BINUS UNIVERSITY

People
Innovation
Excellence

Course	: COMP6100/Software Engineering
Effective Period	: Desember 2017

Software Project Management and Software Metrics

Session 27-28

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 30, 31 and 32

People
Innovation
Excellence

Learning Objectives

LO 4 : Analyze the software project management and the proposed potential business project

Contents

Team Coordination and Communication

- ***Formal, impersonal approaches*** include software engineering documents and work products (including source code), technical memos, project milestones, schedules, and project control tools, change requests and related documentation, error tracking reports, and repository data.
- ***Formal, interpersonal procedures*** focus on quality assurance activities applied to software engineering work products. These include status review meetings and design and code inspections.
- ***Informal, interpersonal procedures*** include group meetings for information dissemination and problem solving and "collocation of requirements and development staff."
- ***Electronic communication*** encompasses electronic mail, electronic bulletin boards, and by extension, video-based conferencing systems.
- ***Interpersonal networking*** includes informal discussions with team members and those outside the project who may have experience or insight that can assist team members.

BINUS UNIVERSITY

People
Innovation
Excellence

Problem Decomposition

- Sometimes called ***partitioning*** **or** ***problem elaboration***
- Once scope is defined …
  - It is decomposed into constituent functions
  - It is decomposed into user-visible data objects

  or
  - It is decomposed into a set of problem classes
- Decomposition process continues until all functions or problem classes have been defined

