SOFTWARE QUALITY

CPTS 583

Software Product Quality Metrics and Measurement (II)

-- *Customer metrics, quality attributes metrics, time/cost metrics, complexity metrics*

Outline

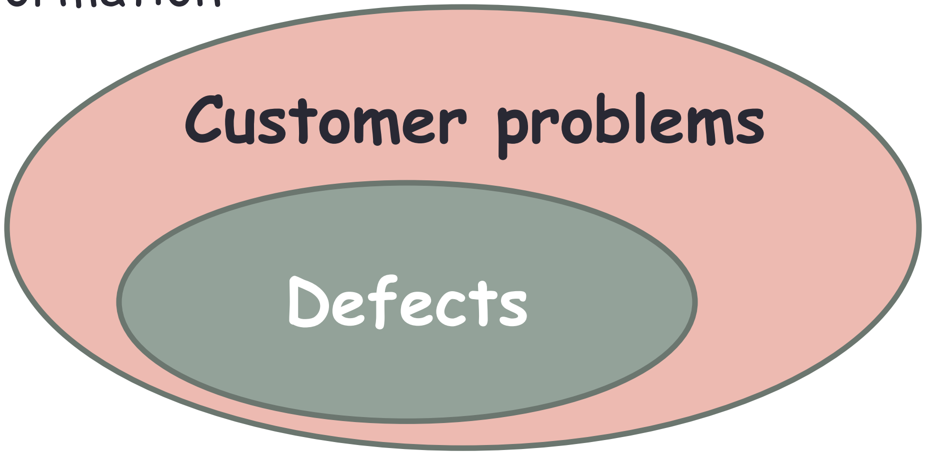
- Customer metrics
  - Problem metric
  - Satisfaction metric
- Quality attributes metrics
  - Maintainability, Integrity
  - Functionality, Reliability, Usability, Availability
- Time/Cost metrics
- Complexity metrics
  - Cyclomatic complexity
  - Halstead complexity

Customer problems

Problems encountered by customers tell about product quality !

- Defect problems
    - Valid defects in product

- Non-defect problems
    - Usability problems
    - Unclear documentation/information
    - Usage/operation errors
    - Other user errors



Customer problem metrics

- Measuring customer problems

  Problems per user month (**PUM**) =

$$\frac{\text{Total problems reported during a time period T}}{\text{Total license-months during T}}$$

- Problems: both defect and non-defect ones
- License-months: # licenses installed x T in months

