CP317 Software Engineering

week 7-2 - Metrics

Shaun Gao, Ph.D., P.Eng.

Agenda

- Review week 7-1 topics
- Metrics
- Types of software metrics
  - Process metrics
  - Product metrics
  - Project metrics
- Defect analysis
- Ishikawa diagrams
- Normalization in software metrics
- Software complexity, cyclomatic complexity measure
- Function points
- Summary

Review – week 7-1

- Deployment
  - Concepts
- Types of software deployments
- Deployment tasks
  - Plan, testing, document, training
- Deployment plan
- Deployment strategies
  - Cutover, staged deployment, gradual cutover, parallel, increment
- Deployment mistakes

Introduction to Software Metrics

- The purpose of software metrics
  - Learn from past and improve for the future

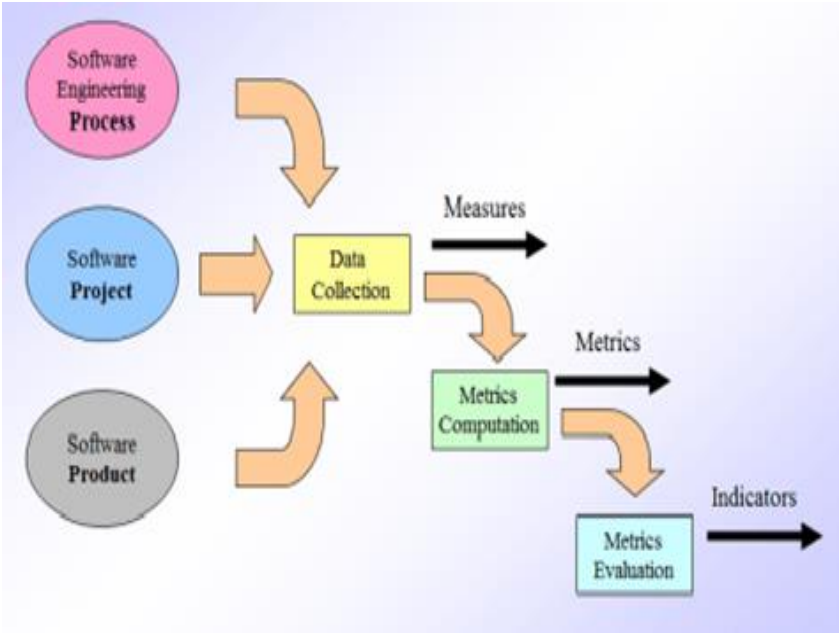
- A software metric is the measurement of software characteristics which are measurable or countable. Software metrics are valuable for many reasons, including measuring software performance, planning work items, measuring productivity, and many other uses.
- Common measurements:
  - Defect rate,
  - Line of code,
  - Complexity

Software Metrics

- Advantages of software metrics
  - It is good for comparative study of various design methodologies of software systems
  - It is good for comparing and evaluating the capabilities and productivities of team members
  - Help to prepare software quality specifications
  - It gets an idea about the complexity of the software code
  - It provides feedback to software engineers and managers about the progress and quality during various phases of the software development life cycle
  - It helps for decision making for future software project.

Types of software metrics

- Software engineering is a process
- Software is a product
- Software is built with projects



