SWEN 772-Software Quality Engineering

W3-1 Software Metrics Overview & 7 Quality Tools

Measurements & Metrics

- Measurements: Raw numbers
- Metrics: (Usually) derived/computed numbers that:
  - Indicate **the extent to which** some objective is being achieved
  - Facilitate **cross-comparison**
  - Can serve as **the basis for actions** to improve achievement of the objective
- Identifying **useful metrics is hard work!**
  - Many times, we can’t find any for some objectives
  - If so, use subjective evaluations

Some Measurements for Software

- **Size:** Function points, story points
- **Time and effort** for different project activities
- **Defects found**, classified by phase/increment occurred, phase/increment found, module, type, severity
- **Failures** and when they occurred
- **Staffing, requirements changes, customer satisfaction** (survey results), etc.

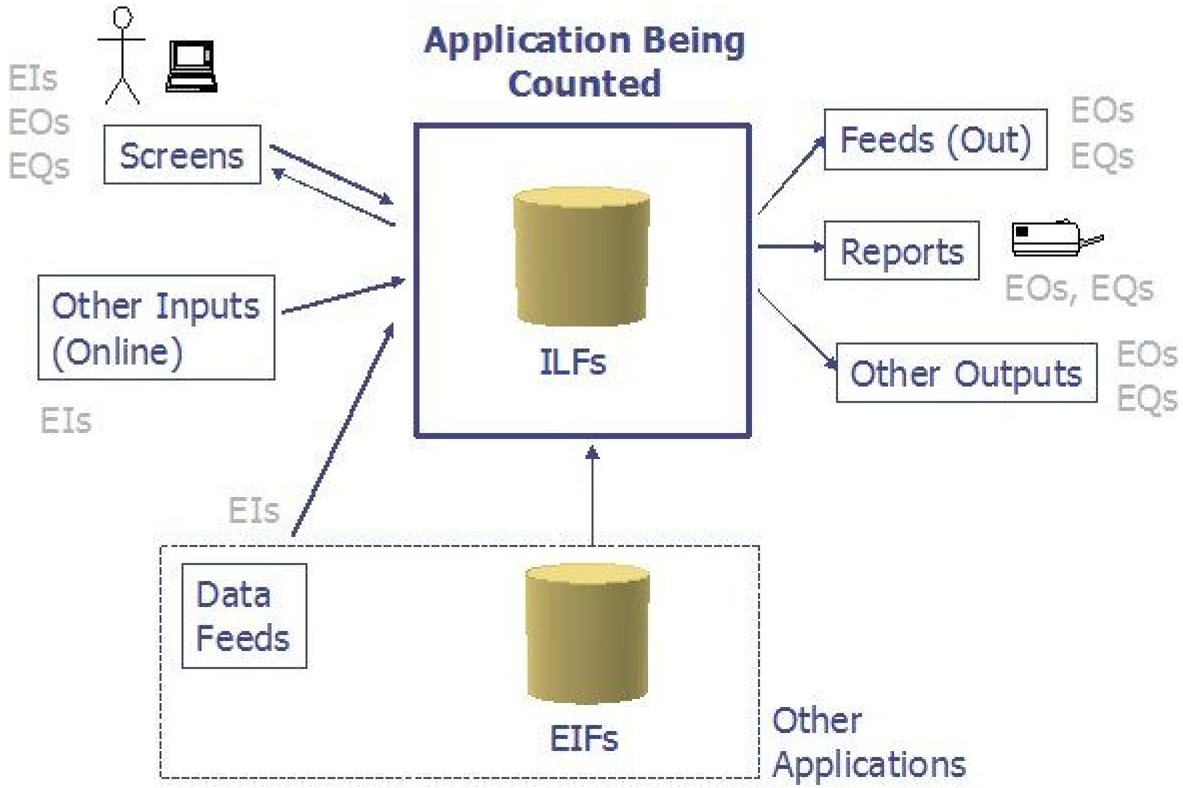
Lines of Code

- Physical
  - How many lines of text?
    - With or without comments
    - With or without whitespace
- Logical
  - Attempt to count executable statements
  - for (i = 0; i < 10; i++) printf("hello"); /* How many loc is this? */

Function Points

- Function point metric uses five major (weighted) components to obtain its value:
- -Number of external Inputs x 4 (EIs)
- -Number of external outputs x 5 (EOs)
- -Number of external inquiries x 4 (EQs)
- -Number of internal logical files x 10 (ILFs)
- -Number of external interface files x 7 (EIFs)

Function Points (Cont)



src: https://alvinalexander.com/FunctionPoints/FPArchitecture2.jpg

Function Points (Cont)

- These weights may be adjusted higher or lower if the complexity of the software is particularly low or high. The values above are average complexity.
- These are used to calculate function counts (FCs). Function count is derived as follows:
- $FC = (\Sigma (\Sigma W_{ij} * X_{ij}))$ -- where W_{ij} are the weighting factors above, and X_{ij} is the numbers of each component in the application.

14 GSC's

- The second step is to evaluate 14 general system characteristics and rate on a scale from 0 to 5 based on
- -Distributed functions
- -Performance
- -Heavily used config
- -transaction rate
- -online data entry
- ...etc

VAF and Final Calculation

- VAF = 0.65 + 0.01 $\sum C_i$ -- where Ci is the score for each individual general system characteristic (the 14 things above).
- .65 is a weighting value. When GSCs are low, .65 is used. When GSC are high, 1.35 is used. The .01 is to put everything on the same scale (or, at least, that's what it appears to do. No source seems to specify what it does.)
- The final function point value is:
- FP = Function Count (FC) x Value Adjustment Factor(VAF)

Metrics for Software

- **Product** Metrics
  - Indicate the quality of the product produced
- **Project** Metrics
  - Indicate whether process execution (business aspects) are on track
- **In-Process** Metrics
  - "Barometers" or "dashboard" to indicate whether the process appears to be "working normally"
    - Allows making changes while there is still a chance to have an impact on the project
  - Useful during the development and maintenance process to identify problems and areas for improvement

Software Metrics – Things to Consider

- As you see each metric, think about:
  - How **useful** is it? How would this be used?
  - How **meaningful** is it?
  - How **easy** is it to gather? How much extra work is it for developers to generate the numbers?
  - Are there ways to "**beat / defeat**" this metric?
    - Can you "make it look good" in ways that don't achieve the objectives?
  - What other metrics do you need to get **a balanced picture**?

Product Metrics

- Performance
  - Lots of measurements, lack of good metrics
- Reliability
  - Defect density: Defects per KLOC (KLOC: 1000 lines of code)
  - Failure intensity: Number of failures per (hour of) operation
- Availability
  - Uptime %

Product Metrics - Continued

- Usability
    - SUMI score: user survey results, relative to “state-of-the-art”
- Evolvability, safety, security
    - Metrics are more like measurements, value as indicators debatable
- Overall
    - Customer satisfaction: results of customer surveys
    - Customer reported defects: defect reports per customer-month

Defect Density (Example)

- Shipped Source Instructions
  - SSI = SSI (prev release)
  - + CSI (new and changed source instructions)
  - - Deleted code
  - - Changed code (avoid double count)
- Total Defects/KSSI

Customer Perspective

- Customer Problems Metric -- Problems Per User Month(PUM)
- PUM =
  - Total problems reported (true and non-defect-oriented) for month
  - ***div*** Total Number of License-months of software during period
- What might be a weakness here?
- Is there a way to “break” this?

Project Metrics

- Cycletime
  - Elapsed time from requirements to delivery
  - Sprint length, epoch length
- Productivity
  - Size of delivered software / total effort
  - Any weaknesses?
- Rate of Requirements Change
  - % of requirements that changed plotted vs. time
  - Any weaknesses?

