SEENG 421: Software Metrics

Winter 2008 Edition

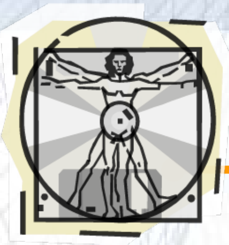
Department of Electrical & Computer Engineering, University of Calgary

B.H. Far（far@ucalgary.ca）

http://www.enel.ucalgary.ca/People/far/Lectures/SENG421/index.html

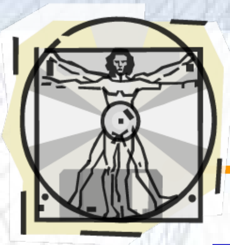
Outline of The Course

- **Module 1:** Measurement theory
- **Module 2:** Software product and process measurements
- **Module 3:** Measurement management



1: Measurement Theory

- Overview of software metrics
- The basics of measurement
- Framework for software measurement
- Empirical investigation

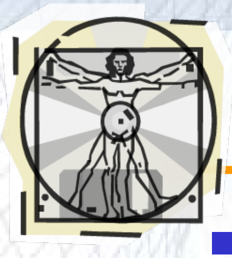


2: SE Measurement

- Software metrics data collection
- Analyzing software measurement data
- Measuring internal product attributes: size and structure (complexity)
- Measuring external product attributes: quality and reliability
- Making process predictions: estimate effort, size, release date, etc.

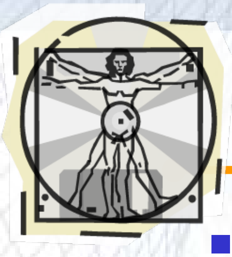
3: Measurement Management

- Planning measurement
- Resource management: productivity, teams, and tools
- Benefits of metrics
- Support for measurement



At the End …

- What is software measurement about?
- Why software measurement is important?
- What does empirical investigation mean in the SE context?
- Is software measurement equivalent to software metrics? What makes them different?
- What are common software metrics that you already know?
- What attributes of the software you suggest to be measured?
- What is software measurement process?
- How to implement a software measurement plan?
- What are challenges and difficulties of applying software metrics?



Reference Books

- *Software Metrics: A Rigorous and Practical Approach*, (2nd ed.) (638p.), N.E. Fenton and S.L. Pfleeger, PWS Publishing, 1998, ISBN 0-534-95425-1. (Recommended)
- *Metrics and Models in Software Quality Engineering* (344 pages), Stephen H. Kan, Addison-Wesley, 1995, ISBN: 0201633396. (Recommended)
- *Software Metrics: Measurement for Software Process Improvement*, B.A. Kitchenham, Blackwell Pub, 1996, ISBN 1855548208.
- *Software Metrics: Establishing a Company-wide Program* (288 pages), R.B. Grady, D.L. Caswell, Prentice Hall, 1998, ISBN 0138218447.
- *Applied Software Measurement: Assuring Productivity and Quality*, C. Jones, McGraw-Hill, 1996.
- *Software Engineer's Reference Book*, J. McDermid (Edt.), Butterworth Heinemann, 1993, ISBN 0-7506-0813-7.

Moral

- Industrial oriented course.
- Attend lectures: there is more to lectures than to the notes.
- Sit close-up: some of the material is hard to see from the back.

- Feel free to ask questions.

Part 1: Overview of Software Metrics

Introduction

(Chapter 1)

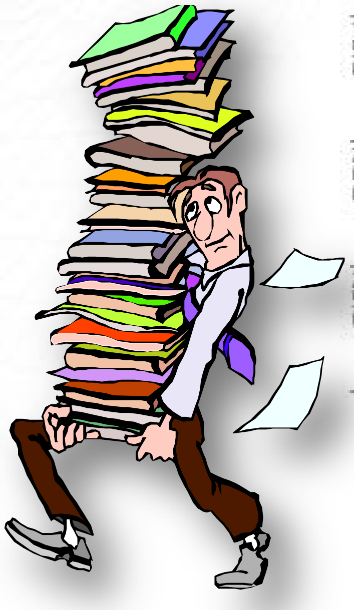
Department of Electrical & Computer Engineering, University of Calgary

B.H. Far（far@ucalgary.ca）

http://www.enel.ucalgary.ca/People/far/Lectures/SENG421/01/

Contents

- What is measurement?
- What are software metrics?
- Scope of software engineering metrics: a chronological review.



What is Measurement?

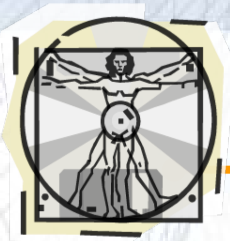
Measurement: Definition /1

- **Measurement** is the process by which numbers or symbols are assigned to attributes of entities (objects) in the real world in such a way as to ascribe them according to defined rules.

$$Object = \left\{ \begin{array}{ll} attribute_1 & (value_{11}, value_{12}, \ldots) \\ attribute_2 & (value_{21}, value_{22}, \ldots) \\ \ldots & \ldots \\ attribute_n & (value_{n1}, value_{n2}, \ldots) \end{array} \right\}$$