Metrics for Software

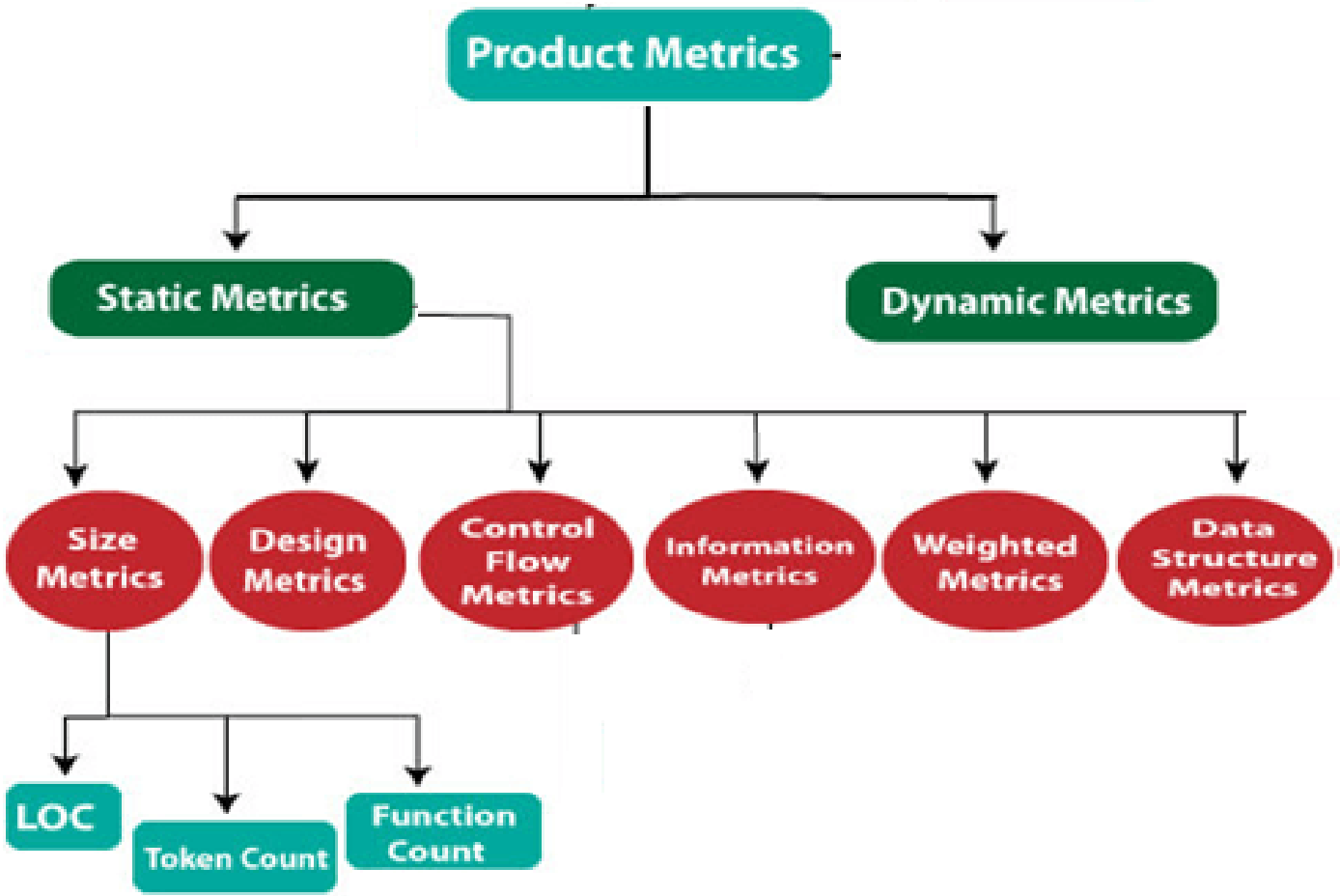
- Product Metrics
  - Indicate the quality of the product produced
- In-process Metrics
  - "Barometers" to indicate whether the process appears to be "working normally"
  - Useful during the development and maintenance process to identify problems and areas for improvement
- Project Metrics
  - Indicate whether project execution (business aspects) are on track

Types of software metrics – cont.

- Product metrics
  - Product metrics are to describes the characteristics of the product such as size, complexity, design features, performance, and quality level.
- Measurements:
  - Size, Line of Code (LoC)
  - Complexity
  - Performance
  - Usability
  - Security