Project Metrics - Continued

- Estimation Accuracy
  - % difference between estimated and actual
  - Can be done for cycletime (completion date), effort
- Staffing Change Pattern
  - % of turnover (entered, left) plotted vs. time
  - High staffing change will impact productivity, quality

In-Process Metrics

- Tracking metrics during a project (“in-process”) provides a powerful monitoring and control tool
  - Ensure that quality is in control
  - React quickly to understand and respond to observed variations

In-Process Metrics: Defects, Reliability

- **Reliability growth pattern**
  - Failures during system testing plotted vs. time
  - Expected: spikes during each release, decrease over time
  - Magnitude of spike related to significance, volume of changes
- Pattern of **defects found** (arrivals) during testing
  - Test defects found plotted vs. time during testing
  - Should decrease significantly close to release
  - Can project “latent defects” (defects left at release) from this
- **Defect density**
  - Defects per KLOC (can be classified by type, module)
    - Highlights “hot spots”
  - Post-release defect density
    - Strong indicator of effectiveness of testing

In-Process Metrics: Maintenance

- Backlog Management Index
  - Rate of problem arrivals / rate of closure
    - Should be close to 1, at least for high severity
- Responsiveness of fixing
  - Average closure time, age of open & closed problems, % late fixes
    - Should stay within target values
- Fix quality
  - Number and % of defective fixes (didn’t work or created new bugs)

In-Process Metrics: Management

- Cost of Quality (CoQ)
  - **Total effort on quality assurance activities**: testing, reviews, procedures
  - Should be as low as possible
- Cost of Poor Quality (CoPQ) -- Technical Debt
  - **Total effort expended on rework**
  - Should be within range (what if it is “too low” -- isn’t that great?)

In-Process Metrics: Management (Continued)

- Phase/sprint containment effectiveness / **defect removal effectiveness**
  - What % of the errors were detected within that phase/sprint?
    - Shows effectiveness of reviews and other quality procedures
  - Preferably around 70% or so
    - If it is 97%, is that good?
- Note: Containment effectiveness can also be applied to incremental development
  - Increment containment effectiveness

Summary

- There are a number of metrics that can give a meaningful picture of what is going on in a project
  - There are **metrics** that can help to **identify problems and areas of improvement** (in-process and post-mortem), as well as metrics that **evaluate results**
  - We need to **think carefully** about what the metrics indicate about the process and product quality
- By designing a quality program that uses **multiple metrics** in conjunction with each other, we can get a **balanced picture**
- Most of the metrics come from **relatively little raw measurement data**: size, effort, defects / failures, timeline data

The Seven Basic Tools

- Checklists (Checksheets)
- Pareto Diagrams
- Histograms
- Run Charts
- Scatter Diagrams (Scatter Plots)
- Control Charts
- Cause-and-Effect (Fishbone) Diagrams

What Are These Tools?

- Simple techniques to:
  - Track quality performance and trends
  - Identify the existence of quality problems
  - Analyze and gain insights into the causes and sources of quality problems
  - Figure out which problems to address
  - Help eliminate quality problems
    - Defect prevention, not just detection and correction
- Basic knowledge for anyone interested in quality, engineering problem solving, and systems design
  - Probably already familiar with most of these

Why Exactly Seven Tools?

- Kaoru Ishikawa promoted the notion of seven basic tools that could be used to address quality
  - Designed for manufacturing environments, but applicable to engineering & management, too
- There are other very useful tools:
  - Templates, workflow automation
  - Pie charts and other graphical representations
  - Relationship diagrams, tree diagrams, etc. ("Seven new quality tools")
  - System dynamics diagrams and influence diagrams
- We learn a basic subset here, others left to "lifelong learning"
  - Corporate training often introduces/uses quality tools & techniques
  - See the American Society for Quality (http://www.asq.org/)

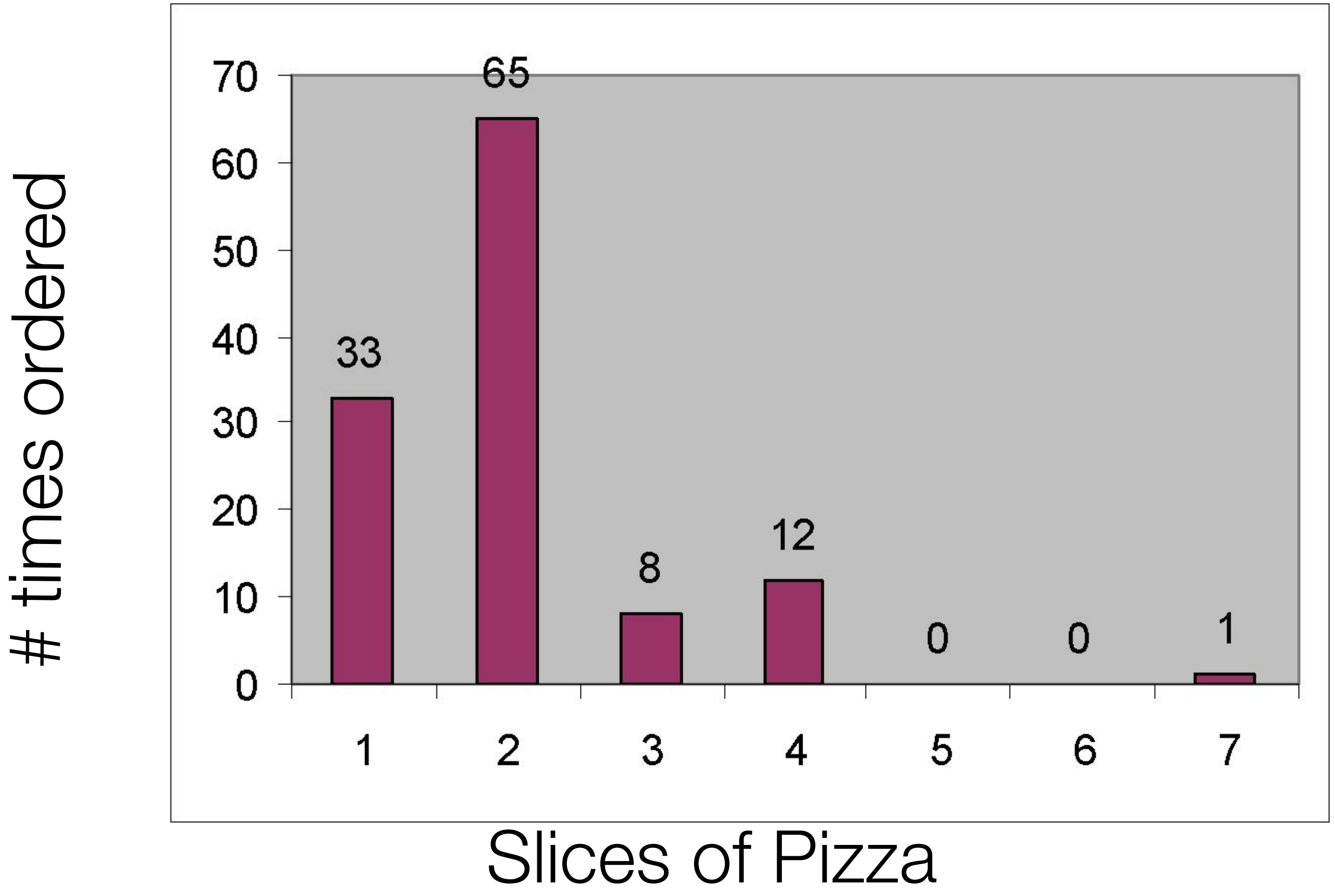
What to Learn About Each Tool

- What is the tool?
- How is it used?
- For what purposes is it useful?
- What value does it add?
- What are its limitations?
- How can it be used effectively?