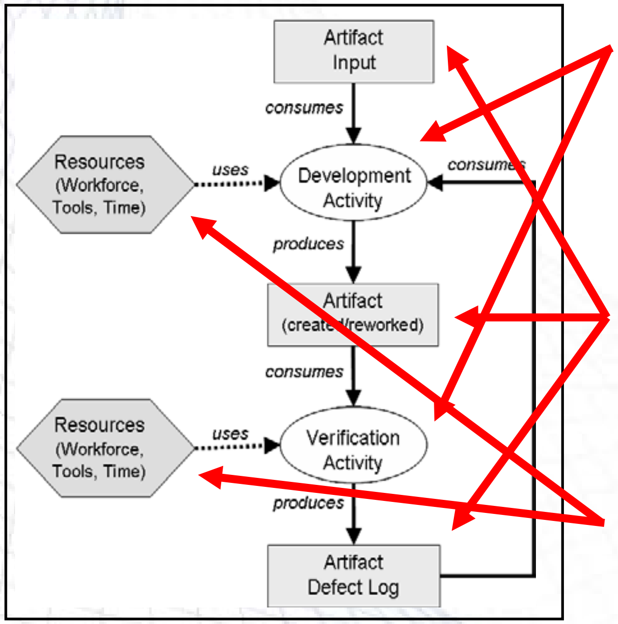
Measurement: Definition /2

- **Metrics** are standards (i.e., commonly accepted scales) that define measurable attributes of entities, their units and their scopes.
- **Measure** is a relation between an attribute and a measurement scale.

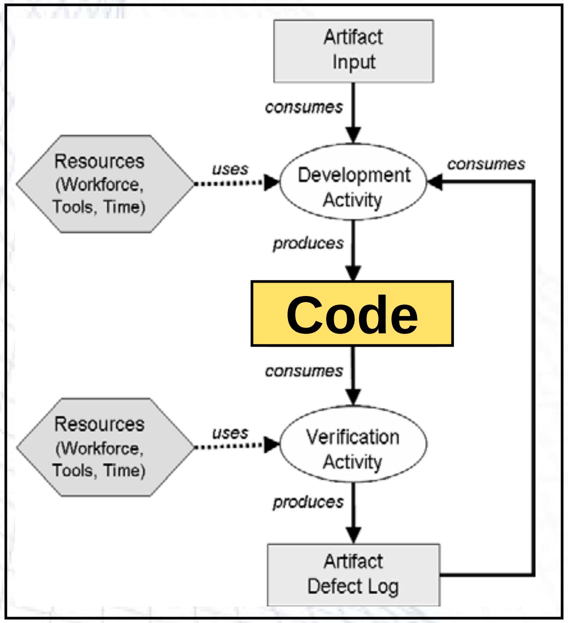


Entity

- **An entity in software measurement can be any of the following:**



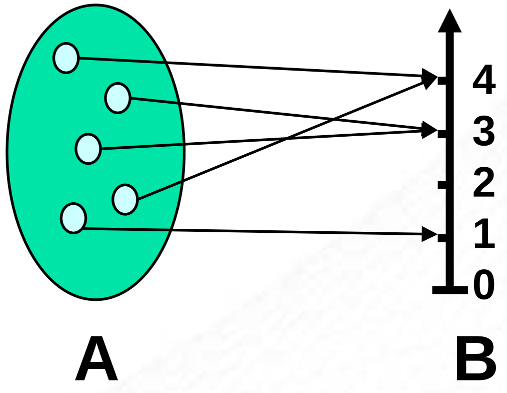
Attribute



- **An attribute is a feature or property of an entity**
  - e.g., blood pressure of a person, cost of a journey, duration of the software specification process
- There are two general types of attributes:
  - Internal attributes can be measured only based on the entity itself,
    - e.g., entity: code, internal attribute: size, modularity, coupling
  - External attributes can be measured only with respect to how the entity relates to its environment
    - e.g., entity: code, external attribute: reliability, maintainability

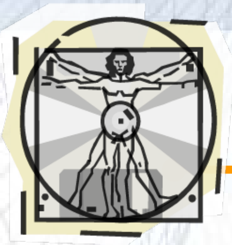
Measurement Example

Entity	Attribute
Requirements	Size, Reuse, Redundancy
Specification	Size, Reuse, Redundancy
Design	Size, Reuse, Modularity, Cohesion, Coupling
Code	Size, Reuse, Modularity, Cohesion, Coupling, Complexity
Test Cases	Size, Coverage



