BINUS UNIVERSITY

People
Innovation
Excellence

Course	: COMP6100/Software Engineering
Effective Period	: Desember 2017

Project Management Concepts and Software Metrics

Session 20 - 21

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

BINUS UNIVERSITY

Contents

The Management Spectrum

The Four P's

- People — the most important element of a successful project
- Product — the software to be built
- Process — the set of framework activities and software engineering tasks to get the job done
- Project — all work required to make the product a reality

People
Innovation
Excellence

BINUS UNIVERSITY

The Management Spectrum

Stakeholders

- ***Senior managers***
  - who define the business issues that often have significant influence on the project.
- ***Project (technical) managers***
  - who must plan, motivate, organize, and control the practitioners who do software work.
- ***Practitioners***
  - who deliver the technical skills that are necessary to engineer a product or application.
- ***Customers***
  - who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
- ***End-users***
  - who interact with the software once it is released for production use.

The Management Spectrum

Software Teams

How to lead?

How to organize?

How to collaborate?

How to motivate?

How to create good ideas?

BINUS UNIVERSITY

The Management Spectrum

Team Leader

- The MOI Model
  - **Motivation.** The ability to encourage (by "push or pull") technical people to produce to their best ability.
  - **Organization.** The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
  - **Ideas or innovation.** The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

People
Innovation
Excellence

The Management Spectrum

Software Teams

The following factors must be considered when selecting a software project team structure ...

- **the difficulty of the problem to be solved**
- **the size of the resultant program(s) in lines of code or function points**
- **the time that the team will stay together (team lifetime)**
- **the degree to which the problem can be modularized**
- **the required quality and reliability of the system to be built**
- **the rigidity of the delivery date**
- **the degree of sociability (communication) required for the project**

People
Innovation
Excellence

The Management Spectrum

Organizational Paradigms

- closed paradigm
  - structures a team along a traditional hierarchy of authority
- random paradigm
  - structures a team loosely and depends on individual initiative of the team members
- open paradigm
  - attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
- synchronous paradigm
  - relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves

suggested by Constantine [Con93]

The Management Spectrum

Agile Teams

- Team members must have trust in one another.
- The distribution of skills must be appropriate to the problem.
- Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained.
- Team is “self-organizing”
  - An adaptive team structure
  - Uses elements of Constantine’s random, open, and synchronous paradigms
  - Significant autonomy

BINUS UNIVERSITY

The Management Spectrum

The Project

- *Projects get into trouble when …*
  - Software people don't understand their customer's needs.
  - The product scope is poorly defined.
  - Changes are managed poorly.
  - The chosen technology changes.
  - Business needs change [or are ill-defined].
  - Deadlines are unrealistic.
  - Users are resistant.
  - Sponsorship is lost [or was never properly obtained].
  - The project team lacks people with appropriate skills.
  - Managers [and practitioners] avoid best practices and lessons learned.

People
Innovation
Excellence

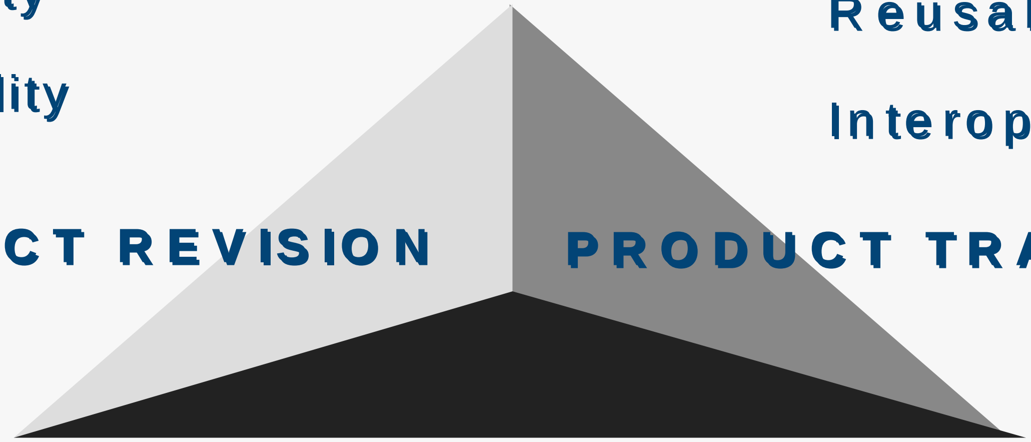
The Management Spectrum

Common-Sense Approach to Projects

- *Start on the right foot.* This is accomplished by working hard (very hard) to understand the problem that is to be solved and then setting realistic objectives and expectations.
- *Maintain momentum. The* project manager must provide incentives to keep turnover of personnel to an absolute minimum, the team should emphasize quality in every task it performs, and senior management should do everything possible to stay out of the team's way.
- *Track progress.* For a software project, progress is tracked as work products (e.g., models, source code, sets of test cases) are produced and approved (using formal technical reviews) as part of a quality assurance activity.
- *Make smart decisions.* In essence, the decisions of the project manager and the software team should be to "keep it simple."
- *Conduct a postmortem analysis.* Establish a consistent mechanism for extracting lessons learned for each project.