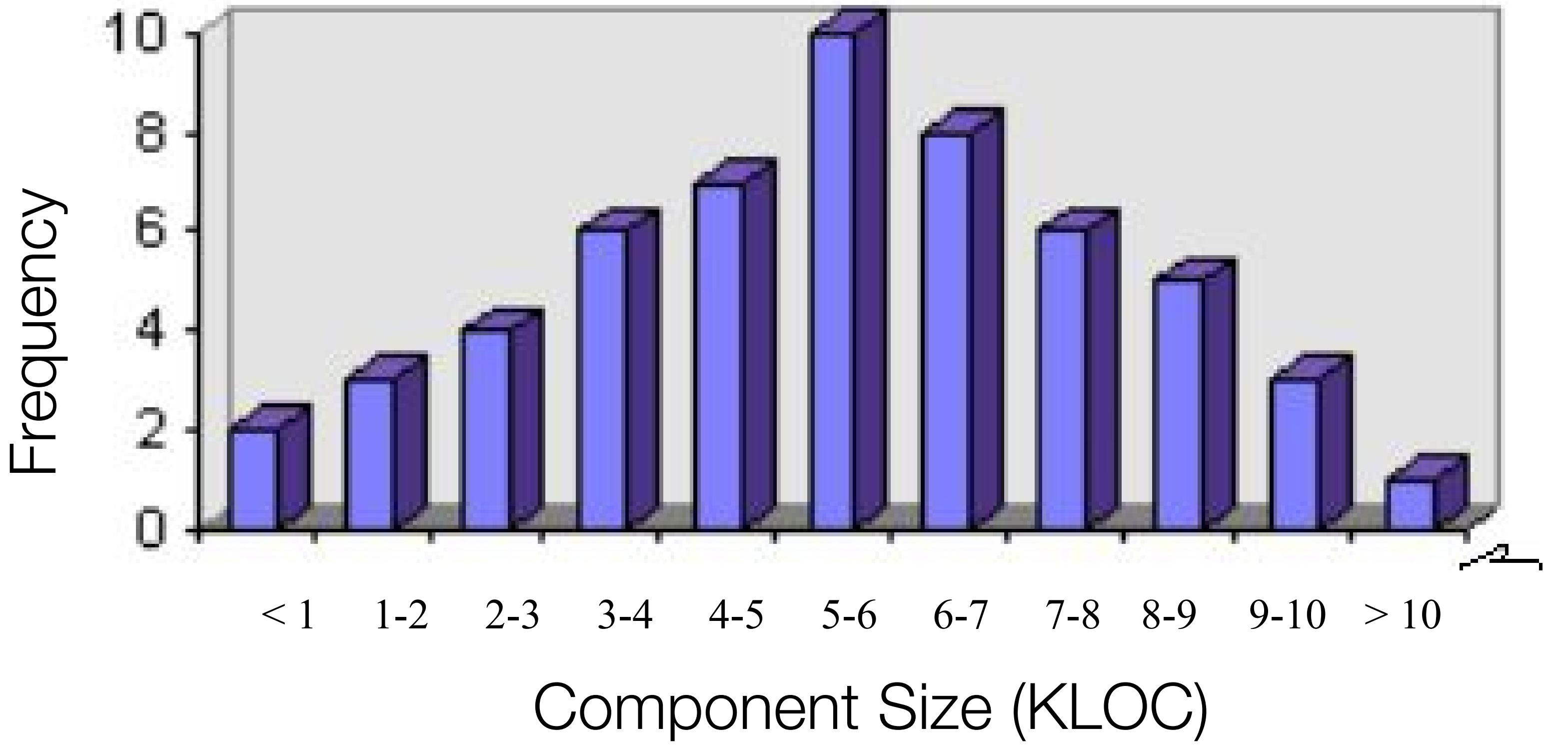
Histogram

- A bar graph showing **frequency counts**
- X axis often a nominal or ordinal scale; Y axis is how often that X value occurred in measurements or observations
- Use/value: **Easy to see relative magnitudes / frequencies**
  - Sometimes low frequency items are of interest
    - For example, dissatisfied customers: histogram may “minimize” these high-impact but infrequent occurrences
  - Can use different color or other ways to highlight importance
- Sometimes multiple bars for each item (e.g. last year / this year), to show trends and changes
- Pie chart representation useful if these are parts of a whole
  - Not very good if there are several low-frequency items of interest
- Sometimes cumulative frequency line added to show “total at or below this level” – useful if X axis is ordinal scale

Histogram Example: A Pizza Shop



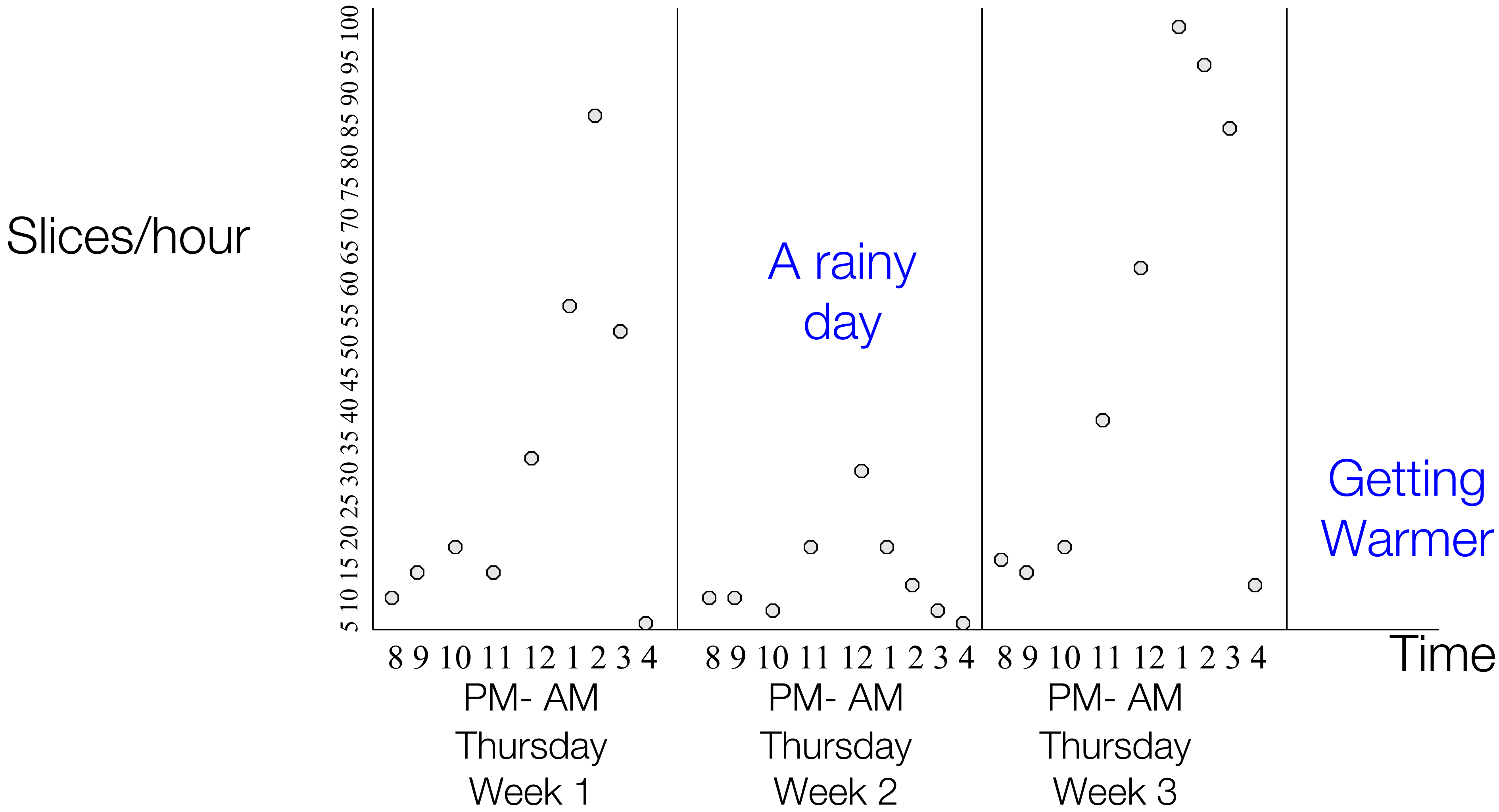
Example: Distribution of Component Size



Run Charts

- Plot of **some measurement/metric vs. (usually) time**
  - Use this when X axis is interval or ratio scale, such as project time, component size, team size, etc.
- Often used to show **trends over time**
  - Easier to spot overall upward or downward trend, or cyclical variations and other patterns
- Visually **separate random from significant variation**
  - Major **spikes or valleys** are triggers for explanation, investigation, or action
- Value: Identification of problems, trends, unexpected good results (may learn a lot from these)