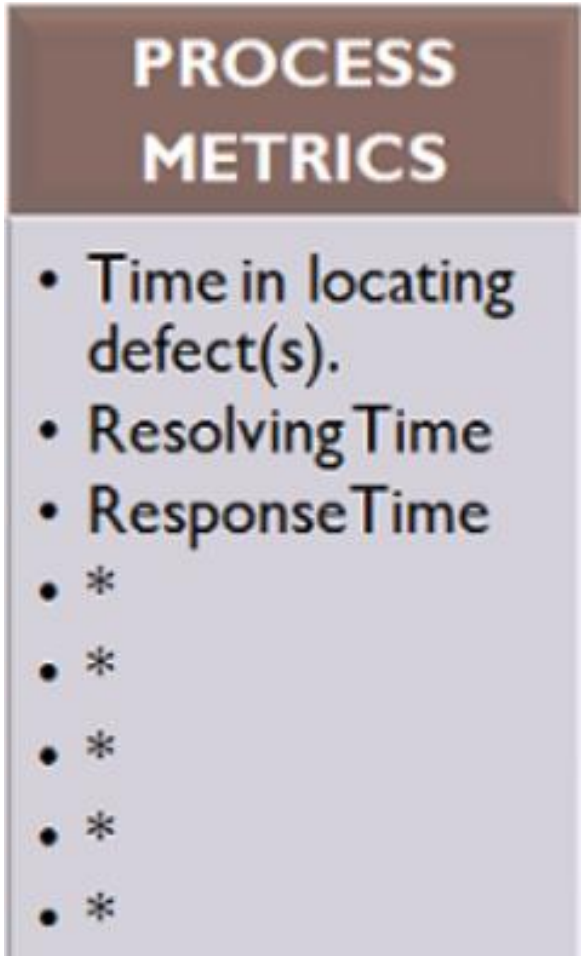
Software product metrics

- Software product metrics



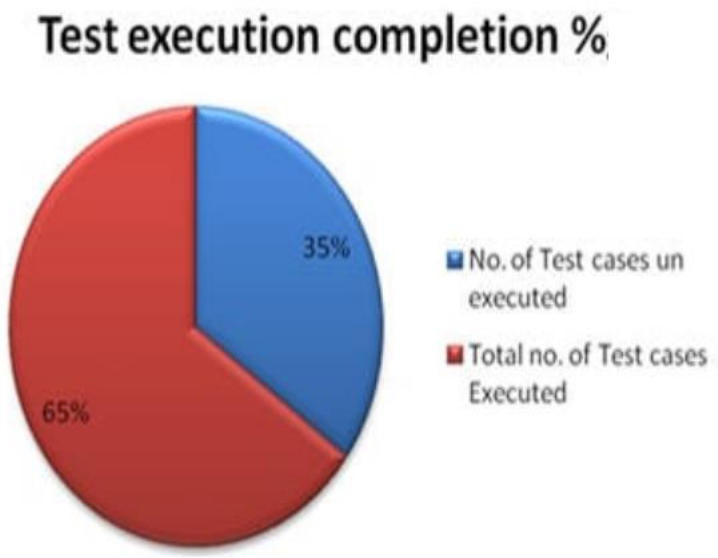
Types of software metrics – cont.

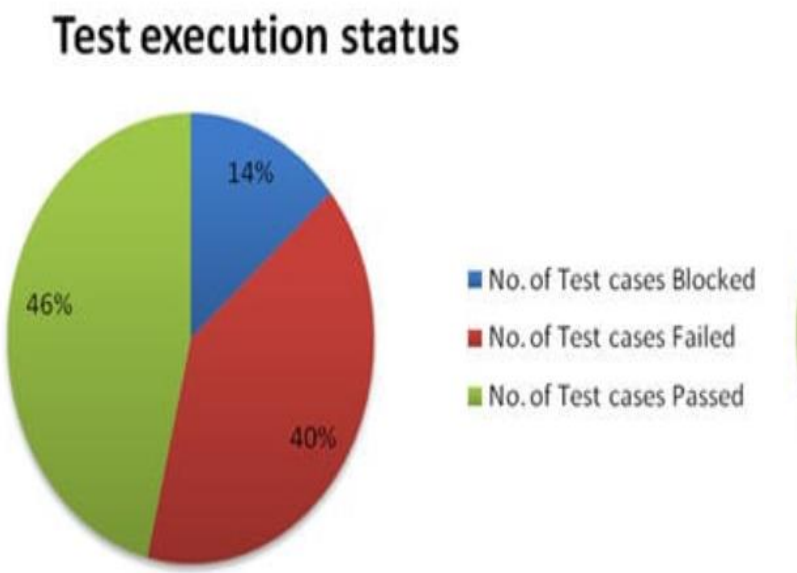
- Process metrics
  - Process metrics are standard measurements that are used to evaluate and benchmark the performance of software engineering processes.
- Measurements
  - Efficiency
  - Productivity
  - Error rate
  - Cost effectiveness

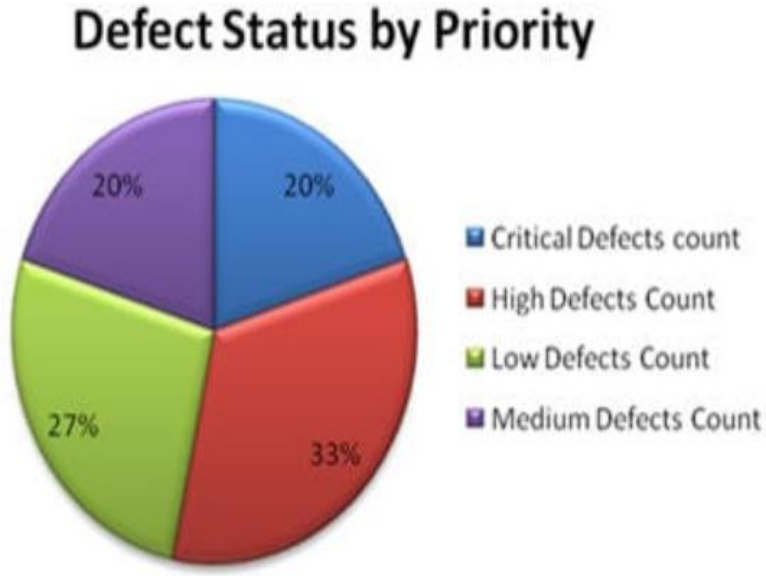


Types of software metrics – cont.