Measurement: Types

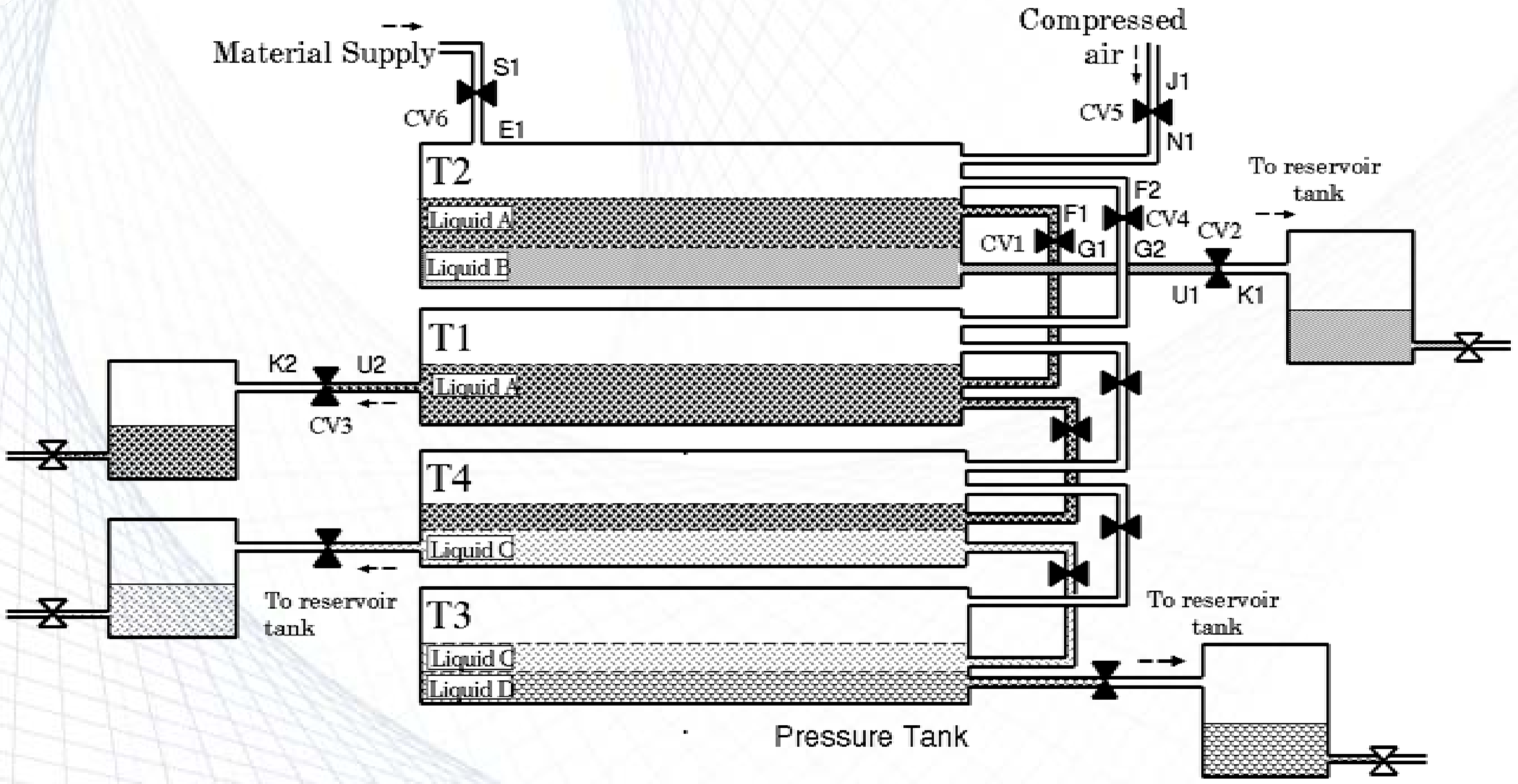
- Measurements are needed as:
  - **Descriptors** of entities already in existence.
  - **Prescriptors** (standards, norms, failure intensity objectives, benchmarks) which entities of certain class or category should satisfy.
  - **Predictors** to estimate properties of entities yet to be designed or implemented.



Measurement : How to

- In order to make entities measurable:
  - What **entities (objects)** should be selected?
  - What **attributes** should be selected?
  - What **values** should be assigned to the attributes?
  - What shall be the **rules (relationships)** ascribed to the attributes and their entities?
- Note: assigned values and/or ascribed rules can be either quantitative or qualitative.

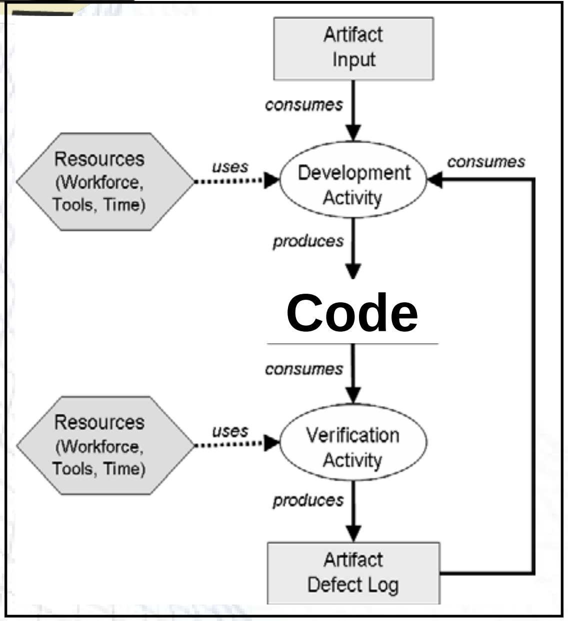
Example 1: Pressure Tank /1



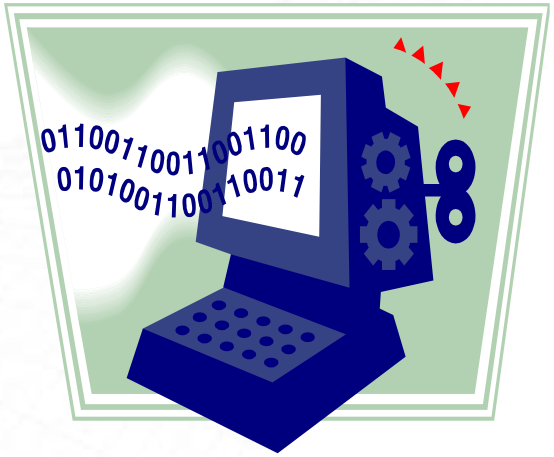
Example 1: Pressure Tank /2

- **Entities (Objects):**
  - Tanks (T_1, T_4), Valves (CV_1, CV_6), Pipes
- **Attributes:**
  - Level of liquid, pressure of liquids, flow of material
- **Values:**
  - Tank full, Tank empty, etc.
- **Rules:**
  - Relations among level, pressure and flow of material.