Run Chart Example for Pizza Shop



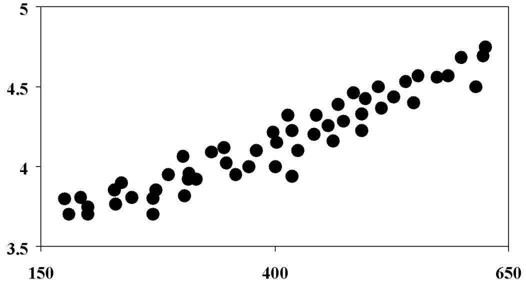
Scatter Diagram

- Used to determine **whether there is really a relationship between two variables**
  - **Fishbone cause-effect** diagramming identifies possible causes
  - Doing a scatter plot can show whether the proposed cause and its effect are correlated
    - Visual plot can show degree of correlation, non-linear correlations
      - Often annotate fishbone diagram to show whether a possible cause-effect has been shown to be statistically correlated
    - Linear correlations if most points are along a straight line
    - Poor (linear) correlation if points scattered all over
- Remember: correlation does not imply a causal relationship!

Scatter Diagrams

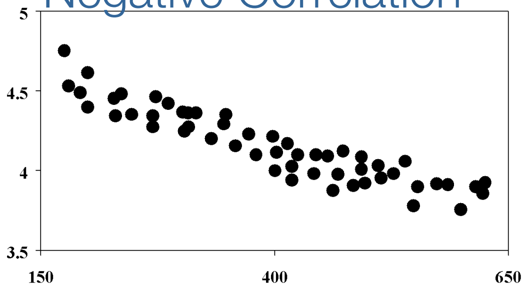
Measuring Relationships Between Variables

Positive Correlation



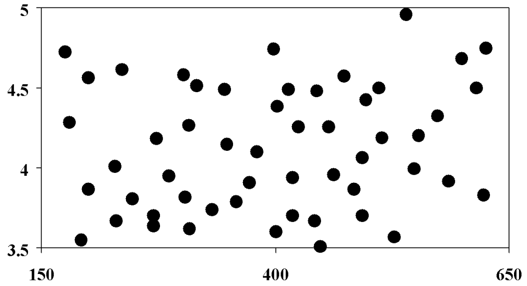
An increase in y may depend upon an increase in x.

Negative Correlation



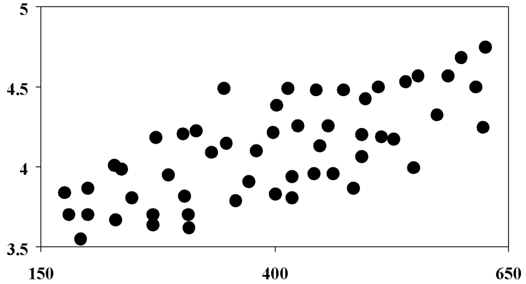
An decrease in y may depend upon an increase in x.

No Correlation



There is no demonstrated connection between x and y.

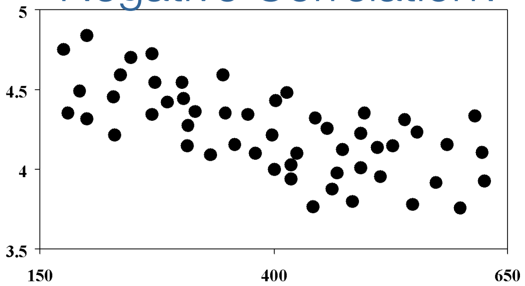
Positive Correlation?



If X is increased, y may also increase.

Non-Linear Correlations?

Negative Correlation?



If X is increased, y may decrease.