- Process metrics example – testing



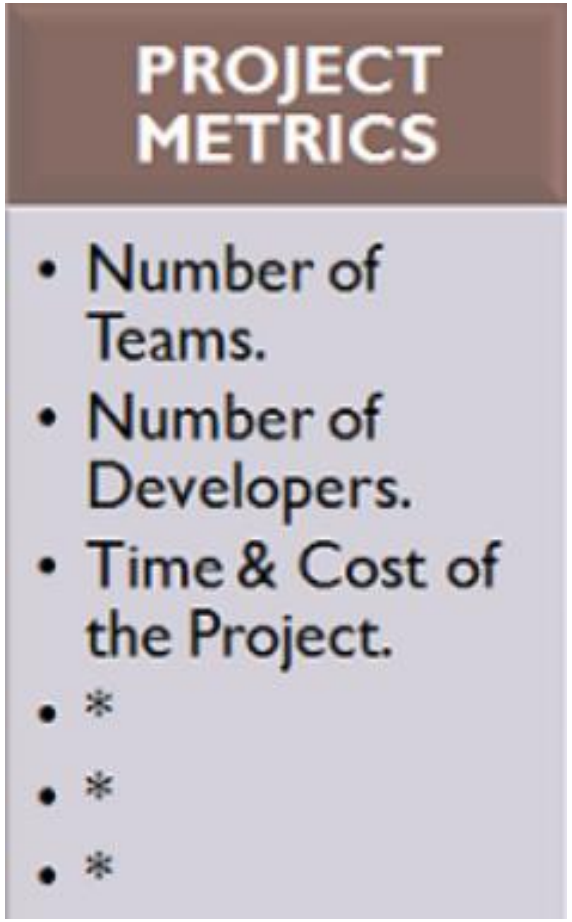




Types of software metrics – cont.

- Project metrics
  - Project metrics describe the project characteristics and execution such as resources, cost, and productivity.

- Measurements: planned vs. actual
  - Number of teams
  - Number of developers
  - Schedule
  - Cost
  - Productivity



