Lecture 8

SOFTWARE PROCESS MANAGEMENT

TOPIC: MEASUREMENT METRICS (FUNCTION POINT)

Size-Oriented Metrics (1)

- 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
  - $ per KLOC
  - Pages of documentation per KLOC

Size-Oriented Metrics (2)

Project	LOC	Effort	$(000)	Pp. doc.	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
Beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Size-Oriented Metrics (3)

- Controversy regarding use of LOC as a key measure
  - According to the proponents
    - LOC is an "artifact" of all s/w development projects
    - Many existing s/w 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

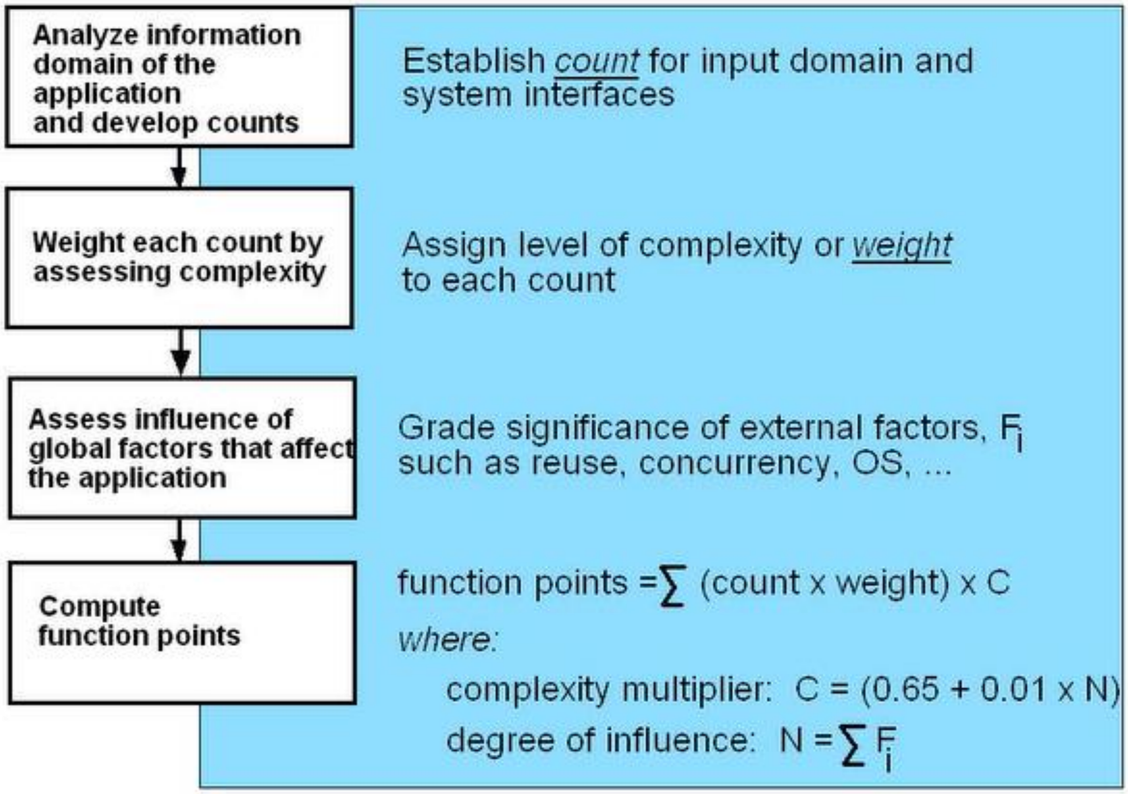
Function-Oriented Metrics

- The most 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

Information Domain

- **Number of external inputs** – from user or another application
- **Number of external outputs**
- **Number of external inquiries** – request from user that generates an on-line output
- **Number of internal logical files** (maintained by system)
- **Number of external interface files** (provides data but not maintained by system)

Computing Function Points



Analyzing the Information Domain

measurement parameter	count		weighting factor simple	avg.	complex		
number of user inputs	3	X	3	4	6	=	12
number of user outputs	5	X	4	5	7	=	25
number of user inquiries	2	X	3	4	6	=	8
number of files	4	X	7	10	15	=	40
number of ext.interfaces	1	X	5	7	10	=	7
count-total							92

complexity multiplier $[0.65 + 0.01 \times \sum(F_i)]$

function points **count-total** $\times [0.65 + 0.01 \times \sum(F_i)]$

Taking Complexity into Account

Factors(F_i) are rated on a scale of 0 (not important) to 5 (essential)

The following are some examples of these factors:

- Is high performance critical?
- Is the internal processing complex?
- Is the system to be used in multiple sites and/or by multiple organizations?
- Is the code designed to be reusable?
- Is the processing to be distributed?
- and so forth . . .