Example 2: Code



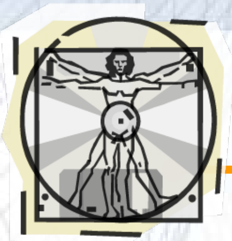
- Entity: Code
- Attribute: Size
- Possible measures:
  - NCSLOC (Not Commented Source Lines of Code)
  - #Statements
  - #Modules
  - #Procedures
  - #Classes
  - #Methods
  - …



Example 3: Availability

- **Entity:** Availability
- **Attributes:** system uptime, system downtime
- **Values:** time in seconds
- **Relations:**

  Availability = uptime / (uptime + downtime)



Software Metrics Challenges

- Measuring physical entities:

entity	attribute	unit	value
Human	Height	cm	178

- Measuring non-physical entities:

entity	attribute	unit	value
Human	IQ	IQ index	89

- SE metrics are mostly non-physical
  - Reliability, maturity, portability, flexibility, maintainability, etc. and relations are unknown

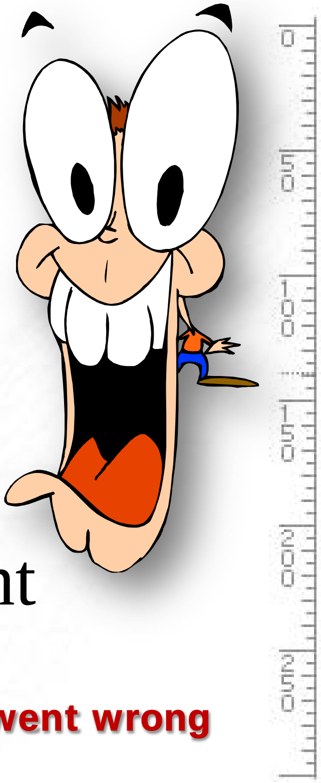
Misleading Metrics!

- Fact (1): Knowledge is power
- Fact (2): Time is money
- Relation (rule) :   power = work / time

Substituting “power” and “time”

- Knowledge = work / money

- As knowledge approaches zero money approaches infinity regardless of the amount of work done!
- Conclusion:
  - The less you know, the more you make!



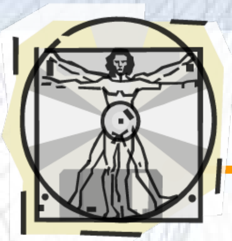
What went wrong here?

- Counter intuitive
- Needs validation

UNIVERSITY OF CALGARY

What is Software Measurement?

What Is Software Measurement?



- Software metrics are measures that are used to quantify software, software development resources, and/or the software development process.
- This includes items which are directly measurable, such as *lines of code*, as well as items which are calculated from measurements, such as *software quality*.

Measurement in SE

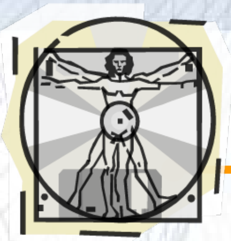
- Measurement in SE is selecting, measuring and putting together many different attributes of the software, and adding our subjective interpretations in order to get a whole picture of the software.
- This is not a trivial task!
- 300+ metrics have been defined.

Measurement in SE /1

- Before a measurement project can be planned
  - Objectives and scope should be established
  - Alternative solutions should be considered
  - Technical and management constraints should be identified.
- This information is required to estimate costs, project tasks, and a project schedule.

Measurement in SE /2

- In order to manage software measurement project one must understand and plan:
  - The goal and scope of work
  - Risks
  - Resources required
  - Tasks to be accomplished
  - Milestones to be tracked
  - Total costs of the project
  - Schedule to be followed



Measurement in SE /3

- Software metrics help us understand the technical process that is used to develop a software product.
  - The process is measured to be improved.
  - The product is measured to increase its quality.

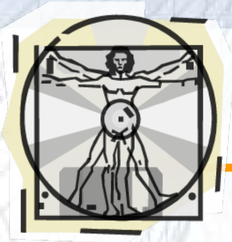
But …

- Measuring software projects is controversial.
- It is not yet clear which are the appropriate metrics for a software project or whether people, processes, or products can be compared using metrics.

Scope of Software Metrics

- Cost and effort estimation
- Productivity measures and models
- Data collection
- Quality models and measures
- Reliability models
- Performance evaluation and models
- Structural and complexity metrics
- Capability maturity assessment
- Management by metrics
- Evaluation of methods and tools