Customer problem metrics

- Reducing PUM -> improve product quality

Improve development process

Improve usability, documents clarity, training/education

defect problems + # non-defect problems

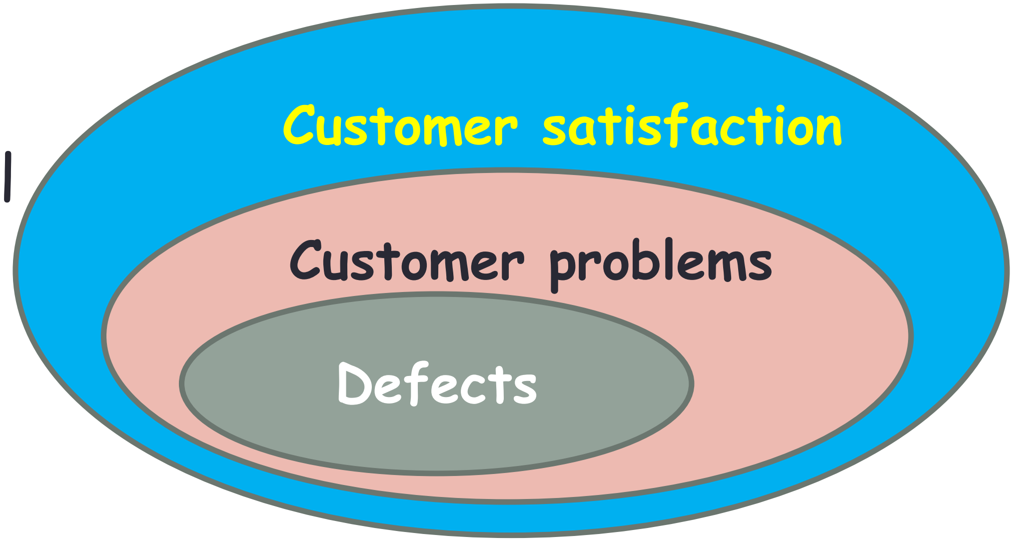
---

licenses installed x # months

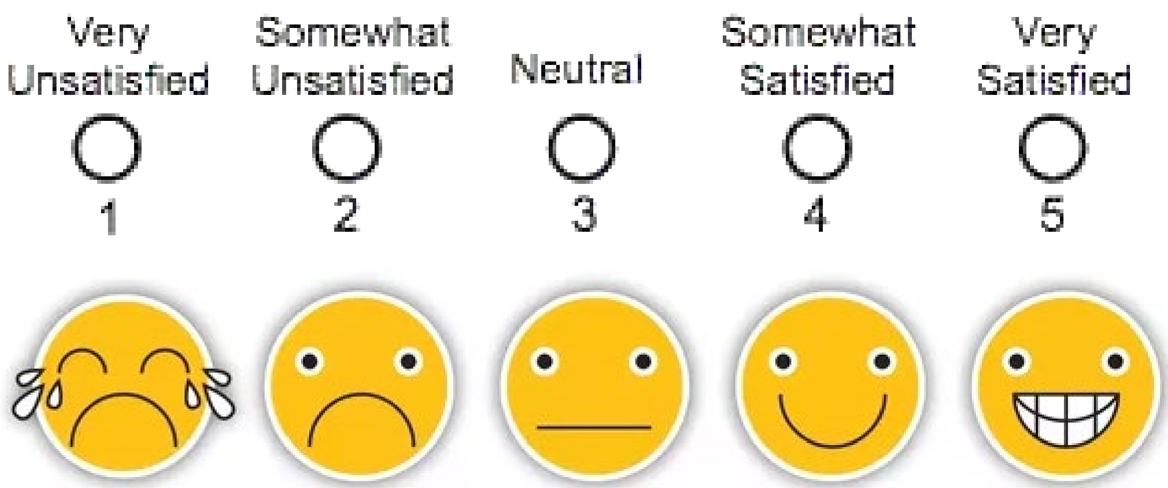
Increase sale

Customer satisfaction metrics

- How customers are satisfied with overall product quality

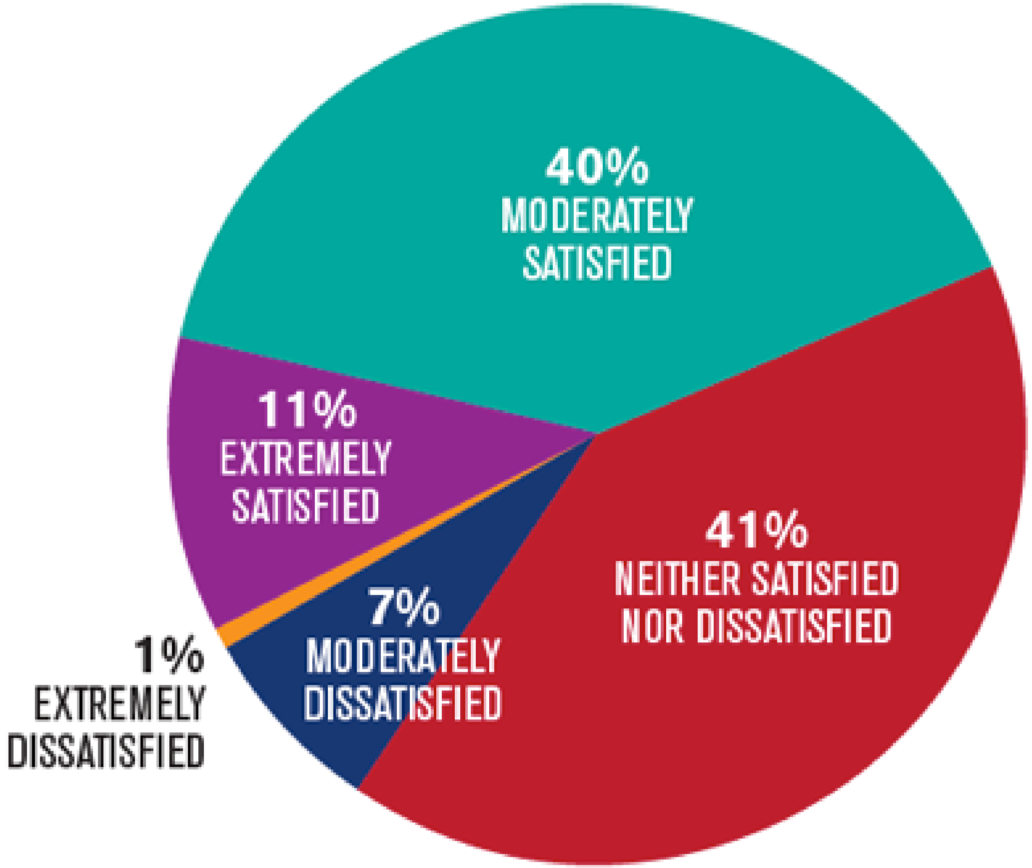


- In a five-point scale:



Customer satisfaction metrics

- Data collection: survey / poll
- Metrics: percentage distribution of ratings



Quality Attributes Metrics

- Maintainability
  - mean time to change (MTTC)
    - the time it takes to analyze the change request, design an appropriate modification, implement the change, test it, and distribute the change to all users
  - spoilage
    - cost of change / total cost of system

- Integrity
  - threat = probability of attack (that causes failure)
  - security = probability attack is repelled

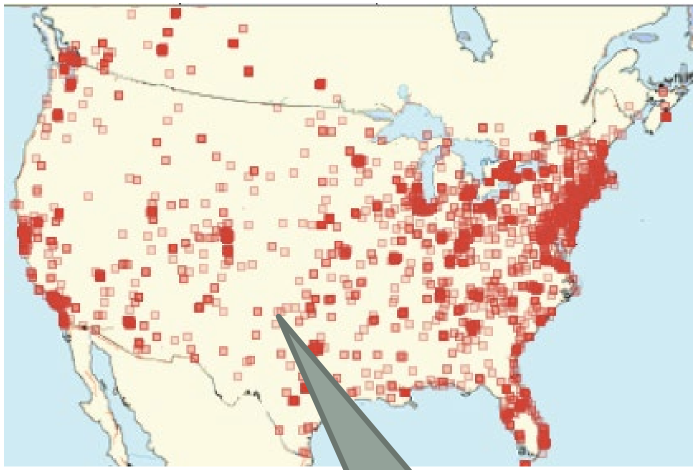
$$\text{Integrity} = \sum [1 - \text{threat} * (1 - \text{security})]$$

Quality Attributes Metrics

- Functionality (functional correctness):
  - Defect density
  - Failure density / severity
- Reliability:
  - Failure rate
  - Mean time between failure (MTBF)
- Usability (user support / learnability / understandability)
  - Help Desk (HD) call density
  - HD severity
  - HD success level
  - HD effectiveness
- Availability

Defect Density

- Ratio-metric which describes how many defects occur for each size/functionality unit of a system
- Can be based on LOC or Function Points (FP).



$$\frac{\# defects}{system\_size}$$

Failure density

- Software system failure density

$$\frac{\#\text{failures in a year}}{system\_size}$$

- # Failures
  - Number of software failures detected during a year of maintenance service
  - Straight total or weighted sum
- System size
  - KLOC or NFP

Failure severity

- Average severity of software system failures (ASSSF)

$$\frac{\text{straight total failures in a year}}{\text{severity weighted total failures in a year}}$$

Weighted number of failures (using weights assigned to different severity levels)

- Example weights

Severity level	Weight
Highly severe	10
Medium severe	5
Low severe	2

Failure Rate

- Rate of failures over time
  - *Versus failure density*

$$\lambda = \frac{R(t_1) - R(t_2)}{(t_2 - t_1) \times R(t_1)}$$

  - **t_1** and **t_2** are the beginning and ending of a specified interval of time
  - **R(t)** is the reliability function, i.e. probability of no failure before time t

Failure Rate

- Example

Calculate the failure rate of system ***X*** based on a time interval of 60 days of testing. The probability of failure at time day 0 was calculated to be ***0.85*** and the probability of failure on day 60 was calculated to be ***0.2***.

$$\lambda = \frac{R(t_1) - R(t_2)}{(t_2 - t_1) \times R(t_1)}$$

$$\lambda = \frac{0.85 - 0.2}{60 \times 0.85}$$

$$= \frac{0.65}{51}$$

$$= 0.013$$ **Failures per day**

Mean Time Between Failure (MTBF)

- Average time period in which a new failure occurs
- Particularly useful in safety-critical applications (e.g., avionics, air traffic control, weapons, etc)

$$MTBF = \frac{1}{\lambda}$$

Mean Time Between Failure (MTBF)

- Example

Consider our previous example where we calculated the failure rate (λ) of a system to be 0.013. Calculate the MTBF for that system.

$$MTBF = \frac{1}{\lambda}$$

$$MTBF$$

$$= 1/\lambda$$

$$= 1/0.013$$

$$= 76.9 \text{ days}$$

This system is expected to fail every 76.9 days.