A Framework for Product Metrics

McCall's Triangle of Quality



Maintainability

Flexibility

Testability

Portability

Reusability

Interoperability

PRODUCT REVISION

PRODUCT TRANSITION

PRODUCT OPERATION

Correctness

Usability

Efficiency

Reliability

Integrity

People
Innovation
Excellence

People
Innovation
Excellence

A Framework for Product Metrics

Measures, Metrics and Indicators

- 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

Measurement Principles

- The objectives of measurement should be established before data collection begins;
- Each technical metric should be defined in an unambiguous manner;
- Metrics should be derived based on a theory that is valid for the domain of application (e.g., metrics for design should draw upon basic design concepts and principles and attempt to provide an indication of the presence of an attribute that is deemed desirable);
- Metrics should be tailored to best accommodate specific products and processes [Bas84]

A Framework for Product Metrics

Measurement Process

- *Formulation*. The derivation of software measures and metrics appropriate for the representation of the software that is being considered.
- *Collection*. The mechanism used to accumulate data required to derive the formulated metrics.
- *Analysis*. The computation of metrics and the application of mathematical tools.
- *Interpretation*. The evaluation of metrics results in an effort to gain insight into the quality of the representation.
- *Feedback*. Recommendations derived from the interpretation of product metrics transmitted to the software team.

A Framework for Product Metrics

Goal-Oriented Software Measurement

- The Goal/Question/Metric Paradigm
  - (1) establish an explicit measurement *goal* that is specific to the process activity or product characteristic that is to be assessed
  - (2) define a set of *questions* that must be answered in order to achieve the goal, and
  - (3) identify well-formulated *metrics* that help to answer these questions.
- Goal definition template
  - Analyze {the name of activity or attribute to be measured}
  - for the purpose of {the overall objective of the analysis}
  - with respect to {the aspect of the activity or attribute that is considered}
  - from the viewpoint of {the people who have an interest in the measurement}
  - in the context of {the environment in which the measurement takes place}.

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

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

People
Innovation
Excellence

Metrics for the Requirements Model

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 $\longrightarrow$ $\boxed{}$

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

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

Metrics for the Design Model

People
Innovation
Excellence

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

Distinguishing Characteristics

Berard [Ber95] argues that the following characteristics require that special OO metrics be developed:

- Localization—the way in which information is concentrated in a program
- Encapsulation—the packaging of data and processing
- Information hiding—the way in which information about operational details is hidden by a secure interface
- Inheritance—the manner in which the responsibilities of one class are propagated to another
- Abstraction—the mechanism that allows a design to focus on essential details

BINUS UNIVERSITY

People
Innovation
Excellence

Metrics for the Design Model

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

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

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 WebApps

- 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?

People
Innovation
Excellence

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]).

Metrics for Testing

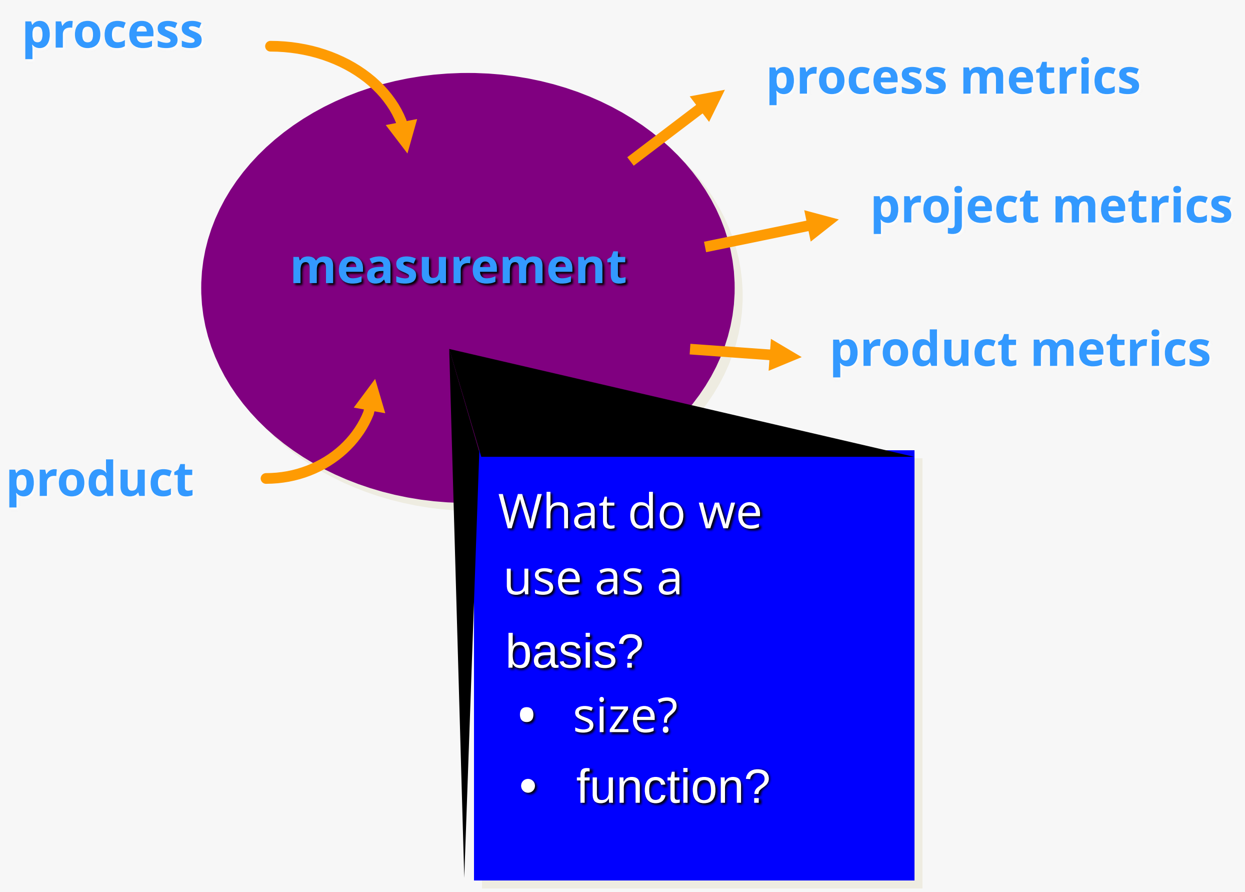
- Testing effort can also be estimated using metrics derived from Halstead measures
- Binder [Bin94] suggests a broad array of design metrics that have a direct influence on the "testability" of an OO system.
  - Lack of cohesion in methods (LCOM).
  - Percent public and protected (PAP).
  - Public access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

