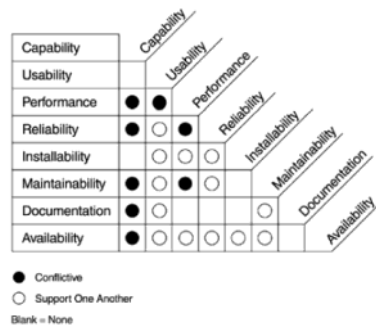


Book 2: until p. 315

Saturday, September 17, 2022 5:37 PM

Interrelationships of Software Attributes - CUPRIMDA



Ratio: ratio results from dividing one quantity by another. two distinct populations and are mutually exclusive.

$$r = \frac{\text{no. males}}{\text{no. females}} \times 100\%$$

Proportion: Proportion is different from ratio in that the numerator in a proportion is a part of the denominator.

$$p = \frac{a}{a+b}$$

Percentage Distribution of Defect Type by Project

Type of Defect	Project A (%)	Project B (%)	Project C (%)
Requirements	115.0	141.0	120.3
Design	125.0	121.8	122.7
Code	150.0	128.6	136.7
Others	110.0	118.6	120.3
Total	100.0	100.0	100.0
(N)	(200)	(105)	(128)

Rate: is associated with the dynamics (change) of the phenomena of interest; measure of change in one quantity (y) per unit of another quantity (x) on which the former (y) depends. Usually the x variable is time:

$$\text{Crude Birth Rate} : \frac{B}{P} \times K$$

$\xrightarrow{\text{births}}$
 $\xrightarrow{\text{population}}$

constant, usually 1000

Product Quality Metrics

- Intrinsic product quality is usually measured by the number of "bugs" (functional defects) or by how long the software can run before encountering a "crash."
- the two metrics are defect density (rate) and mean time to failure (MTTF).
- when an error occurs during the development process, a fault or a defect is injected in the software.
- The probability of failure associated with a latent defect is called its size, or "bug size."

1. We can use the number of unique causes of observed failures to approximate the number of defects in the software:

$$\frac{\text{number of defects}}{\text{size of software (LOC)}}$$

For app software, most defects are found in two years of its release.

Lines of Code Defect Rates

LOC counts are obtained for the total product and for the new and changed code of the new release.

- LOC count is based on source instructions, the two size metrics are called shipped source instructions (SSI) and new and changed source instructions (CSI).
- defect rate metrics per thousand SSI (KSSI) or per thousand CSI (KCSI) are: (1) Total defects per KSSI (a measure of code quality of the total product)

$$\text{KCSI} = \text{KSSI} = 50 \text{ KLOC}$$

$$\text{Defects/KCSI} = 2.0$$

$$\text{Total number of defects} = 2.0 \times 50 = 100$$

2nd release

$$\text{KCSI} = 20$$

$$\text{KSSI} = 50 + 20 \text{ (new and changed lines of code)}$$

$$\text{(assuming 20\% are changed lines of code)} = 66 \text{ Defect/KCSI} = 1.8$$

$$\text{(assuming 10\% improvement over the first release) Total number of additional defects} = 1.8 \times 20 = 36$$

} 1st release

Customer Problems Metric

Metric used measures the problems customers encounter when using the product.

- From the customers' standpoint, all problems they encounter while using the software product, not just the valid defects, are problems with the software
- Even usability problems, unclear documentation or information.
- The problems metric is usually expressed in terms of problems per user month (PUM). PUM is usually calculated for each month after the software is released to the market.
- Reduce the non-defect-oriented problems by improving all aspects of the products (such as usability, documentation), customer education, and support.

Table 4.1. Defect Rate and Customer Problems Metrics

	Defect Rate	Problems per User-Month (PUM)
Numerator	Valid and unique product defects	All customer problems (defects and nondefects, first time and repeated)
Denominator	Size of product (KLOC or function point)	Customer usage of the product (user-months)
Measurement	perspective Producer—software development organization	Customer
Scope	Intrinsic product quality	Intrinsic product quality plus other factors

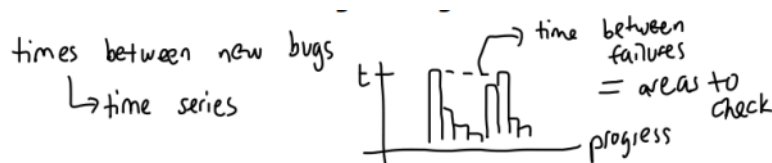
Defect Density During Machine Testing

Defect rate during formal machine testing positively correlated with the defect rate in the field.

- Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process
- metric of defects per KLOC or function point, therefore, is a good indicator of quality while the software is still being tested
- If the defect rate during testing is the same or lower than that of the previous release (or a similar product), then ask: Does the testing for the current release deteriorate?

Time Line: defect rate when tested
 control diagram
↓
by release
(feature)

- The pattern of defect arrivals (or for that matter, times between failures) overall defect rate during testing, different patterns of defect arrivals indicate different quality levels in the field.



- The objective is always to look for defect arrivals that stabilize at a very low level, or times between failures that are far apart, before ending the testing effort and releasing
- defect arrivals (defects reported) during the testing phase by time interval (e.g., week).
- The pattern of valid defect arrivals—when problem determination is done on the reported problems. This is the true defect pattern.

defect arrival customer problems → may not be
time time defects

Metrics for Software Maintenance

When development of a software product is complete and it is released to the market, it enters the maintenance phase of its life cycle.

- defect arrivals by time interval and customer problem calls by time interval are the de facto metrics.
- What can be done during the maintenance phase is to fix the defects as soon as possible and with excellent fix quality. This, although not able to improve the defect rate, can improve customer satisfaction.

Metrics:

1. Fix backlog is a workload statement for software maintenance. Is the maintenance work to do. Fix Backlog Index: reported problems that remain at the end of week/time frame. Make this a trend chart (control chart) and gives info for managing the maintenance process.
2. BMI (Backlog Management Index): Another metric to manage the backlog of open, unresolved, problems:

$$BMI = \frac{\text{num of problems closed in a month}}{\text{total num of problems}} \times 100\%$$

backlog of open, unresolved, problems:

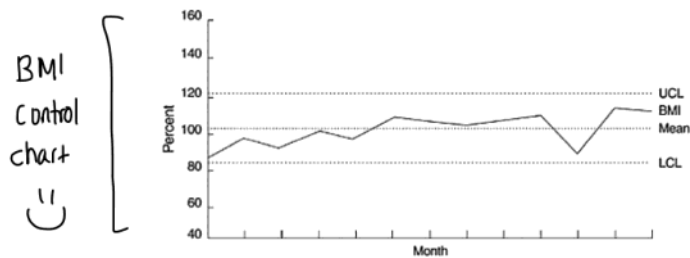
$$BMI = \frac{\text{num of problems closed in a month}}{\text{num of problem arrivals in a month}} \times 100\%$$

↓
time series = capability of maintenance

- if BMI is larger than 100, it means the backlog is reduced. If BMI is less than 100, then the backlog increased.
- the techniques of control charting can be used to calculate the backlog management capability of the maintenance process

Example: control chart for BMI. The mean BMI was 102.9%, indicating that the capability of the fix process was functioning normally

- All BMI values were within the upper (UCL) and lower (LCL) control limits—the backlog management process was in control.



Fix Response Time and Fix Responsiveness

- the time limit within which the fixes should be available for the reported defects, in accordance with the severity of the problems.
- less severe: , the required fix response time is more relaxed. The fix response time metric: Mean time of all problems from open to closed.
- If there are data points with extreme values, medians should be used instead of mean. short fix response time leads to customer satisfaction.

example, the NASA Software Engineering Laboratory spent about 15% of their development costs on gathering and processing data on hundreds of metrics. DO NOT MEASURE EVERYTHING.