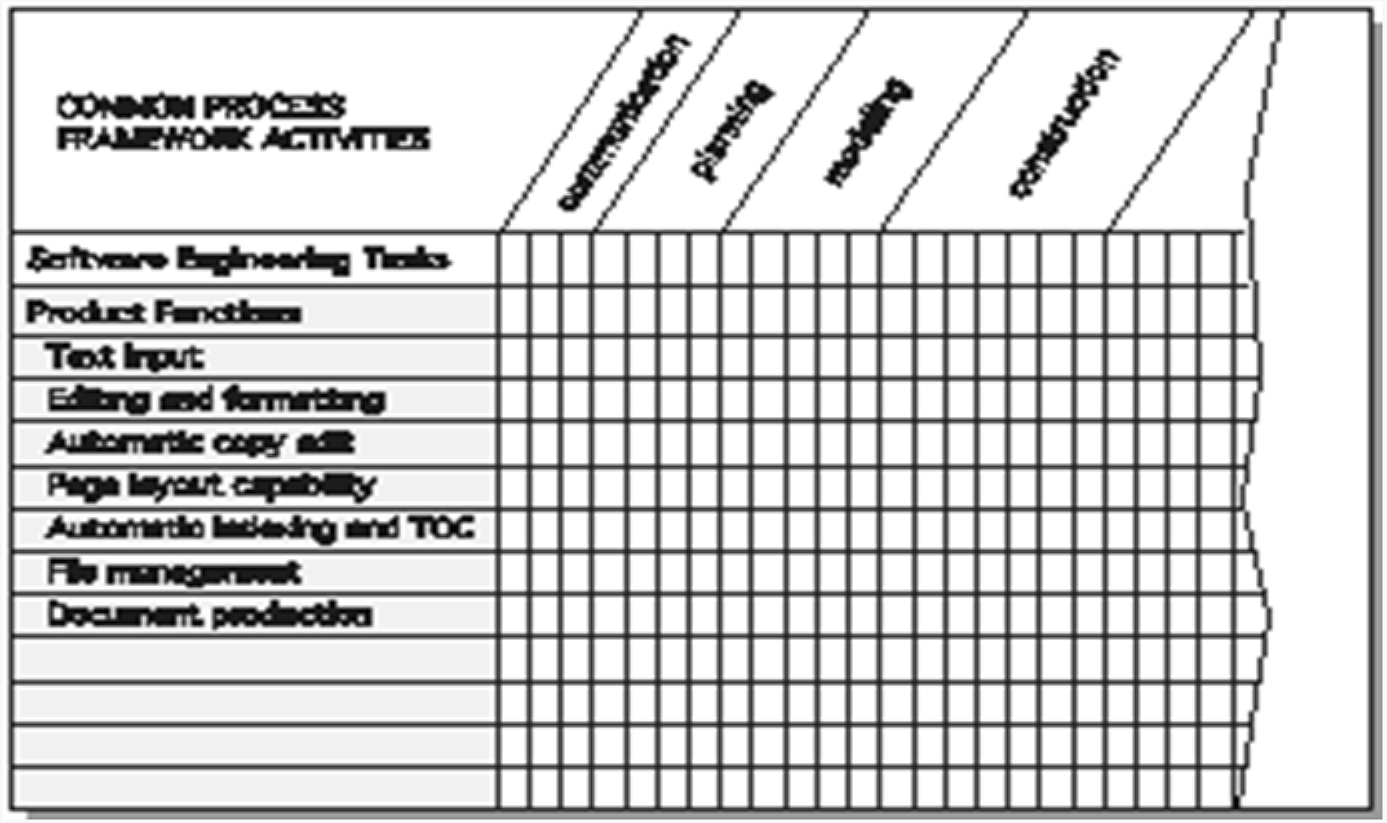
Problem Decomposition

The Process

- Once a process framework has been established
  - Consider project characteristics
  - Determine the degree of rigor required
  - Define a task set for each software engineering activity
    - Task set =
      - Software engineering tasks
      - Work products
      - Quality assurance points
      - Milestones