Example Scatter Plot from Tian Text

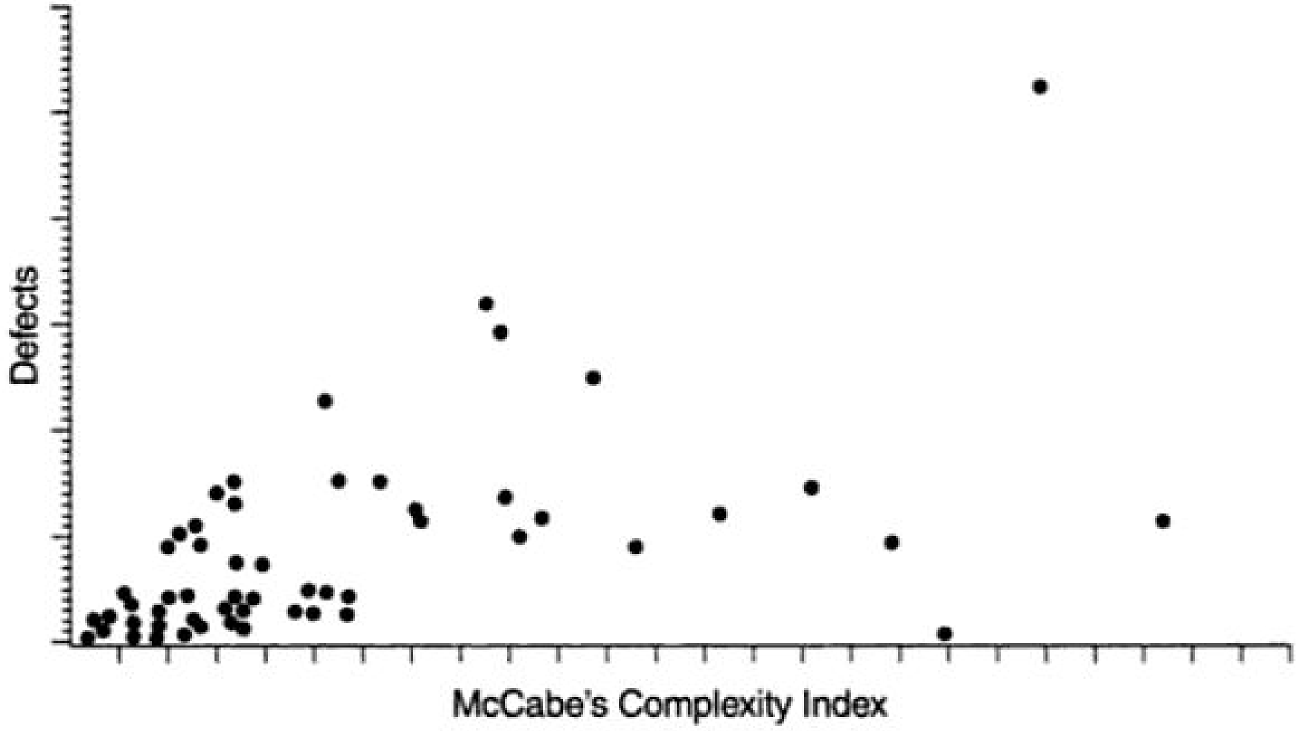


FIGURE 5.9
Scatter Diagram of Program Complexity and Defect Level

Example Scatter Plot from Tian Text

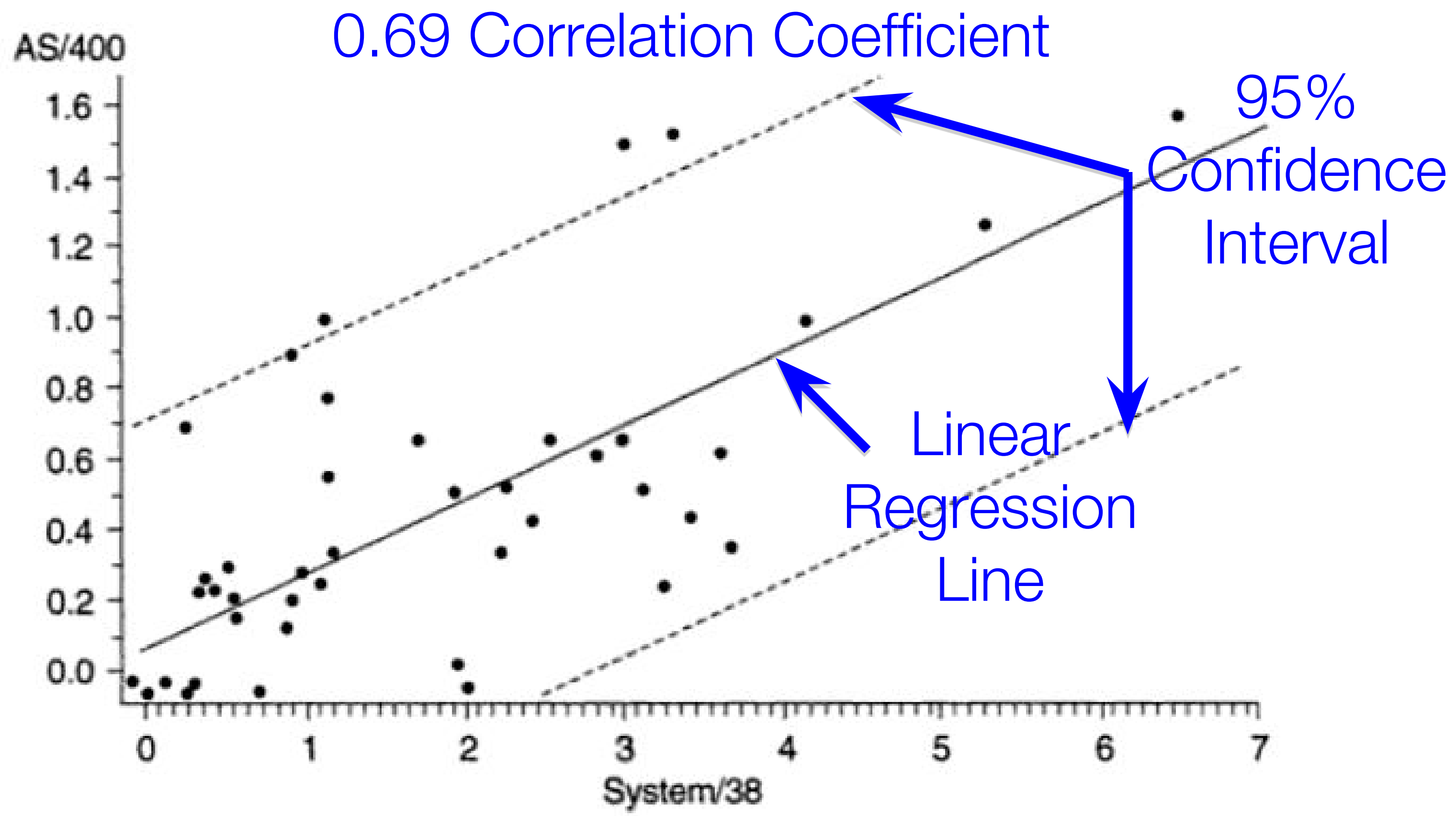


FIGURE 5.10
Correlation of Defect Rates of Reused Components Between Two Platforms

Example Scatter Plot from Tian Text

- Classify the scatter plot according to medians of component defect rate
  - Apply different analysis and improvement strategies to different quadrants

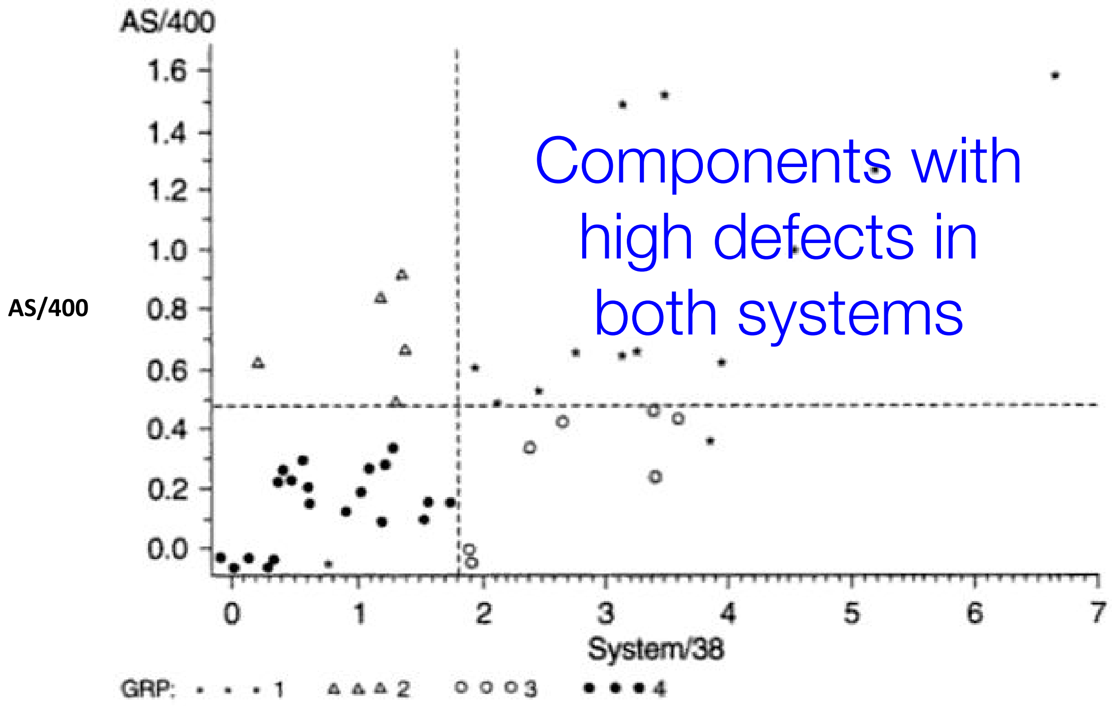
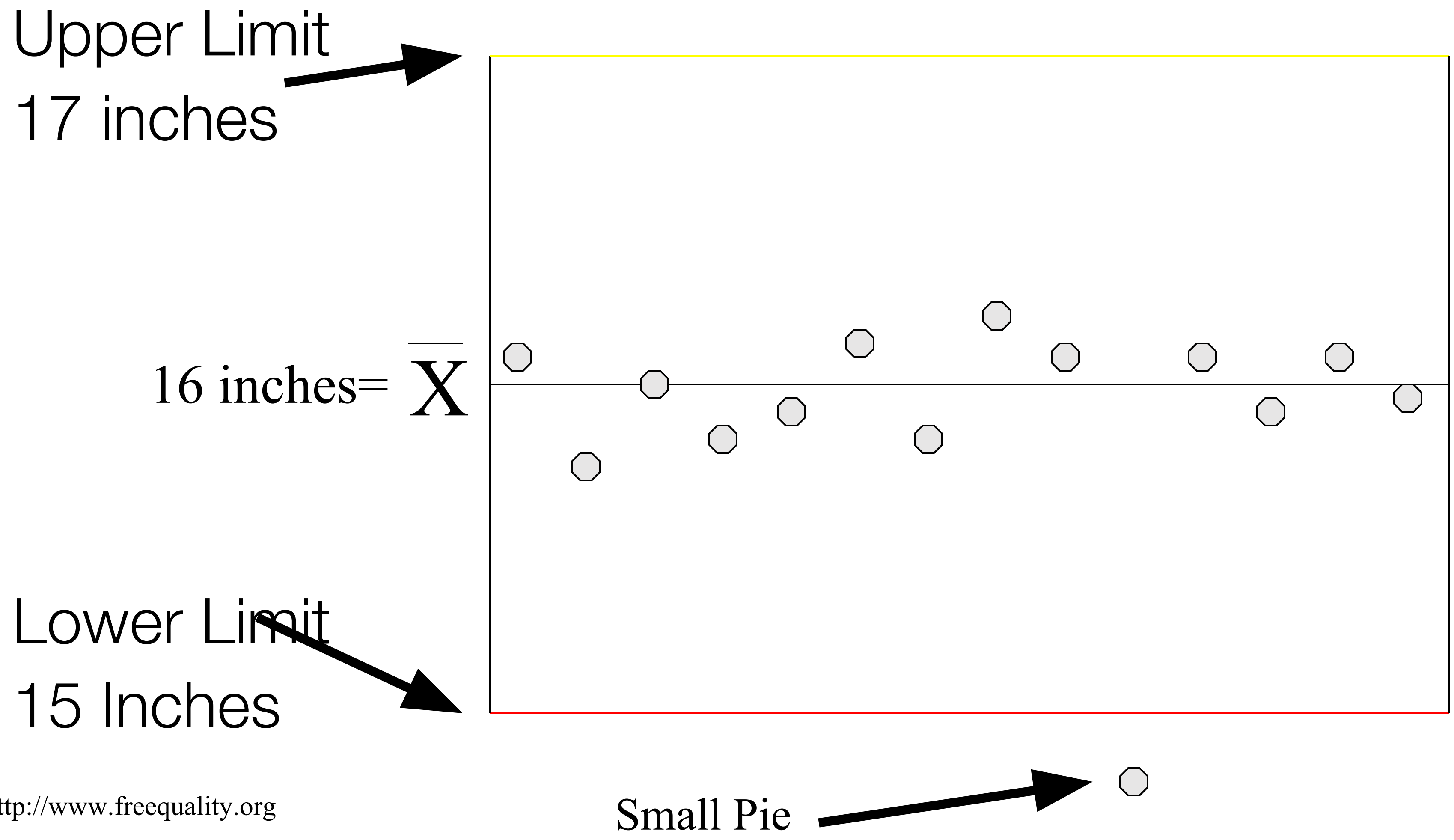