Example: Inspection Summary Form

- records the total number of inspection defects and the LOC estimate for each part (module), as well as defect data classified by defect origin and defect type.
- The following pertains to the defect type classification by development phase:

Product: _____ Component: _____ Release: _____

Inspection Type: _____ (RQ SD IO I1 I2 U1 U2)

Description: _____

Defect Counts										
Tot for Inspection		By Defect Origin					By Defect Type			Total
/ Part Name	LOC	RQ	SD	IO	I1	I2	LO	IF	DO	

Description: _____

[illegible]

- Defect is the way two pieces of logic communicate
 - ↳ two subroutines
 - ↳ messages and panels
- Defect in the description of a function that causes someone to do something wrong.

Recommendations for Small Organizations

1. Implement the defect arrival metric during the last phase of testing, together with number of critical problems (or show stoppers) and the nature of these problems. The latter is a subset of total defects. This brings information with regard to its readiness to ship to customers.
2. After the product is shipped, a natural metric is the number of defects coming in from the field over time: how big is the fix backlog. Calculate the backlog management index metric.
 - maintenance strategy: individual fixes, fix packages, or frequent software upgrades.
3. A product size metric is available as defect density curves over time, during testing or when the product is in the field.
4. When enough empirical data is accumulated, **correlational studies** between in process data and field data of the organization can improve the predictability and consistency of product development

Maintenance Metrics

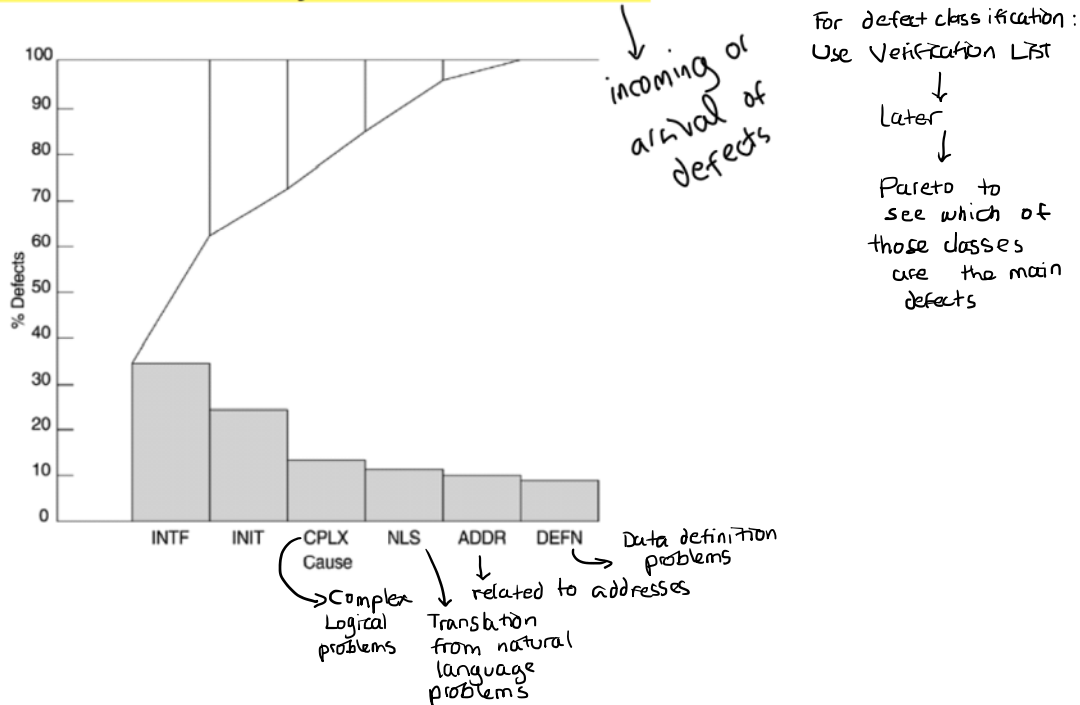
- Fix backlog
 - Backlog management index
 - Fix response time and fix responsiveness
- } do this