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 = $[M_T - (F_a + F_c + F_d)]/M_T$

Metrics in the Process and Project Domain

A Good Manager Measures



What do we use as a basis?

- size?
- function?

People
Innovation
Excellence

Metrics in the Process and Project Domain

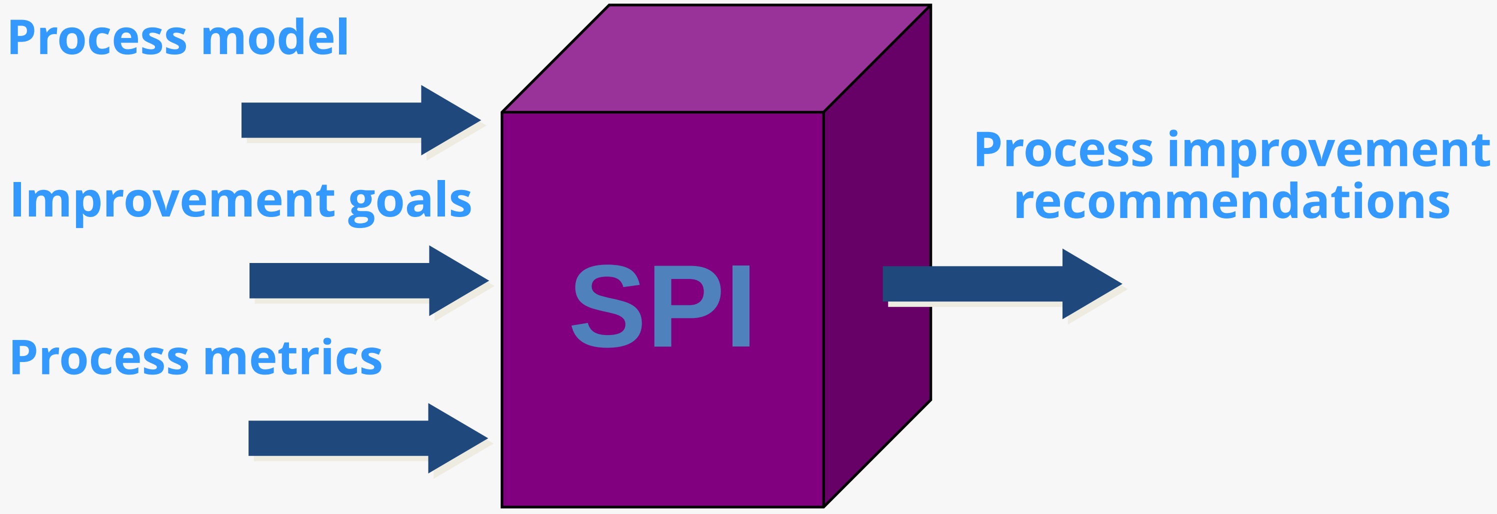
Process Measurement

- 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



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

People
Innovation
Excellence

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.

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

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

People
Innovation
Excellence

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

Software Measurement

People Innovation Excellence

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

People
Innovation
Excellence

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

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

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.

People
Innovation
Excellence

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
- Function Point Measurement

  http://www.functionpoints.com
- Software Metrics

  http://www.spr.com/software-metrics-a-metrics-counsel.html
- SW Metrics service Estimation http://www.charismatek.com/_public4/html/services/pdf/service_estimate.pdf

Q & A

People
Innovation
Excellence