HD/customer services

- HD: Help Desk (user support)
  - Measure software product usability from customer service perspectives
  - Instruct/train customers about product usage
  - Immediate reflect the quality of
    - User interface
    - Documentation (user manual, help menus, etc.)

- HD metrics
  - Call density / severity

HD call density

- HD calls density metric

$$\frac{\#\text{calls in a service year}}{system\_size}$$

- # Calls
  - Number of HD calls received from customers during a year of customer service
  - Straight total or weighted sum
- System size
  - KLOC or NFP

HD call severity

- Average severity of HD calls (ASHC)

$$\frac{\text{straight total HD calls in a service year}}{\text{severity weighted total HD calls in a service year}}$$

Weighted number of HD calls (using weights assigned to different severity levels)

- Example weights

Severity level	Weight
Highly severe	6
Medium severe	3
Low severe	1

HD service success

- HD success metric (HDS)

Customer problems reported in the HD calls are successfully solved

$$\frac{\#\text{HD calls completed on time in a service year}}{\#\text{HD calls received in a service year}}$$

HD service effectiveness/productivity

- HD effectiveness metric (HDE)

$$\frac{\text{\#hours invested in HD services in a service year}}{\text{\# HD calls received in a service year}}$$

- HD productivity metric (HDF)

$$\frac{\text{\#hours invested in HD services in a service year}}{\text{system\_size}}$$

System availability

Hours during which at least one function is unavailable (failed)

- Full availability (FA)

$$\frac{\#\text{hours system in service} - \#\text{hours service unavailable}}{\#\text{hours system in service}}$$

- Considered in one service year
- Unavailable hours include hours when the system totally failed

Example:

- 7x24 expectation
- 300 hours unavailability including total failure
- FA = (365x24 - 300) / (365x24) = 0.9658

The US government mandates that new air traffic control systems must not be unavailable for more than 30 seconds per year

System availability

Hours during which at least one **vital** function is unavailable (failed), including total-failure time

- Vital availability (VitA)

$$\frac{\#\text{hours system in service} - \#\text{hours vital unavailable}}{\#\text{hours system in service}}$$

- Total unavailability (TUA)

$$\frac{\#\text{hours of total system failure}}{\#\text{hours system in service}}$$

Time/cost metric

- Financial cost
  - Recall: cost of good and bad quality
- Time
  - Recall: COCOMO
  - Man-months / man-hours

- In synergy with other metrics
  - Size metrics
    - E.g., time/cost per function point
  - Quality attributes metrics

Software product complexity

- Complexity is an important attribute to measure
- Measuring complexity helps us
  - Predict testing effort
  - Predict defects
  - Predict maintenance costs

- Cyclomatic complexity
  - indicates a program's testability and understandability
  - Measures the number of linearly independent paths comprising the program