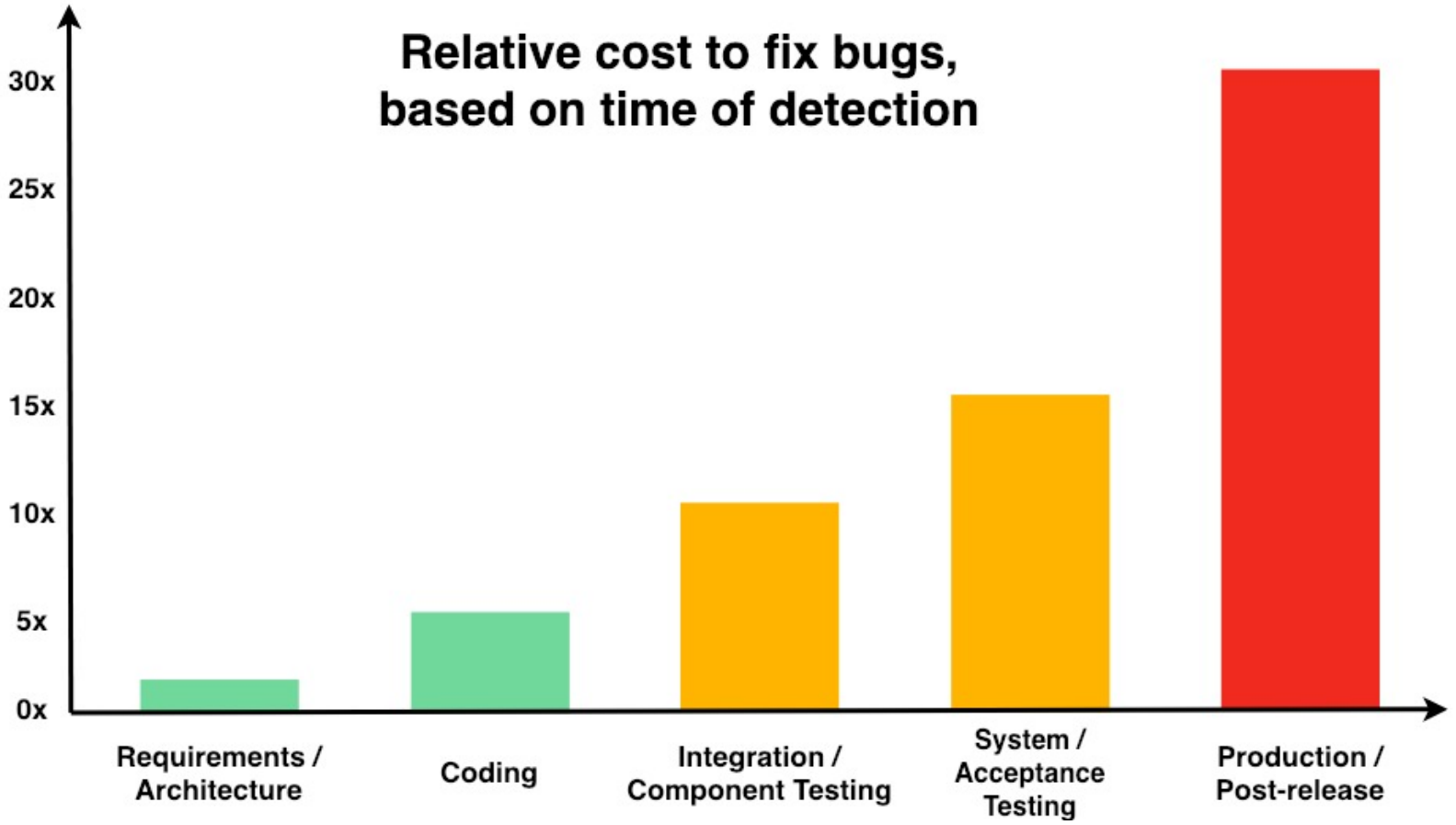
Defect analysis

- Defect analysis is part of the continuous quality improvement in which defects are classified into different categories and are also used to identify the possible causes in order to prevent the problems from occurring again.
  - Discoverer: who found it
  - Severity
  - Time created
  - Root cause: design or coding

Definition:
Defect analysis generally seeks to classify defects into categories and identify possible causes of defects so that they can be prevented or detected earlier

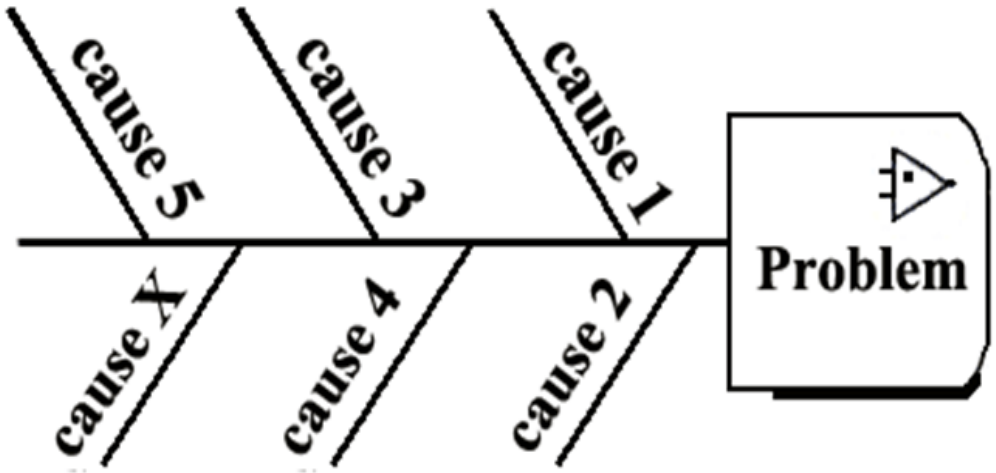
Defect analysis – cont.

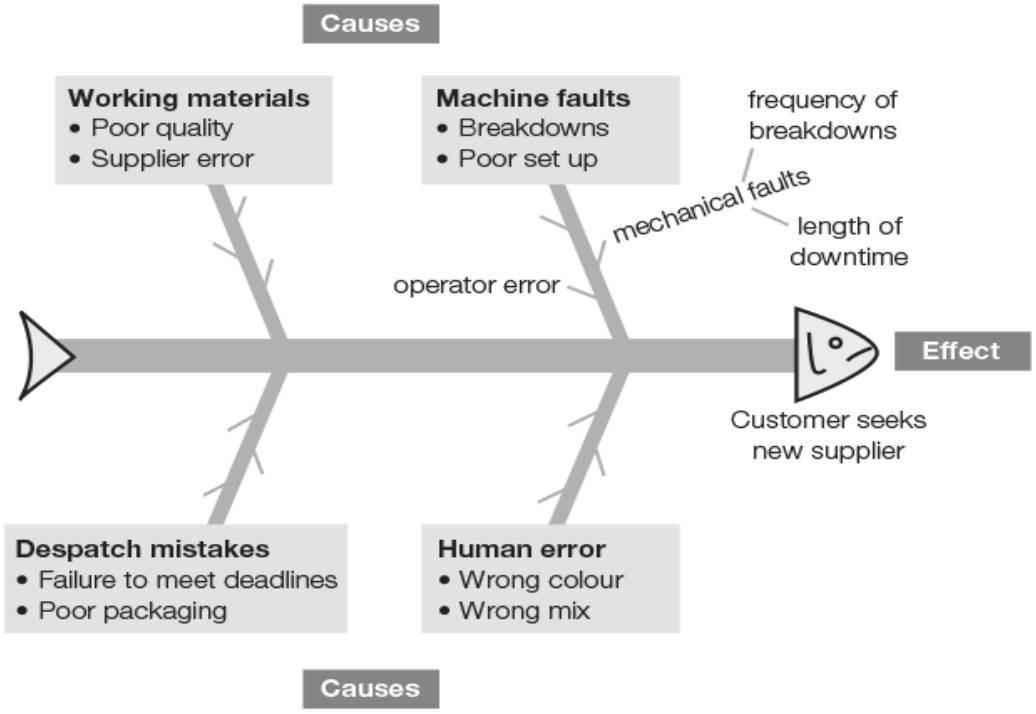
- Defect analysis: timing and economic cost relations