FIGURE 5.11
Grouping of Reused Components Based on Defect Rate Relationship

Control Charts

- Plot of **a metric with control limits** defined
  - Upper control limit: If value of metric exceeds this, take action
  - Lower control limit: If value goes below this, take action
  - Warning levels: If value outside this, check if all is well
- Control limits may be derived statistically or less formally (based on "reasonable" values or other impacts)
  - Formal statistical process control has formulas for deriving limits: often 3 sigma deviation from desired outcome
- Useful to **flag "outlier" values**, such as components with very high defect rates, projects that have parameters outside "normal levels" etc.
- Formal statistical process control not used much in software

Control Charts -- Pizza Example



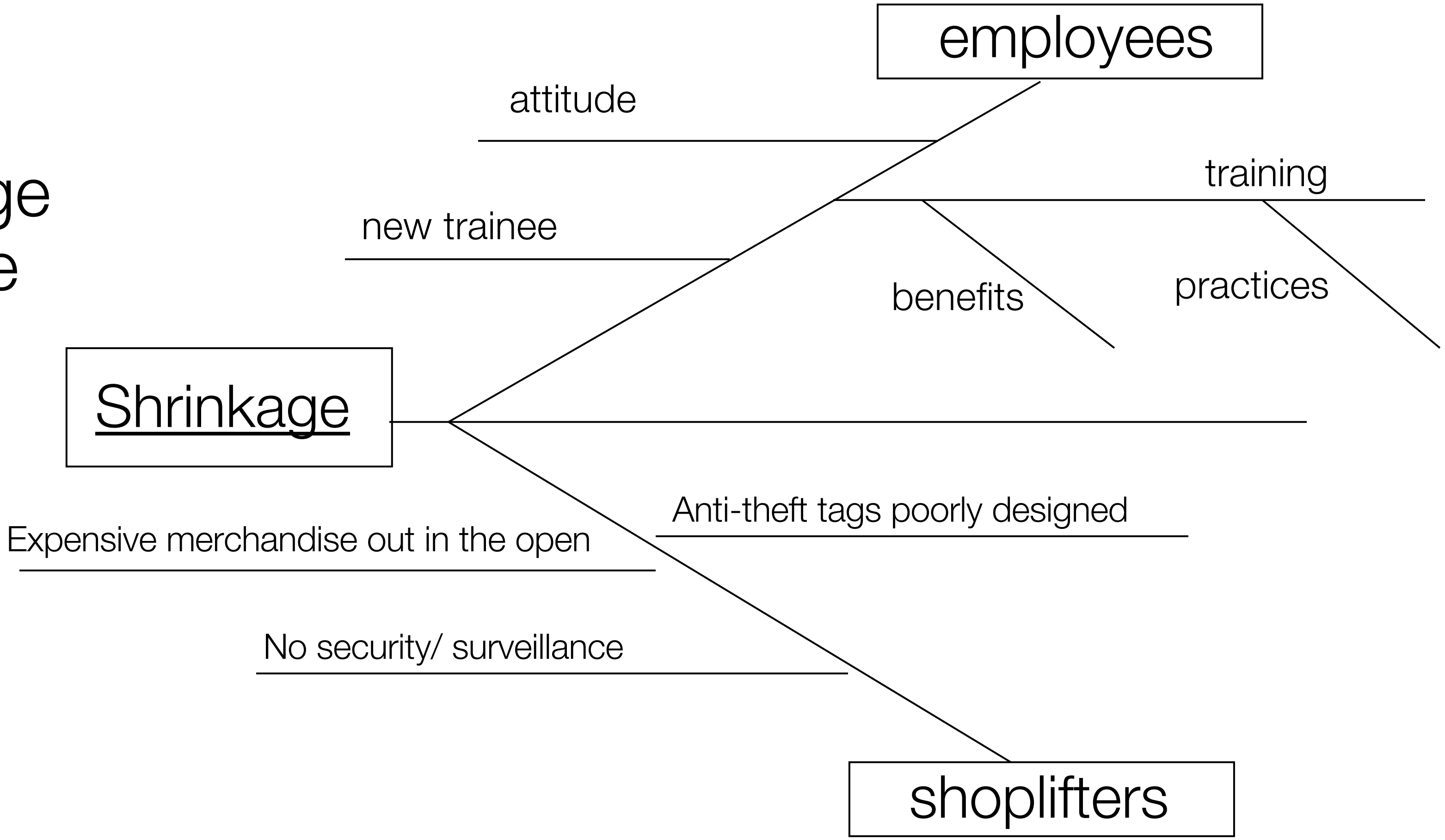
From http://www.freequality.org

Cause-And-Effect (Fishbone) Diagram

- Diagram showing **hierarchical structure of causes** that contribute to a problem or outcome:
    - Problem of interest forms the backbone
    - Spines are causes that contribute to the problem
    - Spines may have bones that represent its contributory factors and so on
- Used in brainstorming to diagram and identify various possible factors contributing to a problem, and to identify causal sequences (A causes B causes C) and root causes
    - Very simple but extraordinarily useful tool
- Initially both minor factors (that occur rarely or contribute very little) and major causes may all get listed