- Halstead complexity
  - Syntactic units as tokens
  - Statistics of tokens

Cyclomatic complexity

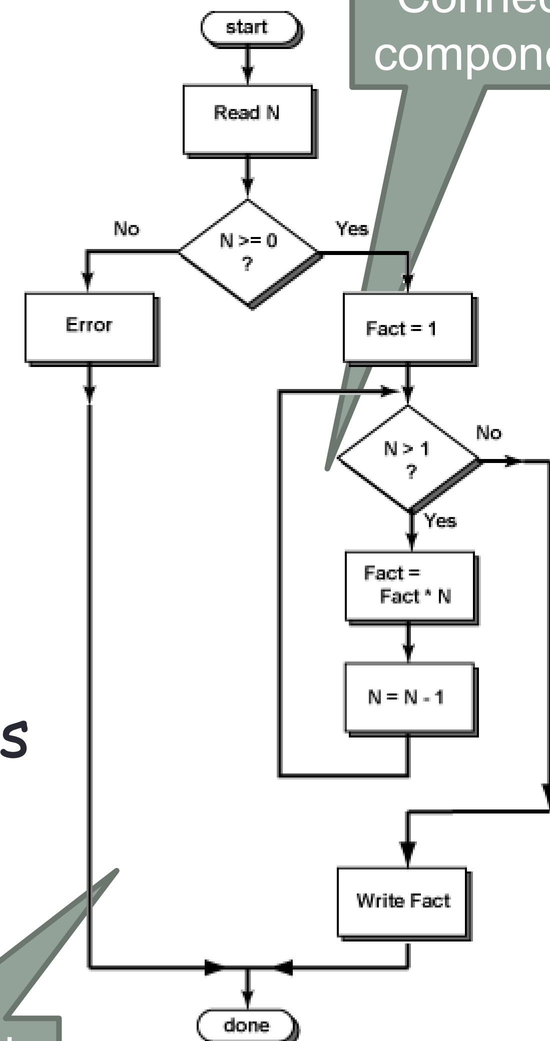
$$M = V(G) = e - n + 2p$$

V(G) = cyclomatic number of Graph G

e = number of edges

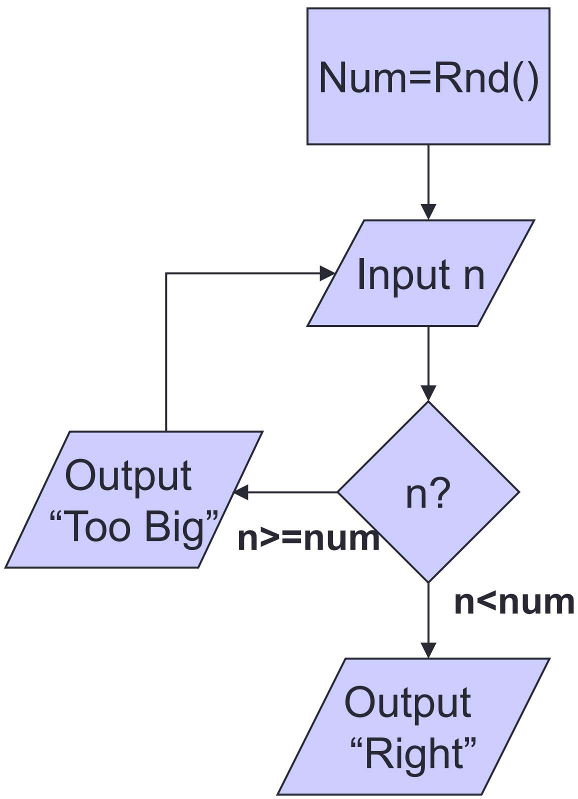
n = number of nodes

p = number of connected components of the graph



Cyclomatic complexity

Consider the following flowchart…



<u>Calculating cyclomatic complexity</u>

$e = 5$, $n=5$, $p=1$

$M = 5 - 5 + (2 \times 1) = 2$

Cyclomatic complexity

- M = #independent execution paths
- $M \le 10$ for good testability and maintainability
- Cyclomatic Complexity is additive

  $$M(G_1 \text{ and } G_2) = M(G_1) + M(G_2)$$

- Adopted in industry

Halstead complexity

- Halstead's software science
  - **Premise:** Any programming task consists of selecting and arranging a finite number of program "tokens"
  - Tokens are basic syntactic units distinguishable by a compiler
  - Computer Program: A collection of tokens that can be classified as either operators or operands

Halstead complexity

- Metrics Primitives
  - n_1 = # of distinct operators appearing in a program
  - n_2 = # of distinct operands appearing in a program
  - N_1 = total # of operator occurrences
  - N_2 = total # of operand occurrences
- Metrics
  - Based on the four primitives

Halstead complexity

Vocabulary (n) $n = n_1 + n_2$

Length (N) $N = N_1 + N_2$

Volume (V) $V = N \log_2(n)$ ← #bits required to represent a program

Level (L) $L = V^* / V$ ← Measure of abstraction and therefore complexity

Difficulty (D) $D = n1/2 * N2/n2$

Effort (E) $E = D * V$

Faults (B) $B = E^{2/3}/S^*$

Where:

$V^* = 2 + n_2 \times \log_2(2 + n_2)$

S^* = average number of decisions between errors (3000 according to Halstead)