More about scope in the next session



Example 1

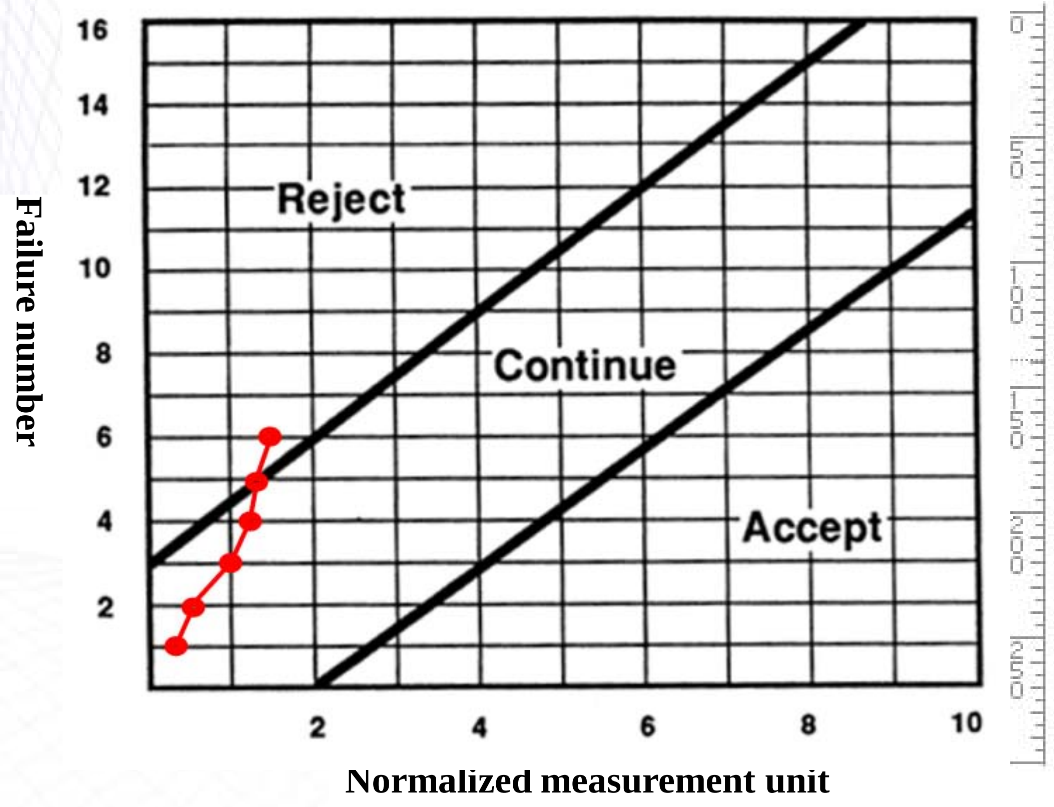
- We are going to buy a new colour laser printer for our department. We have borrowed the printer for the test.
- Maker's data shows that we need to change the toner every 10,000 pages. We would like to have only one failure during the period.
- During test period, we observe that failures occur at 4,000 pages, 6,000 pages, 10,000 pages, 11,000 pages, 12,000 pages and 15,000 pages of output.
- What can we conclude about this printer?

Example 1 (cont'd)

- Failure intensity objective:

λ_F = 1/10000 pages

- Using reliability demonstration chart we can conclude that the printer must be rejected.



Example 2

- We are going to initiate a new game and video on-demand download service. The service is provided to the customers who own PCs and register with the service.
- The customers must use specialized software to download games or videos from the server. The failure intensity of the software is 1 failure per 100 CPU hr. On average, the specialized software system runs 20 CPU hr per week on each client machine and there are 800 customers to be serviced.
- We would like to provide the customers with an on-site repair service. Each serviceperson can make 4 service calls per day. Service personnel are available 5 working days per week.
- How many service personnel do we need to hire?

Example 2 (cont'd)

- How many service personnel do we need?

- Using the value for failure intensity, each system experiences 0.2 failure per 20 hours of operation or 0.2 failure per week, on average.
- The total failures for 800 customers is 160 per week or 32 per day.
- Each serviceperson can visit 4 sites per day, therefore, the number of required personnel is

  32 / 4 = 8.

Example 3

- Failure intensity of a system is usually expressed using FIT (Failure-In-Time) unit which is 1 failure per 10**9 device hours. The failure intensity of an electric pump system used for pumping crude oil in Northern Alberta's oil field is constant and is 10,000 FITs and 100 such pumps are operational. If for continuous operation all failed units are to be replaced immediately, what would be the minimum inventory size of pump units for one year of operation?

Pump's Mean-Time-To-Failure (MTTF)
λ = 10,000 FITs = 10,000 / 109 hour = 1×10**-5 hour**
= 1 failure per 100,000 hours

The 12-month reliability is: (1 year = 8,760 hours)
R(8,760 hours) = exp{-8,760/100,000} = 0.916 and "unreliability" is,
F(8,760) = 1 - 0.916 = 0.084

Therefore, inventory size is 8.4% or minimum 9 pumps should be at stock in the first year.

UNIVERSITY OF CALGARY

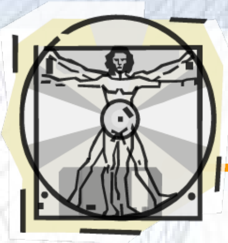
Overview: Scope of Software Metrics

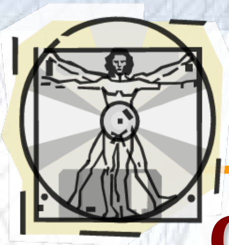
Process, Product and Resources

Scope of Software Metrics

- Cost and effort estimation
- Productivity measures and models
- Data collection
- Quality models and measures
- Reliability models
- Performance evaluation and models
- Structural and complexity metrics
- Capability maturity assessment
- Management by metrics
- Evaluation of methods and tools