Problem Decomposition

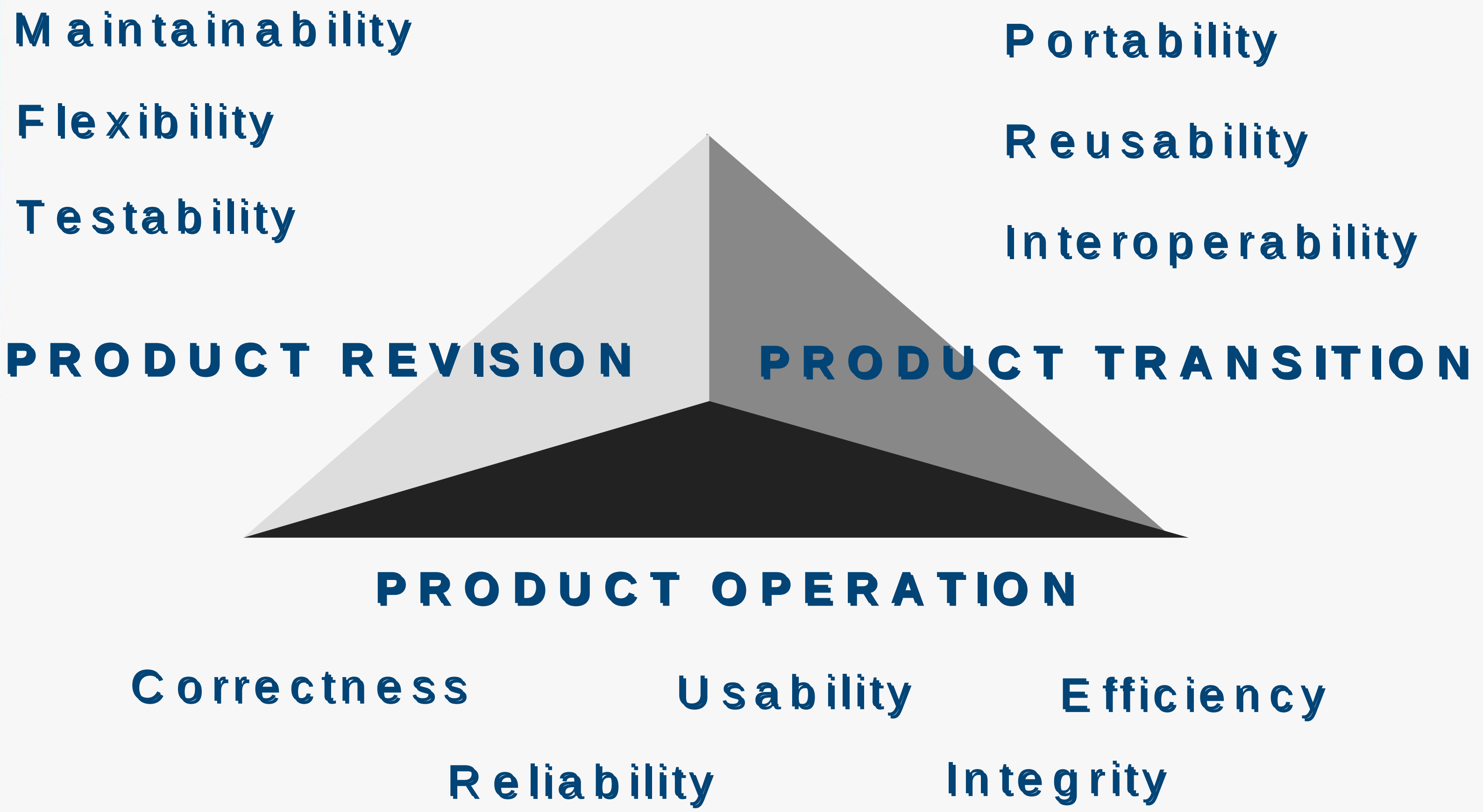
Melding the Problem and the Process



COMMON PROCESS FRAMEWORK ACTIVITIES	communication	planning	modeling	construction	
Software Engineering Tasks					
Product Functions					
Text input					
Editing and formatting					
Automatic copy edit					
Page layout capability					
Automatic indexing and TOC					
File management					
Document production					

A Framework for Product Metrics

McCall's Triangle of Quality



A Framework for Product Metrics

Goal-Oriented Software Measurement

- A *measure* provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process
- The IEEE glossary defines a *metric* as "a quantitative measure of the degree to which a system, component, or process possesses a given attribute."
- An *indicator* is a metric or combination of metrics that provide insight into the software process, a software project, or the product itself

A Framework for Product Metrics

Metrics Attributes

- *Simple and computable.* It should be relatively easy to learn how to derive the metric, and its computation should not demand inordinate effort or time
- *Empirically and intuitively persuasive.* The metric should satisfy the engineer's intuitive notions about the product attribute under consideration
- *Consistent and objective.* The metric should always yield results that are unambiguous.
- *Consistent in its use of units and dimensions.* The mathematical computation of the metric should use measures that do not lead to bizarre combinations of unit.
- *Programming language independent.* Metrics should be based on the analysis model, the design model, or the structure of the program itself.
- *Effective mechanism for quality feedback.* That is, the metric should provide a software engineer with information that can lead to a higher quality end product

People
Innovation
Excellence

Metrics for the Requirements Model

- **Function-based metrics**: use the function point as a normalizing factor or as a measure of the “size” of the specification
- **Specification metrics**: used as an indication of quality by measuring number of requirements by type

Metrics for the Requirements Model

People
Innovation
Excellence

Function-Based Metrics

- The *function point metric* (FP), first proposed by Albrecht [ALB79], can be used effectively as a means for measuring the functionality delivered by a system.
- Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity
- Information domain values are defined in the following manner:
  - number of external inputs (EIs)
  - number of external outputs (EOs)
  - number of external inquiries (EQs)
  - number of internal logical files (ILFs)
  - Number of external interface files (EIFs)

Metrics for the Requirements Model

Function Points

Information Domain Value	Count		Weighting factor simple	average	complex		
External Inputs (EIs)		3	3	4	6	=	
External Outputs (EOs)		3	4	5	7	=	
External Inquiries (EQs)		3	3	4	6	=	
Internal Logical Files (ILFs)		3	7	10	15	=	
External Interface Files (EIFs)		3	5	7	10	=	
Count total							

People
Innovation
Excellence

Metrics for the Design Model

Architectural Design Metrics

- Architectural design metrics
  - Structural complexity = g(fan-out)
  - Data complexity = f(input & output variables, fan-out)
  - System complexity = h(structural & data complexity)
- HK metric: architectural complexity as a function of fan-in and fan-out
- Morphology metrics: a function of the number of modules and the number of interfaces between modules

People
Innovation
Excellence

Metrics for the Design Model

People
Innovation
Excellence

Metrics for OO Design-I

Whitmire [Whi97] describes nine distinct and measurable characteristics of an OO design:

- Size
  - Size is defined in terms of four views: population, volume, length, and functionality
- Complexity
  - How classes of an OO design are interrelated to one another
- Coupling
  - The physical connections between elements of the OO design
- Sufficiency
  - "the degree to which an abstraction possesses the features required of it, or the degree to which a design component possesses features in its abstraction, from the point of view of the current application."