Halstead complexity

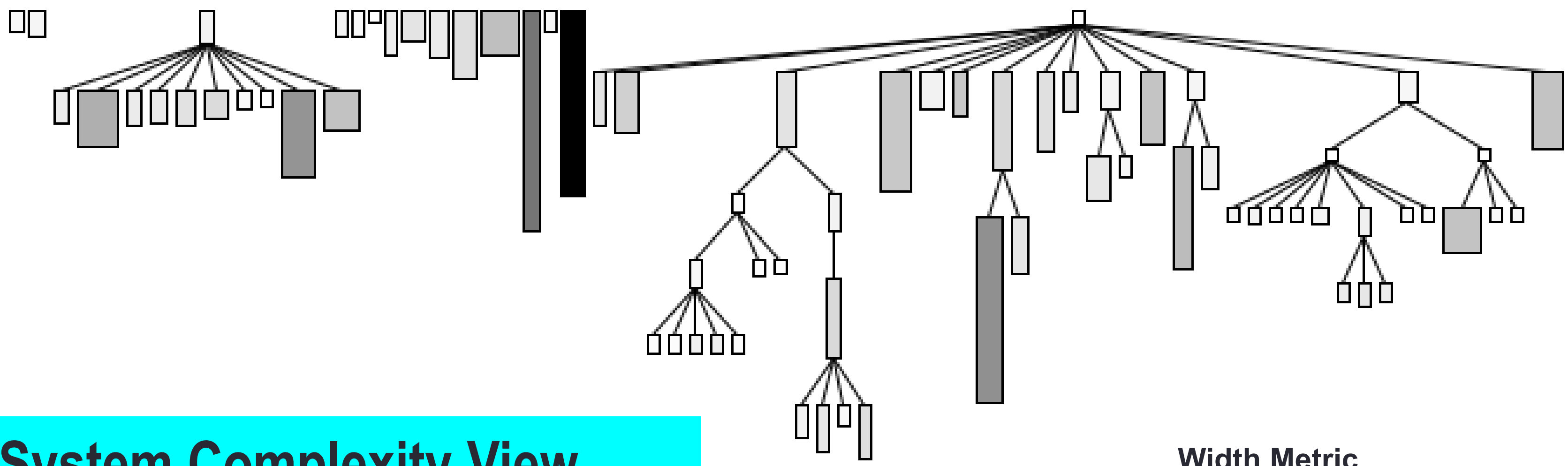
```
Z = 20;
Y = -2;
X = 5;

While  X>0

  Z = Z + Y;

  if  Z > 0 then
     X = X – 1;
  end-if;

End-while;

Print(Z);
```

OPERATOR	COUNT	OPERAND	COUNT
IF-Then- end if	1	Z	5
While End-While	1	Y	2
=	5	X	4
;	8	20	1
>	2	-2	1
+	1	5	1
-	1	0	2
print	1	1	1
( )	1		

$n_1 = 9$ $N_1 = 21$ Length: $N = 21+17 = 38$

$n_2 = 8$ $N_2 = 17$ Volume: $V = 38 \log_2(17)=155$

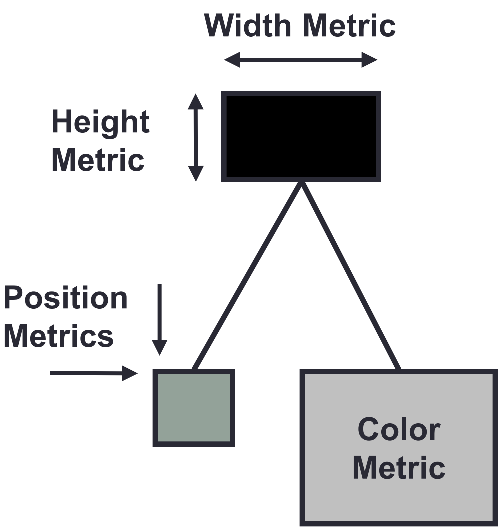
System complexity view



System Complexity View

Nodes = Classes
Edges = Inheritance Relationships

Width = Number of Attributes
Height = Number of Methods
Color = Number of Lines of Code



Summary

- Measuring product quality from customer's perspectives
  - Customers' problems and satisfaction metrics (PUM, five-point ratings and rating category distribution)
- Measuring product quality with respect to quality attributes
  - Maintainability (MTTC, spoilage), Integrity (threat, security)
  - Functionality (defect/failure density)
  - Reliability (failure rate, MTBF)
  - Usability (HD call density, severity, success, effectiveness)
  - Availability (full availability, vital availability, total unavailability)
- Measuring time/cost
- Measuring complexity
  - Cyclomatic / Halstead's complexity metrics