ITEMS IN RED ARE COVERED IN THIS COURSE





Scope of Software Metrics /1

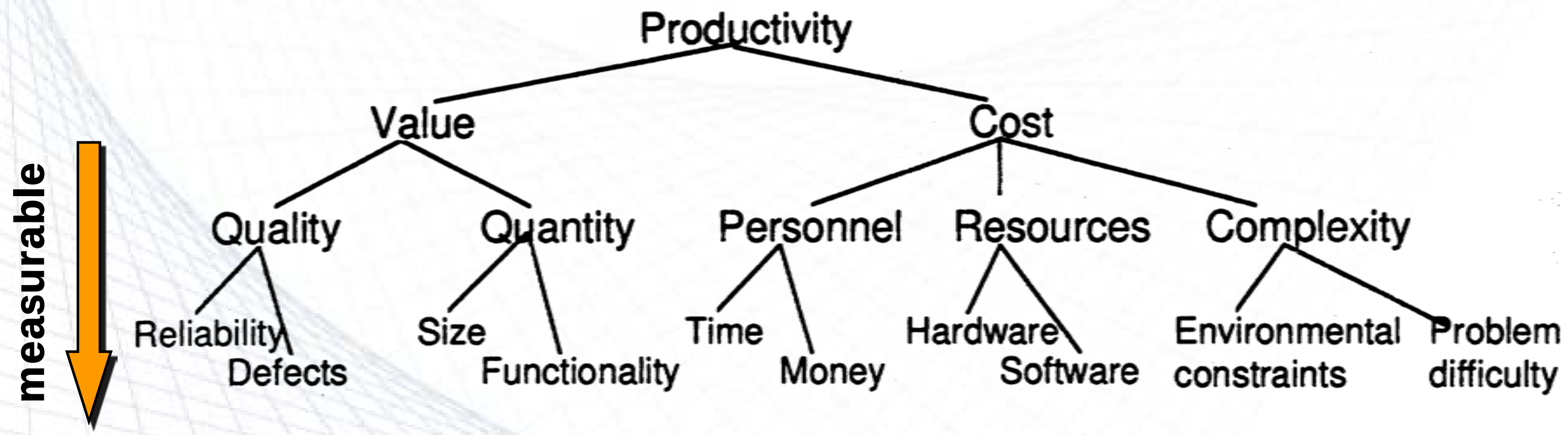
Cost and effort estimation

- Software cost estimation is the process of predicting the amount of effort required to build a software system.
- Estimates for project cost and time requirements are derived during the planning stage of a project.
- Models used to estimate cost can be categorized as either cost models (e.g., Constructive Cost Model COCOMO) or constraint models (e.g., SLIM).
- Experience is often the only guide used to derive these estimates, but it may be insufficient if the project breaks new ground.
- Many models are available as automated tools.

Scope of Software Metrics /2

Productivity models and measures

- Definition: The rate of output per unit of input.
  - Productivity = size / effort
  - Productivity = LOC / person-month
- Productivity model based on decomposition to measurable attributes:



Scope of Software Metrics /3

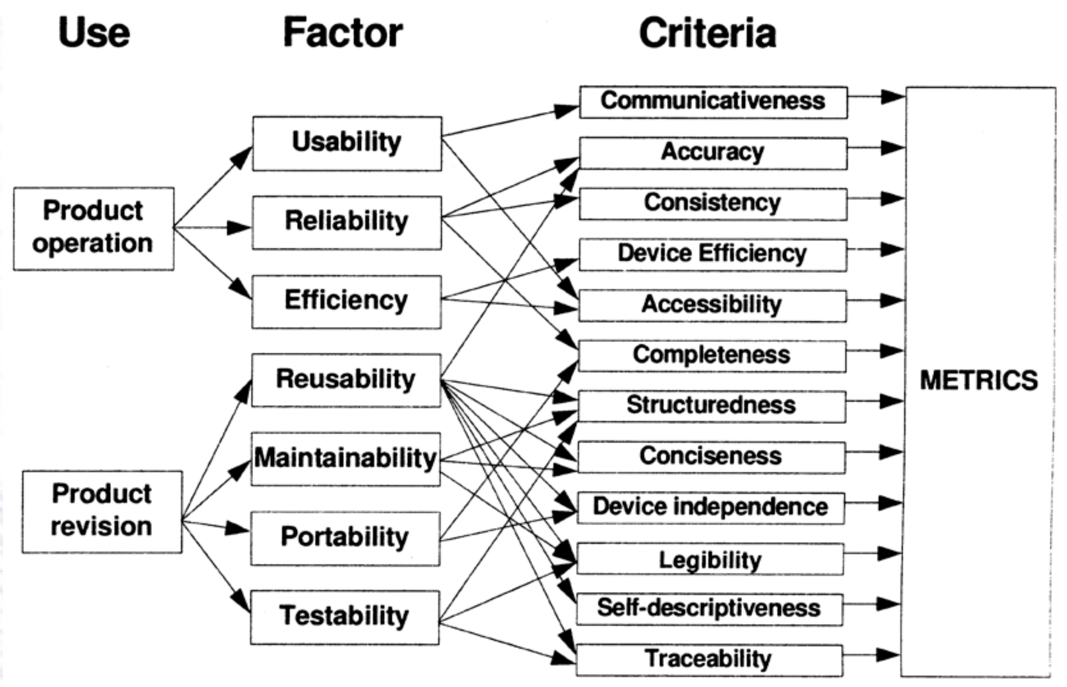
Data collection

- Very critical and very hard step.
  - What data should be collected?
  - How it should be collected?
  - Is collected data reproducible?
- **Example:** software failure data collection
  1) Time of failure
  2) Time interval between failures
  3) Cumulative failure up to a given time
  4) Failures experienced in a time interval

Scope of Software Metrics /4

Quality models and measures

- Software quality measurement (Rubey and Hartwick 1968~)
- McCall's quality factors (1977~), ISO 9126