Metrics for the Design Model

Metrics for OO Design-II

- Completeness
  - An indirect implication about the degree to which the abstraction or design component can be reused
- Cohesion
  - The degree to which all operations working together to achieve a single, well-defined purpose
- Primitiveness
  - Applied to both operations and classes, the degree to which an operation is atomic
- Similarity
  - The degree to which two or more classes are similar in terms of their structure, function, behavior, or purpose
- Volatility
  - Measures the likelihood that a change will occur

Metrics for the Design Model

People
Innovation
Excellence

Class-Oriented Metrics

Proposed by Chidamber and Kemerer [Chi94]:

- weighted methods per class
- depth of the inheritance tree
- number of children
- coupling between object classes
- response for a class
- lack of cohesion in methods

Metrics for the Design Model

Class-Oriented Metrics

Proposed by Lorenz and Kidd [Lor94]:

- class size
- number of operations overridden by a subclass
- number of operations added by a subclass
- specialization index

Metrics for the Design Model

Class-Oriented Metrics

The MOOD Metrics Suite [Har98b]:

- Method inheritance factor
- Coupling factor
- Polymorphism factor

People
Innovation
Excellence

Metrics for the Design Model

Operation-Oriented Metrics

Proposed by Lorenz and Kidd [Lor94]:

- **average operation size**
- **operation complexity**
- **average number of parameters per operation**

BINUS UNIVERSITY

People
Innovation
Excellence

Metrics for the Design Model

Component-Level Design Metrics

- **Cohesion metrics:** a function of data objects and the locus of their definition
- **Coupling metrics:** a function of input and output parameters, global variables, and modules called
- **Complexity metrics:** hundreds have been proposed (e.g., cyclomatic complexity)

Metrics for the Design Model

Interface Design Metrics

- **Layout appropriateness:** a function of layout entities, the geographic position and the “cost” of making transitions among entities

Design Metrics for Web and Mobile Apps

- Does the user interface promote usability?
- Are the aesthetics of the WebApp appropriate for the application domain and pleasing to the user?
- Is the content designed in a manner that imparts the most information with the least effort?
- Is navigation efficient and straightforward?
- Has the WebApp architecture been designed to accommodate the special goals and objectives of WebApp users, the structure of content and functionality, and the flow of navigation required to use the system effectively?
- Are components designed in a manner that reduces procedural complexity and enhances the correctness, reliability and performance?

Code Metrics

- **Halstead's Software Science:** a comprehensive collection of metrics all predicated on the number (count and occurrence) of operators and operands within a component or program
  - It should be noted that Halstead's "laws" have generated substantial controversy, and many believe that the underlying theory has flaws. However, experimental verification for selected programming languages has been performed (e.g. [FEL89]).

BINUS UNIVERSITY

Metrics for Testing

- Testing effort can also be estimated using metrics derived from Halstead measures
- Binder [Bin94] suggests a broad array of design metrics that have a direct influence on the "testability" of an OO system.
  - Lack of cohesion in methods (LCOM).
  - Percent public and protected (PAP).
  - Public access to data members (PAD).
  - Number of root classes (NOR).
  - Fan-in (FIN).
  - Number of children (NOC) and depth of the inheritance tree (DIT).