Cause-and-effect diagrams - Ishikawa diagrams

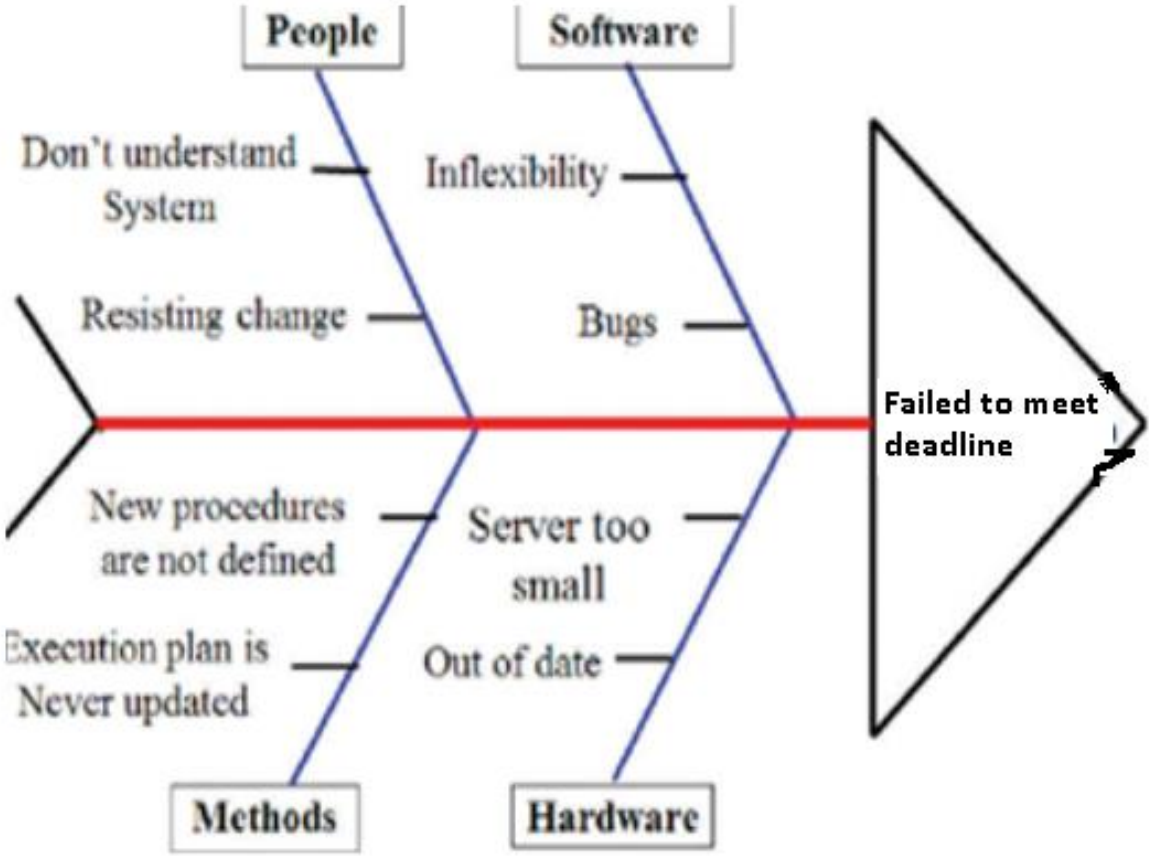
- Ishikawa diagrams are causal diagrams created by Kaoru Ishikawa that show the causes of a specific event. To figure out in which category a defect belongs
- As knows as **cause-and-effect diagrams**,
- Fishbone diagrams





Cause-and-effect diagrams – cont.

- Advantages of using cause-and-effect diagrams
  - Highly visual brainstorming tool which can spark further examples of root causes
  - Quickly identify if the root cause is found multiple times in the same or different causal tree
  - Allows one to see all causes simultaneously
  - Good visualization for presenting issues to stakeholders



Normalization in software metrics

- Normalization is the process of reducing measurements to a standard scale.