Example Fishbone Diagram

- Example: High Inventory Shrinkage at local Drug Store



Design Inspection Example from Tian Textbook

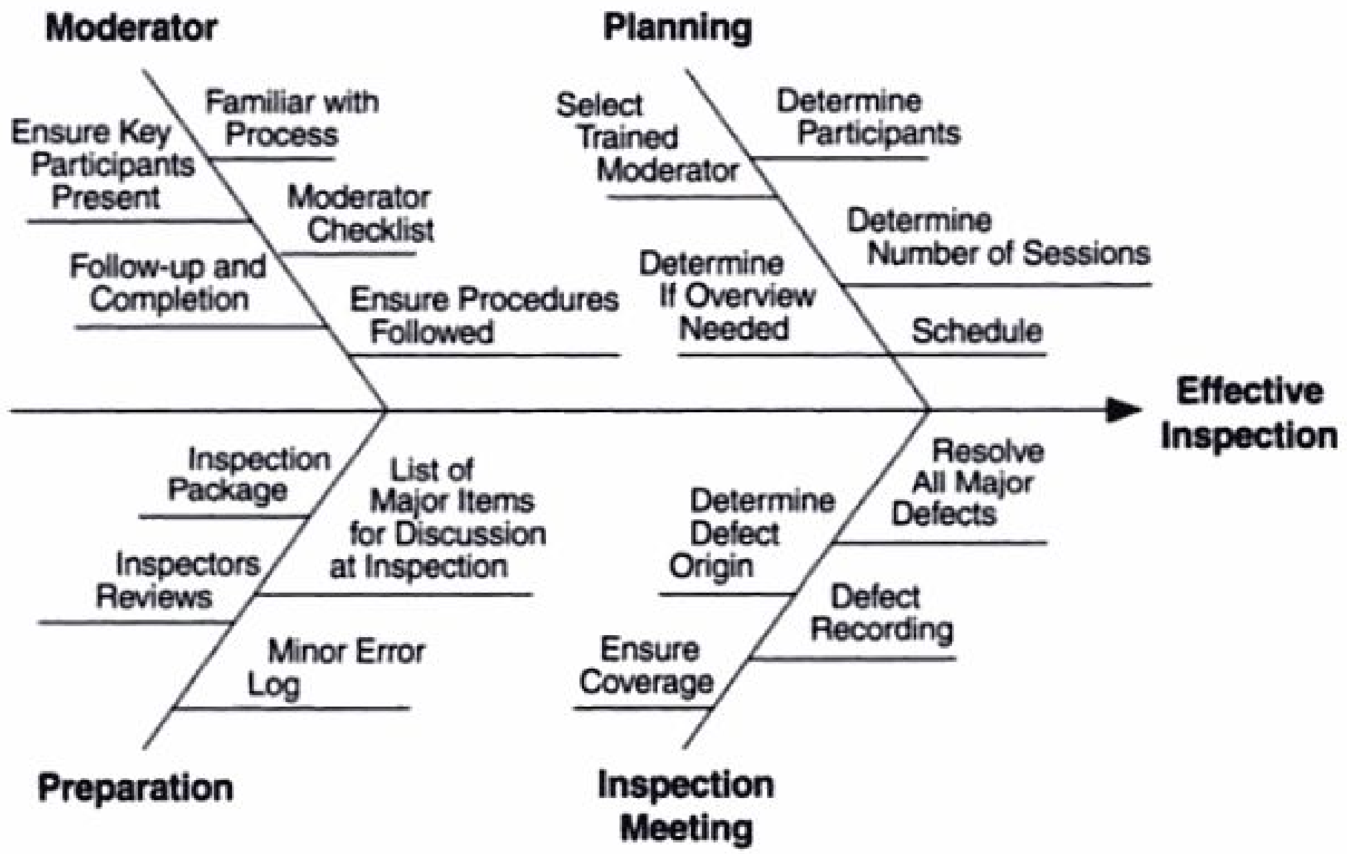
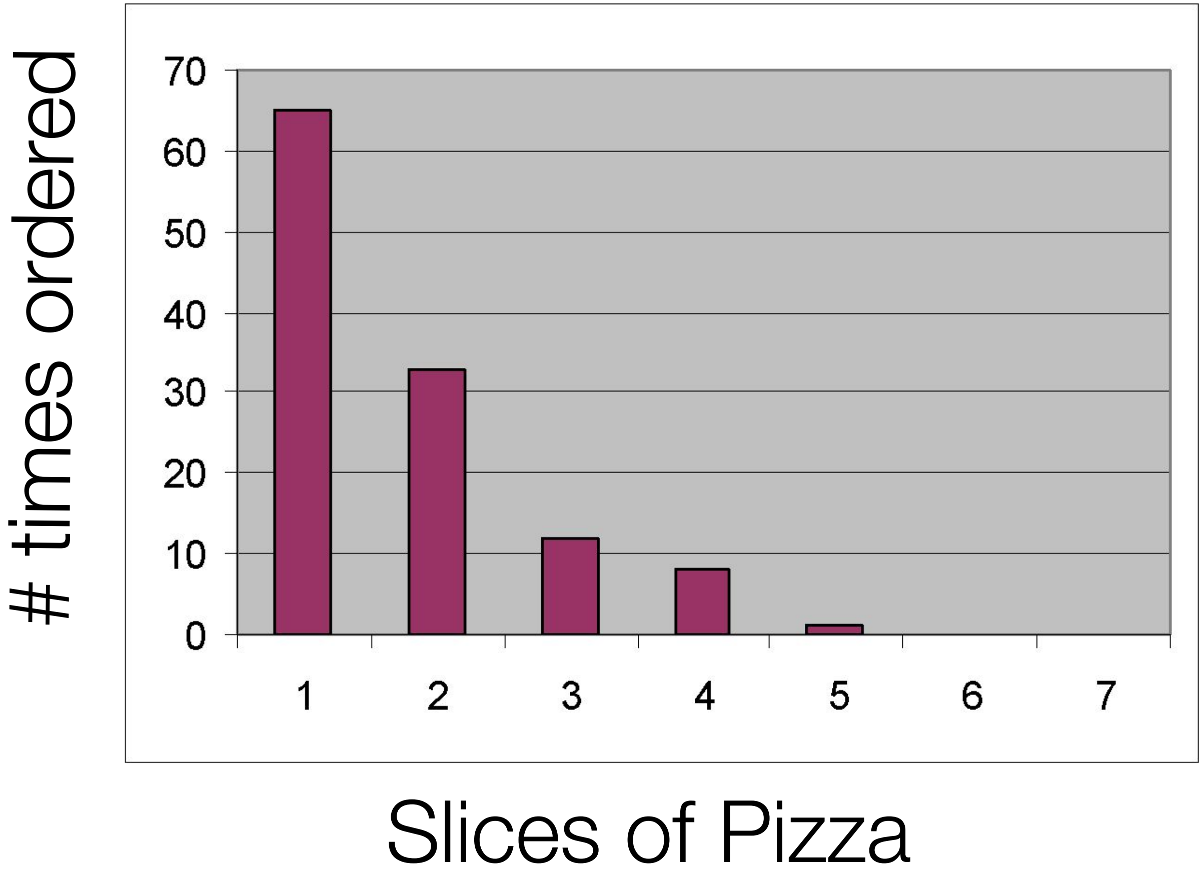