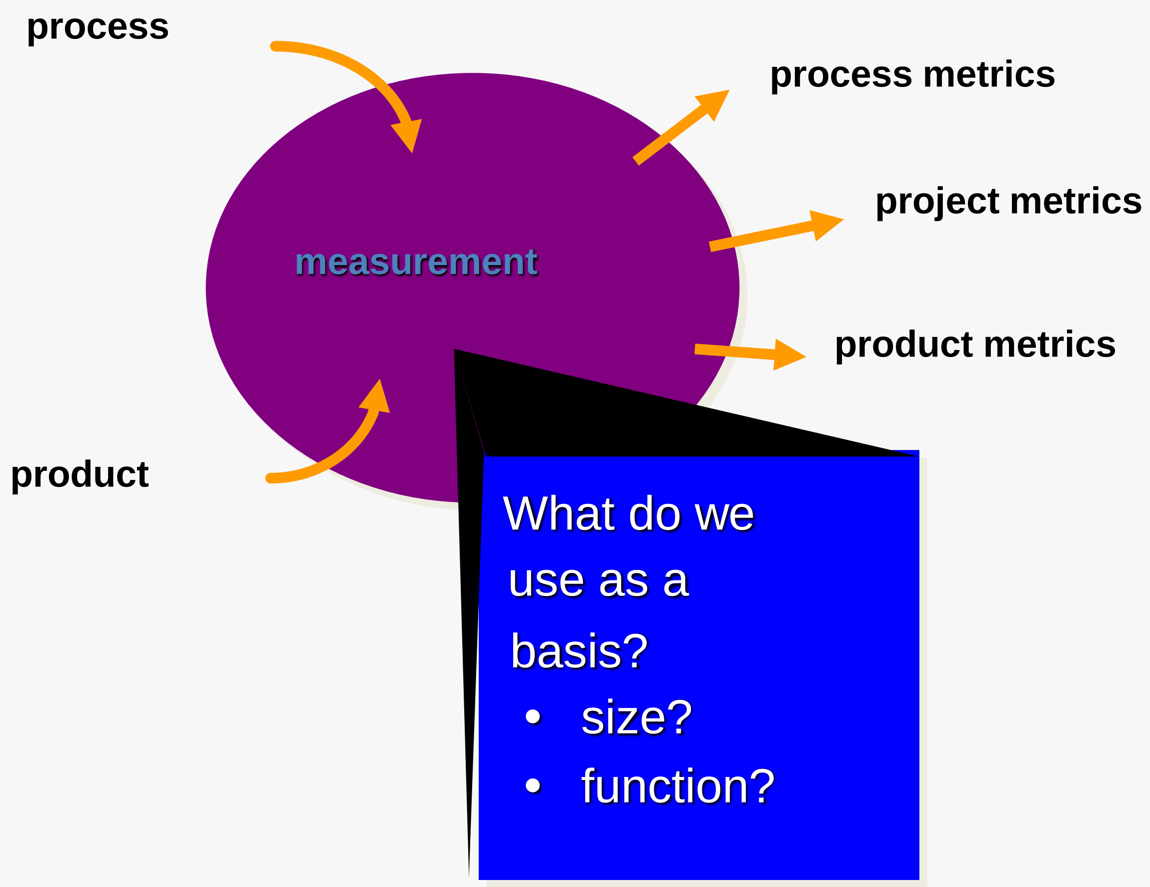
People
Innovation
Excellence

Maintenance Metrics

- IEEE Std. 982.1-1988 [IEE94] suggests a ***software maturity index*** **(SMI)** that provides an indication of the stability of a software product (based on changes that occur for each release of the product). The following information is determined:
  - M_T = the number of modules in the current release
  - F_c = the number of modules in the current release that have been changed
  - F_a = the number of modules in the current release that have been added
  - F_d = the number of modules from the preceding release that were deleted in the current release
- The software maturity index is computed in the following manner:
  - SMI = $[M_T - (F_a + F_c + F_d)]/M_T$
- As SMI approaches 1.0, the product begins to stabilize.

Metrics in the Process and Project Domain

A Good Manager Measures



What do we use as a basis?

- size?
- function?

