on outcomes

- Probable outcomes are
  - Measures of errors uncovered before release of the software
  - Defects delivered to and reported by end-users
  - Work products delivered
  - Human effort expended
  - Calendar time expended
  - Schedule conformance

Project Metrics

- Used during estimation

- Used to monitor and control progress

- The intent is twofold
  - ❑ Minimize the development schedule
  - ❑ Assess product quality on an ongoing basis

- Leads to a reduction in overall project cost

Software Measurement

- Software measurement can be categorized in two ways:
  - ***Direct measures*** of the software process (e.g., cost and effort applied) and product (e.g., lines of code (LOC) produced)
  - ***Indirect measures*** of the product (e.g., functionality, quality, complexity)
- Requires normalization of both size and function-oriented metrics.

Size-Oriented Metrics

- Lines of Code (LOC) can be chosen as the normalization value.

- Example of simple size-oriented metrics
  - Errors per KLOC (thousand lines of code)
  - Defects per KLOC
  - Rate per KLOC
  - Pages of documentation per KLOC

Size-Oriented Metrics cont'd

- Controversy regarding use of LOC as a key measure
  - According to the proponents
    - LOC is an "*artifact*" of all software development projects
    - Many existing software estimation models use LOC or KLOC as a *key input*
  - According to the opponents
    - LOC measures are programming *language dependent*
    - They penalize *well-designed* but *shorter* programs
    - Cannot easily *accommodate* nonprocedural languages
    - *Difficult* to *predict* during estimation