FIGURE 5.16
Cause-and-Effect Diagram of Design Inspection

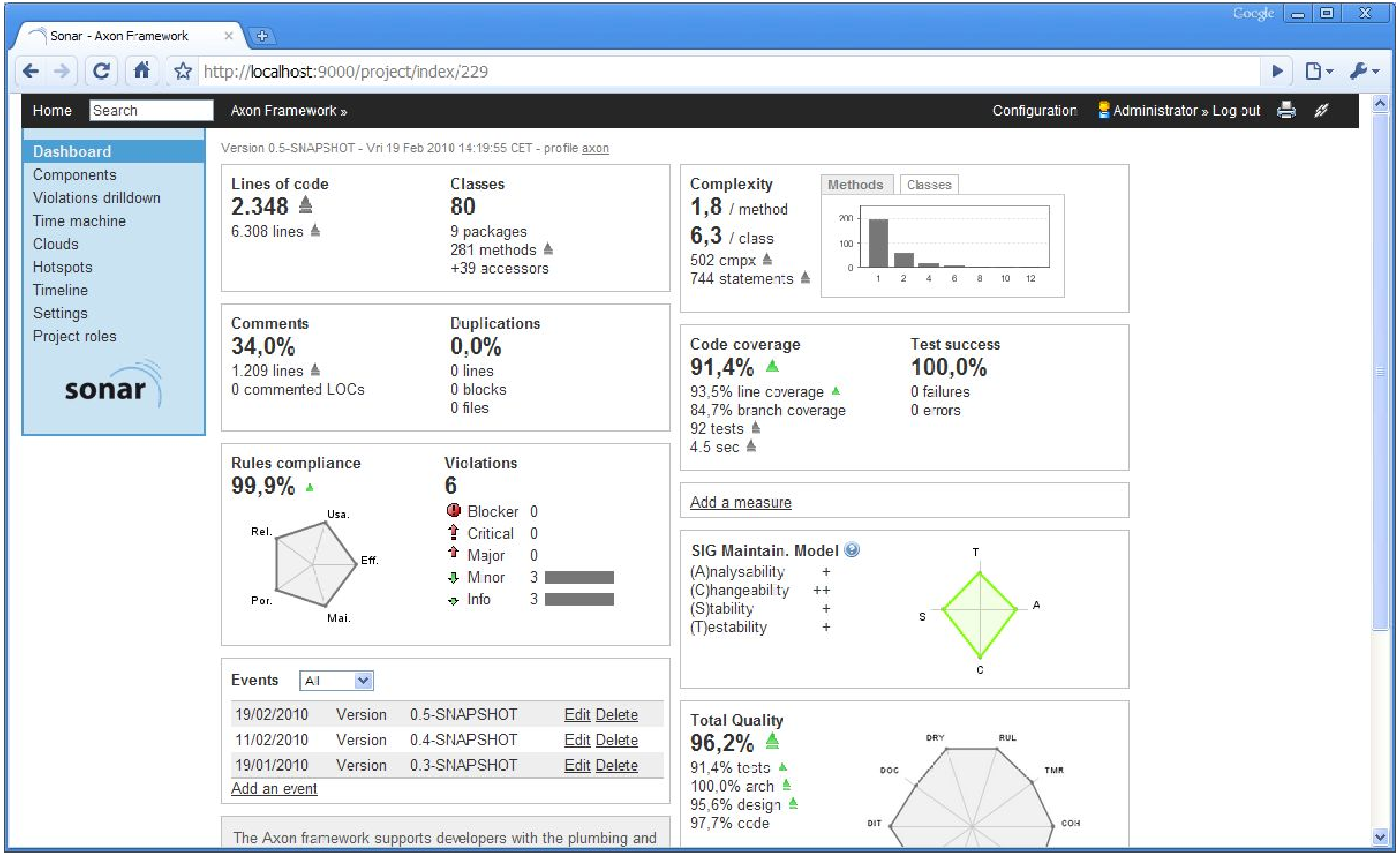
Pareto Diagram

- Histogram arranged by **decreasing frequency**
- Used to identify **causes that contribute most** to the problem
- After fishbone analysis, may do data gathering to figure out the frequency with which each cause contributes to the problem
  - In software, **review reports** are good data sources
- Plot histogram, identify the major causes
- Based on Juran's Pareto Principle – the 80/20 rule
  - "80% of the effects come from 20% of the causes"
  - Indicates general principle that some causes likely to be a lot more significant than others
- Highest cost-benefit from **addressing the most significant problems**
  - Less significant problems may barely be worth addressing

Pizza Shop Example



Quality Metrics Dashboard



Four Basic Defect Prevention Tools

- Checklists
- Templates
- Processes
- Workflow automation

Checklists

- Once we identify the causes of problems, **how do we eliminate them**?
  - Checklists are simple and incredibly effective at preventing & eliminating defects on repetitive tasks
    - To Do lists, "did you ..." on bill payment envelopes, etc.
- Capture knowledge about **common problems** and how to avoid them
- Can be used in **review processes** to identify problems
- **Lightweight**: low additional effort to use (not zero!)
- Checklists that become **too long lose value** (use Pareto analysis!)

Flowcharts (Process Diagrams)

- Flowcharts show **sequencing of activities and decisions**
  - Depiction of processes for doing things
- Streamline the **flow of activities**
- Capture knowledge about **how to perform activities effectively**
- Eliminate problems due to missed activities and badly sequenced activities
- Can be used to **analyze and implement improvement ideas**:
  - Good processes can save work and avoid problems
    - Less than zero cost for improving quality
    - Should always be the goal of process design

Templates (A Type of “Checklist”)

- Templates are another near zero-cost defect elimination mechanism
    - Pre-created document structure
    - Often pre-populated with “boilerplate” stuff: standard explanations, disclaimers etc.
    - Avoids problems due to missing information, incompleteness
        - Avoids problems in activity for which the document is the output
        - Need to fill in form, so get the data/do the activity!
- Problems with templates:
    - Not all sections are always applicable; may sometimes want different structure
    - Can constrain people from doing what they need to
    - Can lead to “automaton” mode where people just fill in form without thinking if that’s the most appropriate thing to do
- Make templates as guidelines, not “set in stone” forms

Workflow Automation

- **Creation of computerized tools that streamline activities**, such as automated check-in and build, automated testing
    - Implements process, templates
    - Eliminates many kinds of defects
    - Saves effort
- **Flexibility is often a major problem**
    - If the needs are different from what the tool supports, can't do it at all (or significant work-arounds)
    - Designing flexible tools which automate workflow is a major technical challenge

Summary

- The quality tools provide a suite of methods for quality analysis and control:
  - **Histograms, run charts, control charts** can identify problems
  - **Fishbone** is used to brainstorm possible causes
  - **Scatter plots** can be used to analyze whether relationships exist
  - **Pareto analysis** identifies which causes are most worth addressing
  - **Checklists, templates, process definition and workflow automation** can prevent problems

Discussions

- Can you manipulate these metrics?
- What are the problems of these metrics?
- Do you think Dashboard really helps?

DRE Table

Phase of Origin

Phase Found

	Req	Des	Code	UT	IT	ST	Field	Total Found	Cum. Found
Req	5							5	5
Des	2	14						16	21
Code	3	9	49					61	82
UT	0	2	22	8				32	114
IT	0	3	5	0	5			13	127
ST	1	3	16	0	0	1		21	148
Field	4	7	6	0	0	0	1	18	166
Total Injected	15	38	98	8	5	1	1	166	
Cum. Injected	15	53	151	159	164	165	166		

Phase Found

(Illustrative example, not real data) Phase of Origin