Metrics in the Process and Project Domain

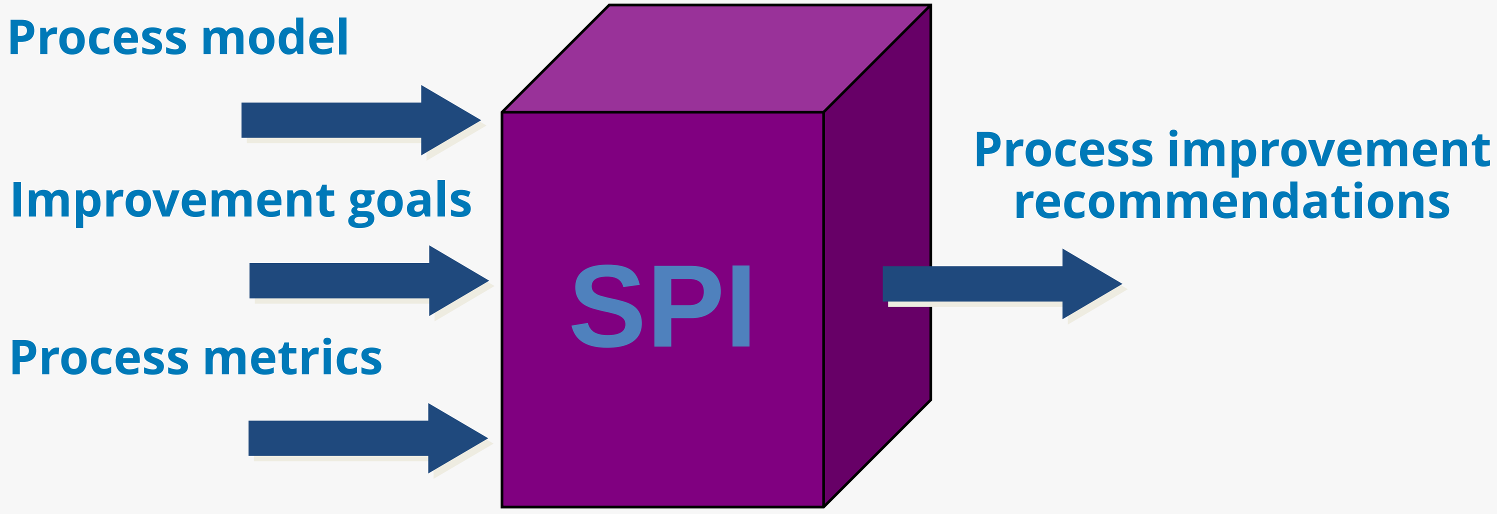
Process Measurement

- We measure the efficacy of a software process indirectly.
  - That is, we derive a set of metrics based on the outcomes that can be derived from the process.
  - Outcomes include
    - measures of errors uncovered before release of the software
    - defects delivered to and reported by end-users
    - work products delivered (productivity)
    - human effort expended
    - calendar time expended
    - schedule conformance
    - other measures.
- We also derive process metrics by measuring the characteristics of specific software engineering tasks.

People
Innovation
Excellence

Metrics in the Process and Project Domain

Software Process Improvement



People
Innovation
Excellence

Metrics in the Process and Project Domain

Process Metrics

- **Quality-related**
  - focus on quality of work products and deliverables
- **Productivity-related**
  - Production of work-products related to effort expended
- **Statistical SQA data**
  - error categorization & analysis
- **Defect removal efficiency**
  - propagation of errors from process activity to activity
- **Reuse data**
  - The number of components produced and their degree of reusability

Metrics in the Process and Project Domain

Project Metrics

- used to minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks
- used to assess product quality on an ongoing basis and, when necessary, modify the technical approach to improve quality.
- every project should measure:
  - *inputs*—measures of the resources (e.g., people, tools) required to do the work.
  - *outputs*—measures of the deliverables or work products created during the software engineering process.
  - *results*—measures that indicate the effectiveness of the deliverables.