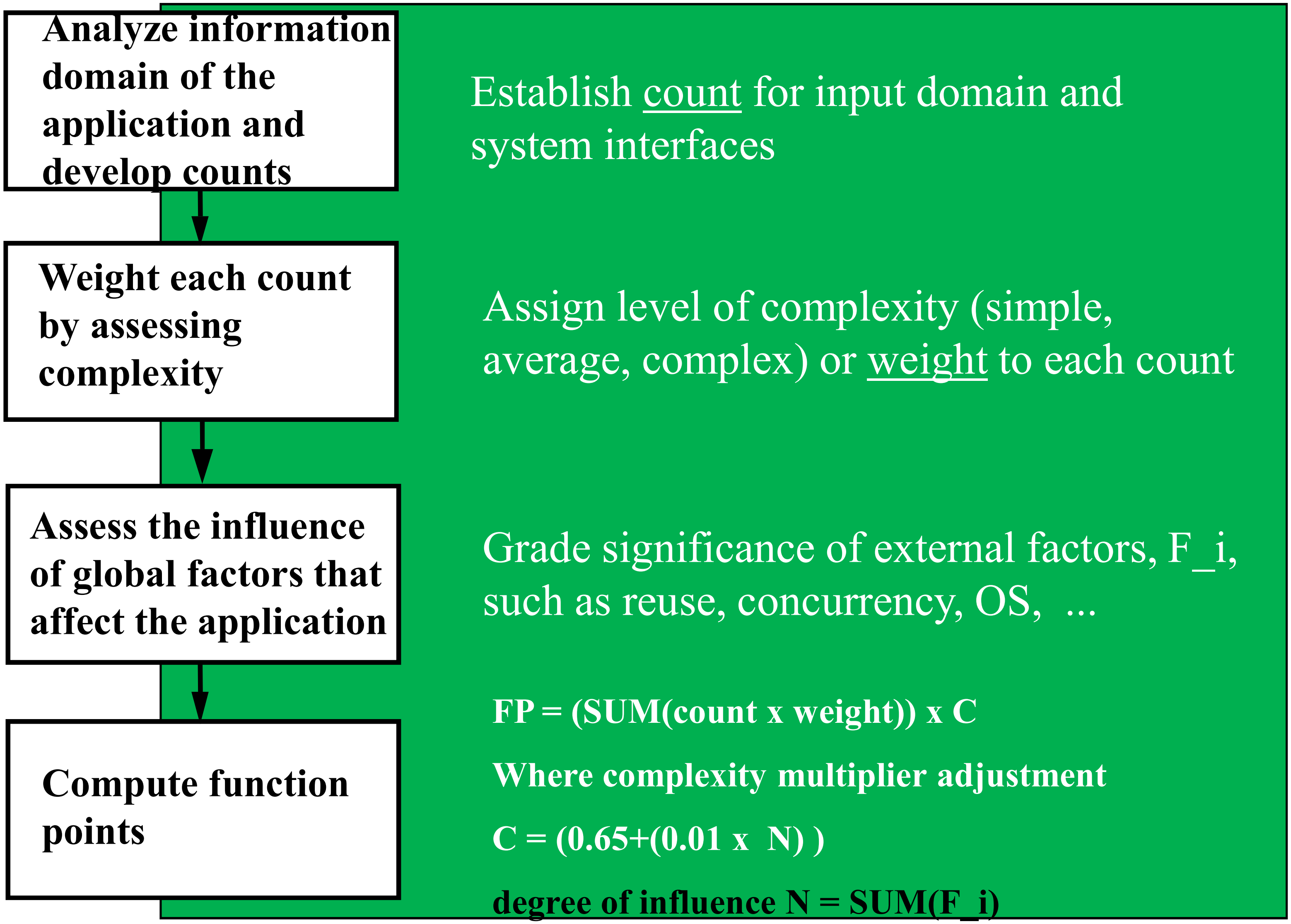
Function-Oriented Metrics

- The widely used function-oriented metric is the function point (FP).

- Computation of the FP is based on characteristics of the software's information domain and complexity.

- Controversy regarding use of FP as a key measure
  - According to the proponents
    - It is programming language *independent*
    - Can be *predicted* before coding is started
  - According to the opponents
    - Based on *subjective* rather than *objective* data
    - Has *no direct physical meaning* – it's just a number

Computing Function Points



Analyze information domain of the application and develop counts

Establish count for input domain and system interfaces

Weight each count by assessing complexity

Assign level of complexity (simple, average, complex) or weight to each count

Assess the influence of global factors that affect the application

Grade significance of external factors, F_i, such as reuse, concurrency, OS, ...

Compute function points

FP = (SUM(count x weight)) x C

Where complexity multiplier adjustment

C = (0.65+(0.01 x N))

degree of influence N = SUM(F_i)

Analyzing the Information Domain

			weighting factor				
measurement parameter	count		simple	avg.	complex		
number of user inputs	□	X	3	4	6	=	□
number of user outputs	□	X	4	5	7	=	□
number of user inquiries	□	X	3	4	6	=	□
number of files	□	X	7	10	15	=	□
number of ext.interfaces	□	X	5	7	10	=	□
count-total →							□
complexity multiplier							□
function points →							□

$$\sum_{Inputs} Wi + \sum_{Output} Wo + \sum_{Inquiry} Win + \sum_{InternalFiles} Wif + \sum_{ExternalInterfaces} Wei$$

Exercise: Function Points

- Compute the function point value for a project with the following information domain characteristics:
  - Number of user inputs: 32
  - Number of user outputs: 60
  - Number of user enquiries: 24
  - Number of files: 8
  - Number of external interfaces: 2
  - Assume that *weights* are *average* and *the degree of influence N is* ***not*** important.
- ***Answer ???***

Exercise: Function Points

- Compute the function point value for a project with the following information domain characteristics:
  - Number of user inputs: 32
  - Number of user outputs: 60
  - Number of user enquiries: 24
  - Number of files: 8
  - Number of external interfaces: 2
  - Assume that *weights* are *average* and *the degree of influence N* *is* ***not*** important.
- Answer:

$$\sum_{Inputs} Wi + \sum_{Output} Wo + \sum_{Inquiry} Win + \sum_{InternalFiles} Wif + \sum_{ExternalInterfaces} Wei$$

=[(32*4)+(60*5)+(24*4)+(8*10)+(2*7)] * 0.65

= 128+300+96+80+14= 618

=618*.65

=401.7

COCOMO

- COCOMO stands for COnstructive COst MOdel
- It is an open system first published by Dr. Barry Bohem
- Works quite well for projects
- Could estimate results within ~20% of the actual values 68% of the time

COCOMO cont'd

- COCOMO has three different models (each one increasing with detail and accuracy):
  - Basic COCOMO
    - used for relatively smaller projects
    - team size is considered to be small
    - Cost drivers depend upon size of the projects .
  - Intermediate COCOMO
    - It is used for medium sized projects.
    - The cost drivers are intermediate to basic and advanced COCOMO.
    - Cost drivers depend upon product reliability, computer, personnel and project attributes.
    - Team size is medium.
  - Advanced COCOMO
    - It is used for large sized projects with relatively large teams.
    - The cost drivers depend upon requirements, analysis, design, testing and maintenance.