Scope of Software Metrics /5

Reliability models

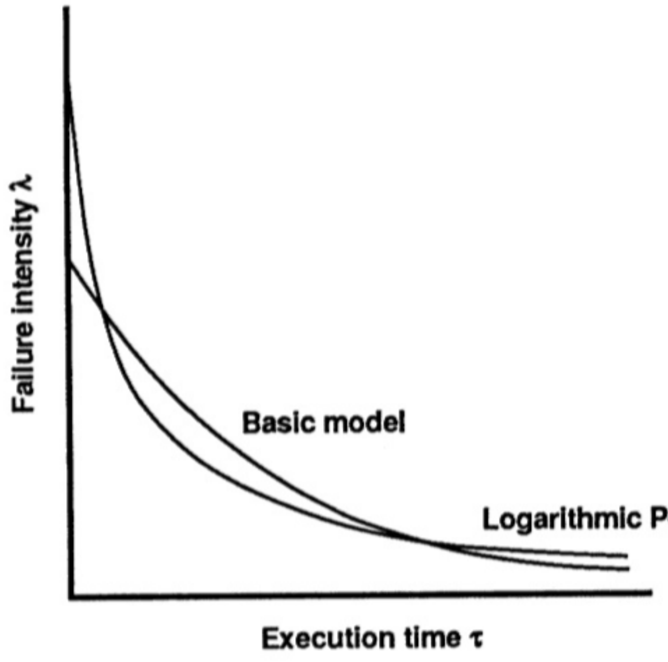
- Plot the change of failure intensity (λ) against time.
- Many models are proposed. The most famous ones are basic exponential model and logarithmic Poisson model.
- The basic exponential model assumes finite failures in infinite time; the logarithmic Poisson model assumes infinite failures.
- Automated tools such as CASRE are available.

Reliability Models

- Failure intensity (λ) versus execution time (τ)

(B) $$\lambda(\tau) = \lambda_0 e^{\left(-\frac{\lambda_0}{\nu_0}\right)\tau}$$

(P) $$\lambda(\tau) = \frac{\lambda_0}{\lambda_0 \theta \tau + 1}$$



Scope of Software Metrics /6

Performance evaluation and models

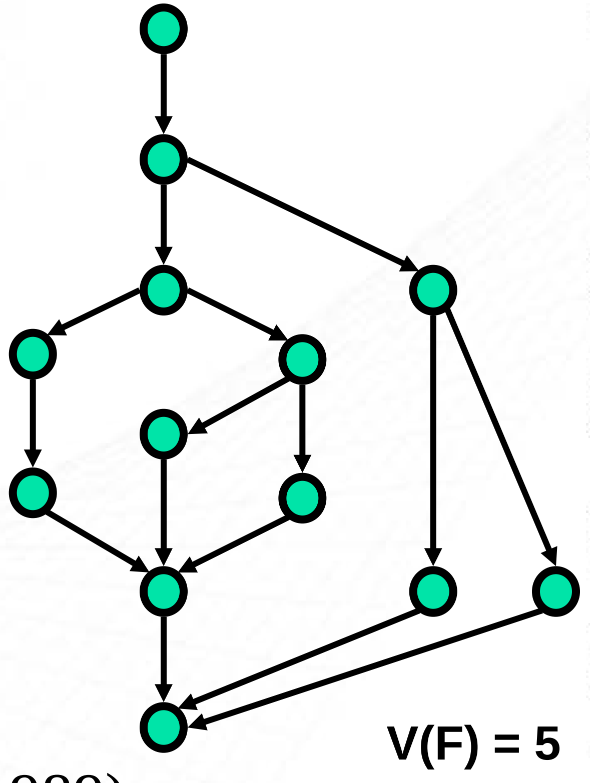
- Using externally observable performance characteristics such as response time and completion rate (Ferrari 1978~)
- Efficiency of algorithms (Garey 1979~)

Scope of Software Metrics /7

Structural and complexity metrics

- Control-flow structure
- Data-flow structure
- Data structure
- Information flow attributes

- Complexity metrics (1979~)
  - Cyclomatic complexity (McCabe 1989) defining number of independent paths in execution of a program



Scope of Software Metrics /8

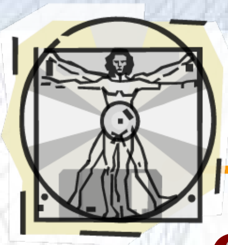
Management by metrics

- Metrics for project control (1980~)
  - Metrics related to specification quality
  - Metrics for the design model
  - Metrics for documentation
  - Checking and testing metrics
  - Resource metrics
  - Change metrics

Scope of Software Metrics /9

Evaluation of methods and tools

- Efficiency of methods (1991~)
- Efficiency and reliability of tools
- Certification test of acquired tools and components
- Benchmarking



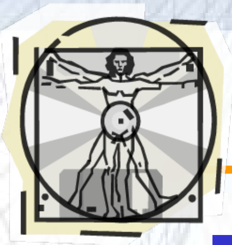
Scope of Software Metrics /10

Capability maturity assessment

- US Software Engineering Institute (SEI) model (1989): CMM grading using five-level scale.
- ISO 9001: Quality systems: models for quality assurance in design/development, production, installation and servicing (1991)
- ISO 9000-3: Guidelines for application of ISO 9001 to the development, supply and maintenance of software (1991)

How to Implement?