Applying the 7 Quality Tools to Software Development

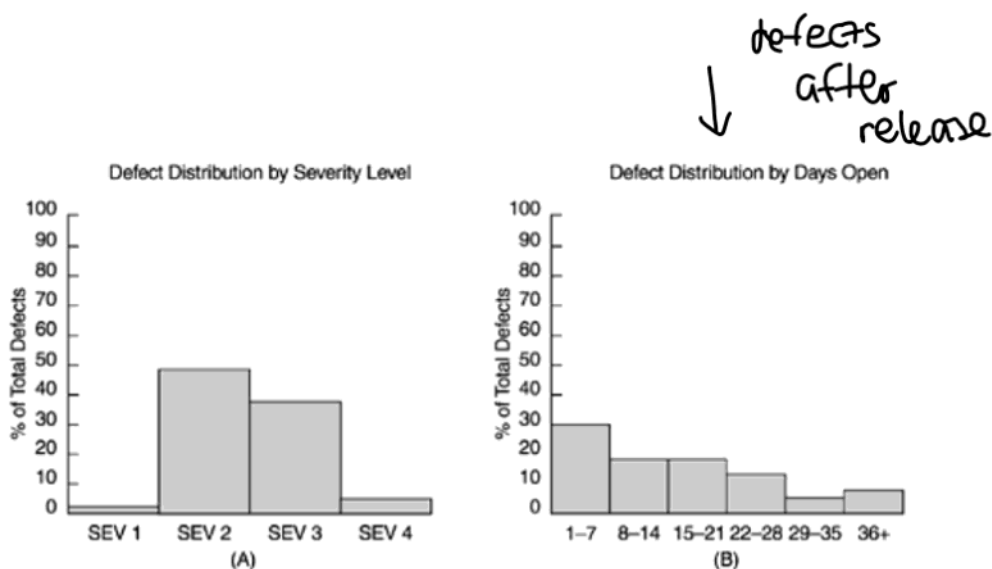
1. Pareto Diagram

- software defects or defect density never follow a uniform distribution: there are always patterns of clusterings—defects cluster in a minor number of modules or components, a few causes account for the majority of defects, some tricky installation problems account for most of the customer complaints.
- Example: Pareto analysis of the causes of defects for an IBM Rochester product. Interface problems (INTF) and data initialization problems (INIT) were found to be the dominant causes for defects. By focusing on these two areas: improvement was observed.

Figure 5.3. Pareto Analysis of Software Defects



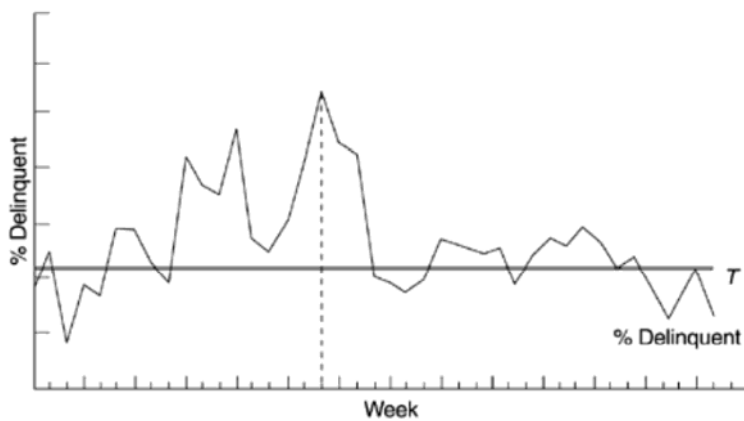
2. Histogram



3. Run charts

For example, the weekly arrival of defects and defect backlog during testing phases can be monitored via run charts.

- real-time statements of quality as well as workload.
- Example: For each delinquent defect report, causal analysis was done and corresponding actions implemented.



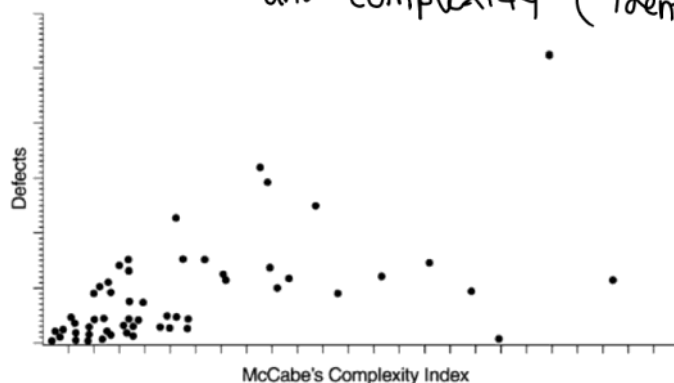
What to do: Causes and actions to Reduce Delinquent Fixes