- The point of normalization is to make variables comparable to each other.
- Example:
  - How to compare the two?
    - LOC per men = LOC/pm
    - Number of bugs per KLOC = Bugs/KLOC

TABLE 10-1: Attributes for Projects Ruction and Fracas

	PROJECT RUCTION	PROJECT FRACAS
Developers	3	7
Time (months)	1	24
Effort (pm)	$3 \times 1 = 3$	$7 \times 24 = 168$
LOC	1,210	75,930
Bugs	6	462

TABLE 10-2: Normalized Metrics for Projects Ructino and Fracas

	PROJECT RUCTION	PROJECT FRACAS
LOC / pm	$1{,}210 \div 3 = 403$	$75{,}930 \div 168 = 452$
Bugs / KLOC	$6 \div 1.21 = 4.96$	$462 \div 75.93 = 6.08$

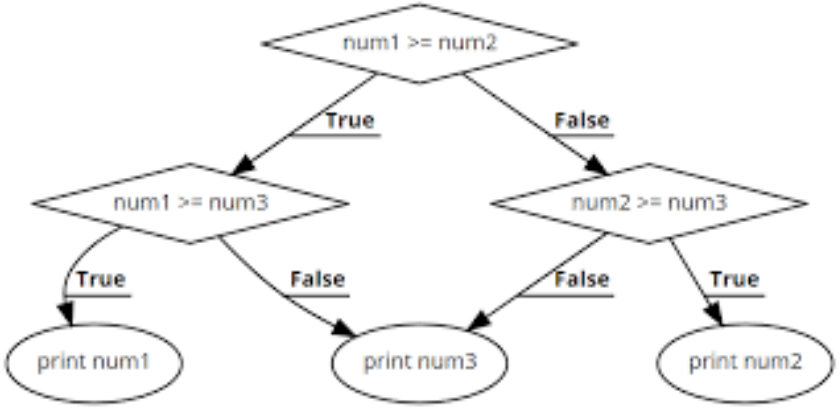
Software complexity

- Software complexity is a technique to describe a specific set of characteristics of software code. These characteristics all focus on how your code interacts with other pieces of code.
  - Scale
  - Diversity
  - Connectivity
  - Dynamics
  - Refinement

- **measure certain aspects of the software (lines of code, # of if-statements, depth of nesting, …)**
- **use these numbers as a criterion to assess a design, or to guide the design**
- **interpretation: higher value ⇒ higher complexity ⇒ more effort required (= worse design)**
- **two kinds:**
  - **intra-modular:** inside one module
  - **inter-modular:** between modules

Software complexity – cont.

- Cyclomatic complexity – proposed by McCabe in 1976
- Cyclomatic complexity measure is a graph driven model that is based on decision-making constructs of program control statements such as if-else, do-while, switch-case and etc.
- The formula of cyclomatic complexity measure is:
  - **V(G) = e – n + 2**
  - Where e is total number of edges; n is total number of nodes