Intermediate COCOMO

- Estimates the software development effort by using cost driver variables besides the size variable used in Basic COCOMO.

- Product Attributes
  - RELY: Required Software Reliability. The extent to which the software product must perform its intended functions satisfactorily over a period of time.
  - DATA: Data Base Size. The degree of the total amount of data to be assembled for the data base.
  - CPLX: Software Product Complexity. The level of complexity of the product to be developed.
- Computer Attributes
  - TIME --- Execution Time Constraint. The degree of the execution constraint imposed upon a software product.
  - STOR --- Main Storage Constraint. The degree of main storage constraint imposed upon a software product.
  - VIRT --- Virtual Machine Volatility. The level of the virtual machine underlying the product to be developed.
  - TURN --- Computer Turnaround Time. The level of computer response time experienced by the project team developing the product.

Intermediate COCOMO cont'd

- Personnel Attributes
  - ACAP: Analyst Capability. The level of capability of the analysts working on a software product.
  - AEXP: Applications Experience. The level of applications experience of the project team developing the software product.
  - PCAP: Programmer Capability. The level of capability of the programmers working on the software product.
  - VEXP: Virtual Machine Experience. The level of virtual machine experience of the project team developing the product.
  - LEXP: Programming Language Experience. The level of programming language experience of the project team developing the product.

- Project Attributes
  - MODP: Use of Modern Programming Practices. The degree to which modern programming practices (MPPs) are used in developing software product.
  - TOOL: Use of Software Tools. The degree to which software tools are used in developing the software product.
  - SCED: Schedule Constraint. The level of schedule constraint imposed upon the project team developing the software product.

COCOMO cont'd

- Software projects take three common classes:
  - Organic mode projects
    - "relatively small software teams develop software in a highly familiar, in-house environment" [Bohem]
    - Used for relatively smaller teams.
    - There is a proper interaction among the team members and they coordinate their work.
  - Embedded mode projects
    - operate within tight constraints, product is strongly tied to "complex of hardware, software, regulations, and operational procedures" [Bohem]
    - Team members are highly skilled.
    - Team members are familiar with the system under development.
  - Semi-detached mode projects
    - intermediate stage somewhere between organic and embedded.
    - It lies between organic mode and embedded mode in terms of team size.
    - It consists of experienced and inexperienced staff.
    - Some team members are unfamiliar with the system under development.

COCOMO

- COCOMO uses two equations to calculate effort in man months (MM) and the number of months estimated for project (TDEV)
- MM is based on the number of thousand lines of delivered source instructions (KDSI)
  - ***$MM = a * (KDSI)^b * EAF$***
  - ***$TDEV = c * (MM)^d$***
- EAF is the Effort Adjustment Factor derived from the Cost Drivers. EAF for the basic model is 1.
- The values for a, b, c, and d differ depending on which class mode is being used

Mode	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

COCOMO Cont'd

- A simple example:

  Project is a flight control system (mission critical) with 310,000 DSI in embedded mode.

  Reliability must be very high (RELY=1.40).

  Compute the:

  - ❑ Effort (MM)
  - ❑ Schedule (TDEV)
  - ❑ Average Staffing

COCOMO Cont’d

- A simple example:

  Project is a flight control system (mission critical) with 310,000 DSI in embedded mode.

  Reliability must be very high (RELY=1.40).

- So we can calculate:

- MM (Effort) = 3.6*(310)$^{1.20}$ *1.4 = 4921 Man Months
- TDEV (Schedule) = 2.5*(4921)$^{0.32}$ = 38 months
- Average Staffing = 4921MM/38 months = 130 Persons

COCOMO Conclusions

- COCOMO is the most popular software cost estimation method.
- Easy to do, small estimates can be done by hand
- Many different commercial version based on COCOMO are available – they supply support and more data, but at a price