Computing Function Points

measurement parameter	count		weighting factor				
			simple	avg.	complex		
number of user inputs	3	X	3	4	6	=	12
number of user outputs	5	X	4	5	7	=	25
number of user inquiries	2	X	3	4	6	=	8
number of files	4	X	7	10	15	=	40
number of ext.interfaces	1	X	5	7	10	=	7
count-total							92
complexity multiplier $[0.65 + 0.01 \times \sum(F_i)]$							1.07
function points count-total $\times [0.65 + 0.01 \times \sum(F_i)]$							98.44

Uses of Function Points(FP)

But how long will the project take and how much will it cost?

- If programmers in an organization produce average 16 function points per month. Thus . . .

  98.44 FP divided by 16 = 6 man-months

- If the average programmer is paid $5,200 per month (including benefits), then the [labor] cost of the project will be . . .

  6 man-months X $5,200 = $31,200

Pros & Cons of FP

- 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

Reconciling LOC and FP Metrics

Programming Language	LOC per Function point			
	Avg.	Median	Low	High
Access	35	38	15	47
Ada	154	—	104	205
APS	86	83	20	184
ASP 69	62	—	32	127
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
Clipper	38	39	27	70
COBOL	77	77	14	400
Cool:Gen/IEF	38	31	10	180
Culprit	51	—	—	—
DBase IV	52	—	—	—
Easytrieve+	33	34	25	41
Excel47	46	—	31	63
Focus	43	42	32	56
FORTRAN	—	—	—	—
FoxPro	32	35	25	35
Ideal	66	52	34	203
IEF/Cool:Gen	38	31	10	180
Informix	42	31	24	57
Java	63	53	77	

Typical Size-Oriented Metrics

- errors per KLOC (thousand lines of code)
- defects per KLOC
- $ per LOC
- pages of documentation per KLOC
- errors per person-month
- errors per review hour
- LOC per person-month
- $ per page of documentation

Typical Function-Oriented Metrics

- errors per FP (thousand lines of code)
- defects per FP
- $ per FP
- pages of documentation per FP
- FP per person-month

Comparing LOC and FP

Programming Language	LOC per Function point avg.	median	low	high
Ada	154	-	104	205
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
COBOL	77	77	14	400
Java	63	53	77	-
JavaScript	58	63	42	75
Perl	60	-	-	-
PL/1	78	67	22	263
Powerbuilder	32	31	11	105
SAS	40	41	33	49
Smalltalk	26	19	10	55
SQL	40	37	7	110
Visual Basic	47	42	16	158

Representative values developed by QSM

Why Opt for FP?

- Programming language independent
- Used readily countable characteristics that are determined early in the software process
- Does not "penalize" inventive (short) implementations that use fewer LOC that other more clumsy versions
- Makes it easier to measure the impact of reusable components

Object-Oriented Metrics

- Number of scenario scripts (use-cases)
- Number of support classes (required to implement the system but are not immediately related to the problem domain)
- Average number of support classes per key class (analysis class)
- Number of subsystems (an aggregation of classes that support a function that is visible to the end-user of a system)

WebApp Project Metrics

- Number of static Web pages (the end-user has no control over the content displayed on the page)
- Number of dynamic Web pages (end-user actions result in customized content displayed on the page)
- Number of internal page links (internal page links are pointers that provide a hyperlink to some other Web page within the WebApp)
- Number of persistent data objects
- Number of external systems interfaced
- Number of static content objects
- Number of dynamic content objects
- Number of executable functions

Measuring Quality

- Correctness — the degree to which a program operates according to specification
- Maintainability—the degree to which a program is amenable to change
- Integrity—the degree to which a program is impervious to outside attack
- Usability—the degree to which a program is easy to use

Defect Removal Efficiency

$$\text{DRE} = E/(E + D)$$

where:

E is the number of errors found before delivery of the software to the end-user

D is the number of defects found after delivery.