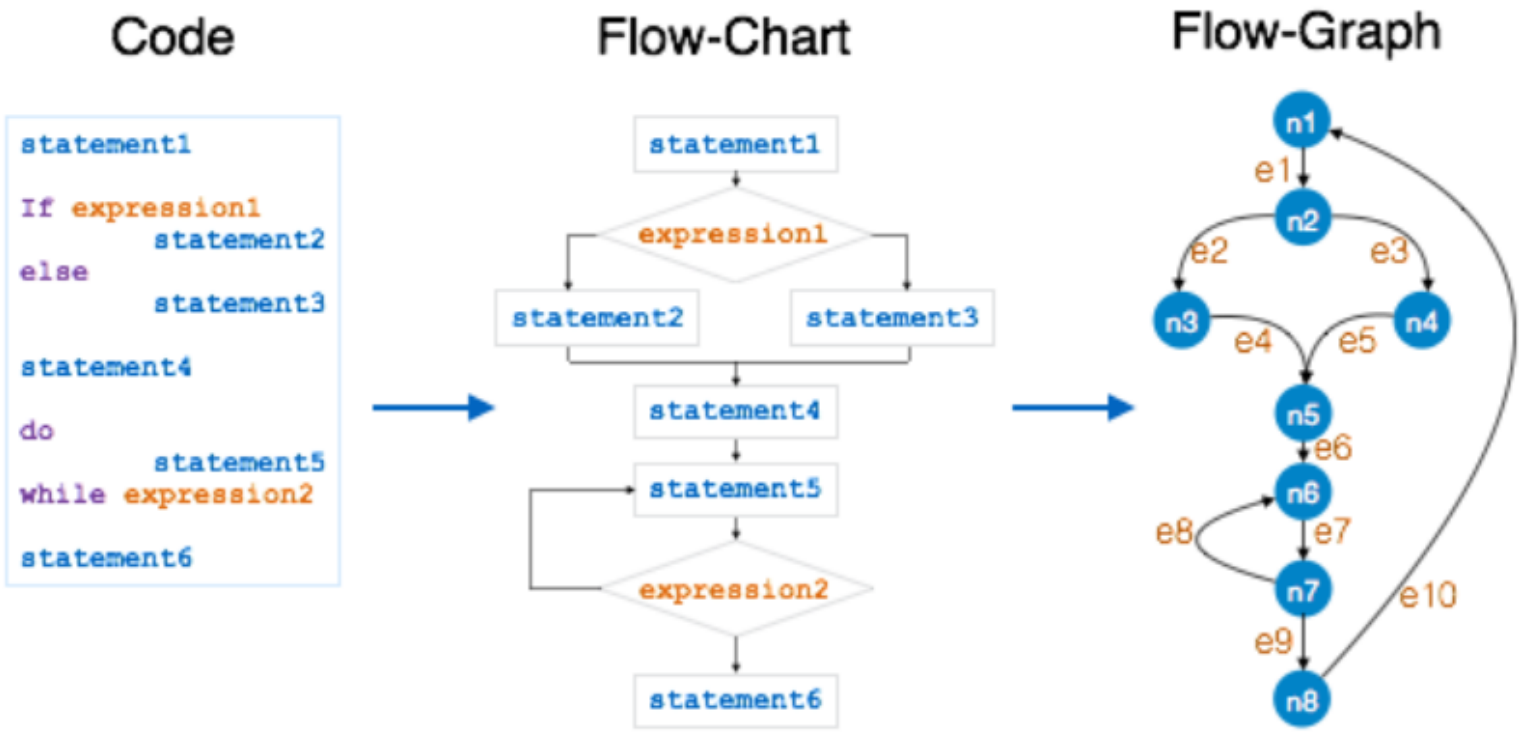


Software complexity – cont.

- Cyclomatic complexity example
  - e = 10
  - n = 8
  - Cyclomatic Complexity

V(G) = 10 - 8 + 2

The larger,
The higher complexity



Function point metrics

- Function point metrics, developed by Alan Albercht of IBM in 1979
- Function point: it is a unit of measurement to express the amount of business functionality that a software system (as a product) provides to a user.
- Function point metrics measure functionality from the users point of view, that is, on the basis of what the user requests and receives in return.
- Function point analysis measures software by quantifying the functionality the software provides to the user based primarily on logical design. The objective of function point analysis is to
  - Measure functionality that the user requests and receives.

Function point metrics – cont.

- Estimate the following items
  - Inputs: the number of times data moves into the software and update the software internal data
  - Outputs: the number of times outputs move out of the software
  - Inquiries: the number of times the software performs a query/response action
  - Internal files: the number of internal logic files used by the software
  - External files: the number of files that the software uses that are maintained by other software.
- Function point = inputs complexity + outputs complexity +inquiries complexity + internal files complexity + external files complexity

Function point metrics – cont.

- Function point calculation example

			Complexity				
Category	Number		Low	Medium	High		Result
Inputs	10	×	3	4	(6)	=	60
Outputs	5	×	(4)	5	7	=	20
Inquiries	4	×	3	(4)	6	=	16
Internal Files	23	×	(7)	10	15	=	161
External Files	2	×	(5)	7	10	=	10
					Total (raw FP)		267

FIGURE 10-5: In this example, the application's raw FP value is 267.

Function point metrics – cont.

- Another example of Function Point calculation

Computing FPs

Measurement Parameter	Count		Weighing factor			
			Simple	Average	Complex	
1. Number of external inputs (EI)	12	*	3	4	6 =	72
2. Number of external Output (EO)	60	*	4	5	7 =	420
3. Number of external Inquiries (EQ)	9	*	3	4	6 =	54
4. Number of internal Data structure (ILF)	6	*	7	10	15 =	90
5. Number of external data structure	3	*	5	7	10 =	30
6. Number of external interfaces	3	*	5	7	10 =	30
7. Number of transformation	36	*			———	36
8. Number of transitions	24	*			———	24
Count-total →						756

function points,the required index is 756.

Summary

- Metrics
  - Concept
- Types of software metrics
  - Process metrics
  - Product metrics
  - Project metrics
- Defect analysis
- Ishikawa diagrams
- Normalization
- Software complexity
  - Cyclomatic complexity measure
- Function points
  - Concept, calculation

Announcement

- Group project
  - Please start draft SDD
  - The due date of SDD is Nov. 22