- The eight steps required to implement a software measurement program are:
  - Document the software development process
  - State the goals
  - Define metrics required to reach goals ← GQM
  - Identify data to collect
  - Define data collection procedures
  - Assemble a metrics toolset
  - Create a metrics database
  - Define the feedback mechanism



Who Benefits From Measurement

- **Managers**
  - What does each process cost?
  - How productive is the staff?
  - How good is the code being developed?
  - Will the user be satisfied with the product?
  - How can we improve?
- **Engineers**
  - Are the requirements testable?
  - Have we found all the failures?
  - Have we met our product or process goals?
  - What can we predict about our software product in the future?

Exercise

- Suppose that you are asked to study various software development tools and recommend the best three to your company. The following table shows a list of available development tools.

Exercise (cont'd)

Tool Name/Vendor	Languages Supported	Platforms	Features
Bean Machine IBM	Java	Windows, OS2, Unix	Best: Visual applet and JavaBean generation
CodeWarrior Pro Metrowerks	Java, C, C++, Pascal	Unix, Windows, Mac	Best: if you need to support Unix, Windows, and Mac platforms
Java Workshop Sun Microsystems	Java	Solaris, Windows	Better: Written 100% in Java; tools based on a web browser metaphor
JBuilder Imprise	Java	Windows, AS400	Better: database support
Visual Cafe for Java Symantec	Java	Windows	Good: multithreaded debugger
VisualAge IBM	Java	Unix, Windows	Good: includes incremental compiler and automatic version control
Visual J++ Microsoft	Java	Windows	Fair: All the bells and whistles for Windows

Exercise (cont'd)

- What are the *entities*, *attributes* and their *values* in your model?

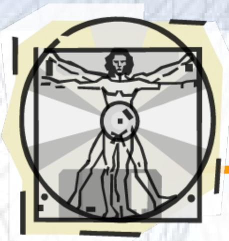
Entity	***Attribute***	***Value***
Development Tool	Language supported	Java, C, C++, Pascal
	Platform	Win, Unix, Mac, OS2, AS400
	Feature	Fair, Good, Better, Best

Conclusion

- Without measurements there is no way to determine if the process/product are improving.
- Metrics allow the establishment of meaningful goals for improvement. **A baseline from which improvements can be measured can be established.**
- Metrics allow us to identify the causes of defects which have major effect on software development.
- When metrics are applied to a product they help identify:
  - which user requirements are likely to change
  - which modules are most error prone
  - how much testing should be planned for each module

References

- [Bas92] V. R. Basili: Software modeling and measurement: The Goal/Question/Metric paradigm. Technical Report CS-TR-2956, Department of Computer Science, University of Maryland, College Park, MD 20742, September 1992.
- [BEM95] L. Briand, K. El Emam, S. Morasca: Theoretical and Empirical Validation of Software Product Measures. ISERN technical report 95-03, 1995.
- [BMB96] L. C. Briand, S. Morasca, and V. R. Basili: Property-Based Software Engineering Measurement. IEEE Transactions on Software Engineering, Vol.22, No.1, January 1996.
- [EM95] K. El Emam and N. H. Madhavji: Measuring the success of the requirements engineering process. Proceedings of the Second IEEE International Symposium on Requirements Engineering, pp. 204-211, 1995.
- [GL89] D. Galletta and A. Lederer: Some cautions on the measurement of user information satisfaction. Decision Sciences, 20:419-438, 1989.
- [GPC02] M. Genero, M. Piattini, C. Calero: Empirical Validation of Class Diagram Metrics. ISESE 2002: 195-203, 2002.
- [HK81] S. M. Henry, D. G. Kafura: Software Structure Metrics Based on Information Flow. IEEE Trans. Software Eng. 7(5): 510-518, 1981.
- [McC76] T. McCabe: A software complexity measure, IEEE Transactions on Software Engineering 2(4):308-320, 1976.
- [McC81] J. McIver and E. Carmines: Unidimensional Scaling. Sage Publications, 1981.
- [Mor01] S. Morasca: Software Measurement, Handbook of Software Engineering and Knowledge Engineering, Vol. I, World Scientific Publishing, 2001.



References

- [MK93] J.C. Munson, T. M. Koshgoftaar: Measurement of Data Structures Complexity, Journal of Systems and Software, 20:217-225, 1993.
- [NASA/SEL] V. R. Basili: Software Modeling and Measurement: The Goal/Question/Metric Paradigm. Technical Report UNIACS-TR-92-96, University of Maryland, September 1992.
- [Nat92] National Aeronautics and Space Administration: Data Collection Procedures for the Software Engineering Laboratory (SEL) Database, March 1992.
- [Nat94] National Aeronautics and Space Administration: Software Measurement Guidebook. SEL-94-002. July 1994.
- [Nun78] J.Nunnaly: Psychometric Theory, McGraw Hill, 1978.
- [Osg67] C. Osgood, G. Suci, and P. Tannenbaum: The measurement of meaning. University of Illinois Press, 1967.
- [Spe72] P. Spector: Choosing response categories for summated rating scales. Journal of Applied Psychology, 61(3):374-375, 1976.
- [Spe80] P. Spector: Ratings of equal and unequal response choice intervals. The Journal of Social Psychology, 112:115-119, 1980.
- [Spe92] P. Spector: Summated Rating Scale Construction: An Introduction. Sage Publications, 1992.
- [Wey88] E. Weyuker: Evaluating Software Complexity Measures. IEEE Transactions on Software Engineering, 14(9):1357-1365, 1988.
- [Zus91] H. Zuse: Software Complexity: Measures and Methods, de Gruyter, 1991.