Metrics for Small Organizations

- time (hours or days) elapsed from the time a request is made until evaluation is complete, t_{queue}.
- effort (person-hours) to perform the evaluation, W_{eval}.
- time (hours or days) elapsed from completion of evaluation to assignment of change order to personnel, t_{eval}.
- effort (person-hours) required to make the change, W_{change}.
- time required (hours or days) to make the change, t_{change}.
- errors uncovered during work to make change, E_{change}.
- defects uncovered after change is released to the customer base, D_{change}.

Establishing a Metrics Program

- Identify your business goals.
- Identify what you want to know or learn.
- Identify your subgoals.
- Identify the entities and attributes related to your subgoals.
- Formalize your measurement goals.
- Identify quantifiable questions and the related indicators that you will use to help you achieve your measurement goals.
- Identify the data elements that you will collect to construct the indicators that help answer your questions.
- Define the measures to be used, and make these definitions operational.
- Identify the actions that you will take to implement the measures.
- Prepare a plan for implementing the measures.

What Attributes Can We Measure?

- **We want attributes that relate to our goals**
  - time, resources, performance, quality etc.
- **The following type of matrix can help:**

Attributes \ What	Process	Product	Project
Time	What Is our Cycle Time?	How Fast can we Manufacture?	Are We On Schedule?
Resources	What is our Productivity?	What will it Cost?	Expenses vs. Budget?
Performance	Does it Work?	Meets Perf. Goals?	Meets Mgt. Goals?
Quality	In-process Defects?	Post-release Defects?	Customer Satisfaction?

Examples of Entities and Attributes

Entity	*Attribute*
Software Design	Defects discovered in design reviews
Software Design Specification	Number of pages
Software Code	Number of lines of code, number of operations
Software Development Team	Team size, average team experience

Web Engineering Project Metrics (2)

- Let,
  - N_{sp} = number of static Web pages
  - N_{dp} = number of dynamic Web pages
- Then,
  - Customization index, $C = N_{dp}/(N_{dp} + N_{sp})$
- The value of C ranges from 0 to 1

Metrics for Software Quality

- Goals of s/w engineering
  - Produce high-quality systems
  - Meet deadlines
  - Satisfy market need
- The primary thrust at the project level is to measure errors and defects

Measuring Quality

- Correctness
  - Defects per KLOC
- Maintainability
  - Mean-time-to-change (MTTC)
- Integrity
  - Threat and security
  - integrity = Σ [1 – (threat $\times$ (1 - security))]
- Usability

Defect Removal Efficiency (DRE)

- Can be used at both the project and process level
- DRE = E / (E + D), [E = Error, D = Defect]
- Or, $DRE_i = E_i / (E_i + E_{i+1})$, [for i^{th} activity]
- Try to achieve DRE_i that approaches 1

Example: Compute the function point, productivity, documentation, cost per function for the following data:

1. Number of user inputs = 24
2. Number of user outputs = 46
3. Number of inquiries = 8
4. Number of files = 4
5. Number of external interfaces = 2
6. Effort = 36.9 p-m
7. Technical documents = 265 pages
8. User documents = 122 pages
9. Cost = $7744/ month

Various processing complexity factors are: 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.

Solution:

Measurement Parameter	**Count**		**Weighing factor**
1. Number of external inputs (EI)	24	*	4 = 96
2. Number of external outputs (EO)	46	*	4 = 184
3. Number of external inquiries (EQ)	8	*	6 = 48
4. Number of internal files (ILF)	4	*	10 = 40
5. Number of external interfaces (EIF) Count-total →	2	*	5 = 10 378

So sum of all f_i ($i \leftarrow 1$ to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

FP = Count-total * [0.65 + 0.01 *$\Sigma(f_i)$]
= 378 * [0.65 + 0.01 * 43]
= 378 * [0.65 + 0.43]
= 378 * 1.08 = 408

$$\text{Productivity} = \frac{\text{FP}}{\text{Effort}} = \frac{408}{36.9} = 11.1$$

Total pages of documentation = technical document + user document
= 265 + 122 = 387pages

Documentation = Pages of documentation/FP
= 387/408 = 0.94

$$\text{Cost per function} = \frac{\text{cost}}{\text{productivity}} = \frac{7744}{11.1} = \$700$$