BINUS UNIVERSITY

Metrics in the Process and Project Domain

Typical Project Metrics

- Effort/time per software engineering task
- Errors uncovered per review hour
- Scheduled vs. actual milestone dates
- Changes (number) and their characteristics
- Distribution of effort on software engineering tasks

BINUS UNIVERSITY

People
Innovation
Excellence

Software Measurement

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

BINUS UNIVERSITY

Software Measurement

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

People
Innovation
Excellence

Software Measurement

Comparing LOC and FP

Programming	LOC per Function point			
Language	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

Software Measurement

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

BINUS UNIVERSITY

People
Innovation
Excellence

Software Measurement

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)

BINUS UNIVERSITY

People
Innovation
Excellence

Software Measurement

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

Metrics for Software Quality

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

Metrics for Software Quality

Defect Removal Efficiency

$$DRE = E/(E + D)$$

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.**

BINUS UNIVERSITY

Exercise

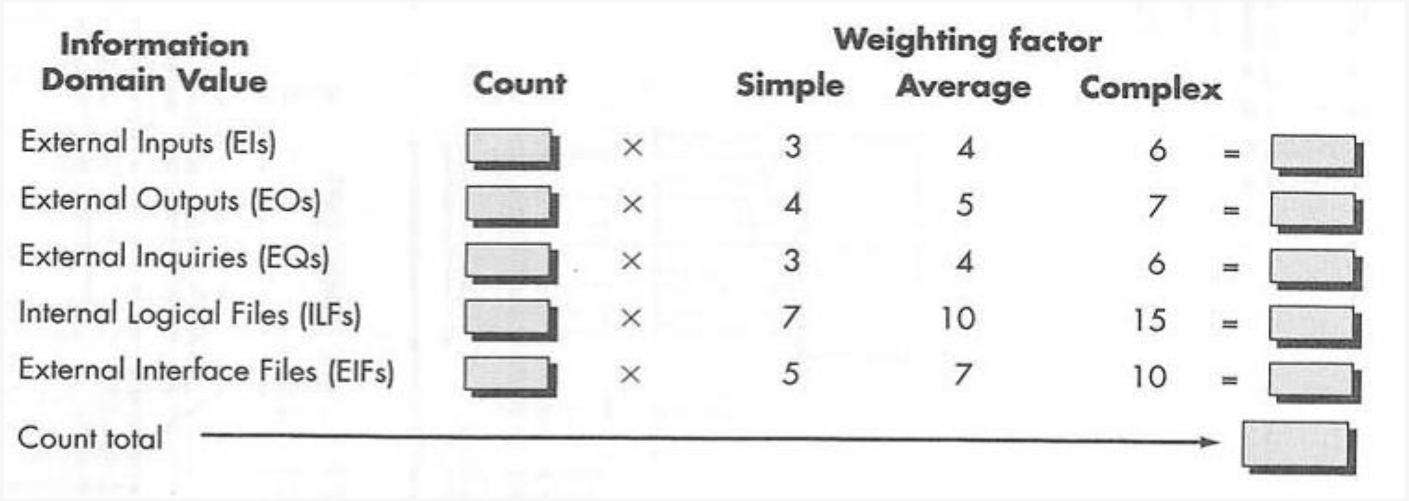
1. You have been appointed a project manager for a major software products company. Your job is to manage the development of the next-generation -version of its widely used wordprocessing software. Because competition is intense, tight deadlines have been established and announced. What team structure would you choose and why?. What software process model(s) would you choose and why?

People
Innovation
Excellence

People
Innovation
Excellence

Exercise

2. **Find the Function Point (FP)**



Information Domain Value	Count		Weighting factor Simple	Average	Complex		
External Inputs (EIs)		×	3	4	6	=	
External Outputs (EOs)		×	4	5	7	=	
External Inquiries (EQs)		×	3	4	6	=	
Internal Logical Files (ILFs)		×	7	10	15	=	
External Interface Files (EIFs)		×	5	7	10	=	
Count total						→	

3. **Assume that the count of all information domain values are 4 and the weighting factor is Average. And also from, the 14 questions to find value adjustment factors is answered 3 as average. If the project software that have been done with FP 120 needs $ 3000, count the estimate cost for the above project ?**

References

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- Project Management

  http://www.pricesystems.com/resources/mf_risks_remedies_facts.asp
- Project Portfolio Management

  http://www.daptiv.com/index.htm
- SW Metrics

  http://www.spc.ca/resources/metrics/
- Function Point Measurement

  http://www.functionpoints.com
- SW Metrics service Estimation

  http://www.charismatek.com/_public4/html/services/pdf/service_estimate.pdf

BINUS UNIVERSITY

Q & A

People
Innovation
Excellence