What to do

Cause	Increase Delinquency Awareness	Delinquency Highest Priority	Work Load No Excuse	Emphasis on Complete and Quality Fix	Use Test Fix Provide Circumvention	Internal Screen Team/APAR Coordinator	Active Communication to Secure Info.	Guideline for Awaiting Customer Info.	Internal APAR Procedure	World Trade APAR Routing	New Function APAR Process
New Function											X
Complex Problem or Fix			X	X	X						
Procedural Problem	X						X	X	X		
Not Reproducible					X						
Waiting for Cust. Response						X	X				
Serialized on Other Fix			X								
Received Late		X									
Waiting on Ext. IBM Group						X	X		X		
Work Load	X	X	X		X						
Mishandled	X	X			X			X	X		
IBM Internal		X						X			

4. Scatter Diagrams

scatter of number of bugs
and complexity (identification?)



5. Control Charts

Software differs from manufacturing in several aspects and such differences make it very difficult, if not impossible, to arrive at useful estimates of the process capability. Software is design and development, not production.

- Therefore, the life-cycle concept is more applicable to software than control charts.

The most approximate charts for software applications are perhaps the p chart, when percentages are involved, and the u chart, when defect rates are used.

- The control limits are calculated as the value of the parameter of interest plus/minus three standard deviations.
- a pair of warning limits, which are normally calculated as the value of the parameter plus/minus two standard deviations.
- For example, control limits for defect rates (u chart) can be calculated as follows:

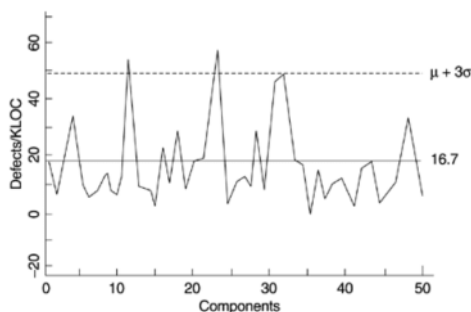
$$\text{Upper Limit} = \bar{\mu} + 3\sqrt{\frac{\bar{\mu}}{n_i}} \quad \text{Lower Limit} = \bar{\mu} - 3\sqrt{\frac{\bar{\mu}}{n_i}} \quad \text{where}$$

$\bar{\mu}$ = value of the center line : cumulative defect rate (weighted average of defect rates) across the subgroups

n_i = the size of subgroup i for the calculation of defect rate.
(LOC)

Subgroups : program modules, components, design segments.

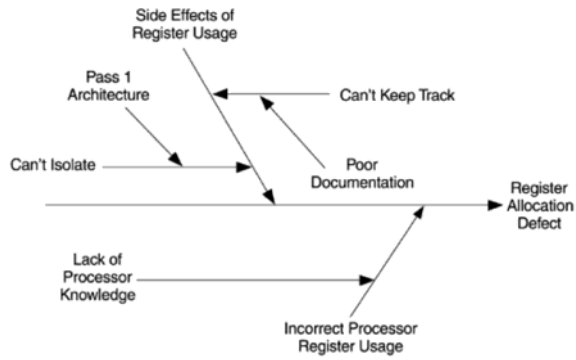
Figure 5.12. Pseudo-Control Chart of Test Defect Rate—First Iteration




Second Iteration: the previously identified error-prone components were removed and the data were plotted again, with a new control limit

6. Cause-and-Effect Diagram

After Pareto diagram and found that defects associated with register allocation were the most prevalent. Then, a cause-and-effect diagram, they conducted brainstorming sessions on those problems.



Summary:

- Lists of verification → classify problems into 5 classes
- Pareto → find main prevalent problems
 -  } choose two ↓
- fishbone → find prevalent problems' causes
- BMI control chart → how to maintain after release (what to do)