Software Engineering

mariaiuliana.dascalu@gmail.com

University POLITEHNICA of Bucharest, Romania
Department of Engineering in Foreign Languages

Software Systems Engineering

- Systems that are composed entirely of software are often considered "just" software projects, not system projects, and no effort is expended to develop a systems engineering approach.

- This neglect of the systems aspects of software product development has contributed to the so-called *long-running software crisis*.

- (Qs1) Why should we apply SE principles when developing software? A code-centric approach is not enough? (Reading2 material)

https://www.dropbox.com/s/ymygdyt2gxdtjoc/Reading2.pdf?dl=0

- (Qs2) What is the *long-running software crisis?* Give an illustrative example. (Reading3 material)

https://www.dropbox.com/s/3dlxpgjako5razh/Reading3.pdf?dl=0

Aspects of Quality Assurance in SwSE

Fault / defect / failure / error

- Fault (according to ISO/CD 10303-226)**:**
  - “an abnormal condition or defect at the component, equipment, or sub-system level which may lead to a failure”
- Failure:
  - the state or condition of not meeting a desirable or intended objective
- Defect:
  - see fault
- Error:
  - the occurrence of an incorrect result produced by a computer

Examples of defects

- The code has only 3 choices for the city but the design document lists 4.
- A "default" path was tested which would have had the result that an error message was printed, but because the default clause had been omitted, no error message was printed, although the data wasn't used.
- When "save" button was selected, the data was lost. Apparently, the data was being saved to one location and being read from a different one. The developer had to redo how the data was being accessed.
- When the tester tried to add one more character than the maximum allowable, an error message was displayed which just showed one word per line.

Discussion

- How would you classify the previous defects to the following categories?
  - Design quality
  - Runtime error
  - Data access
  - User interface

Discussion

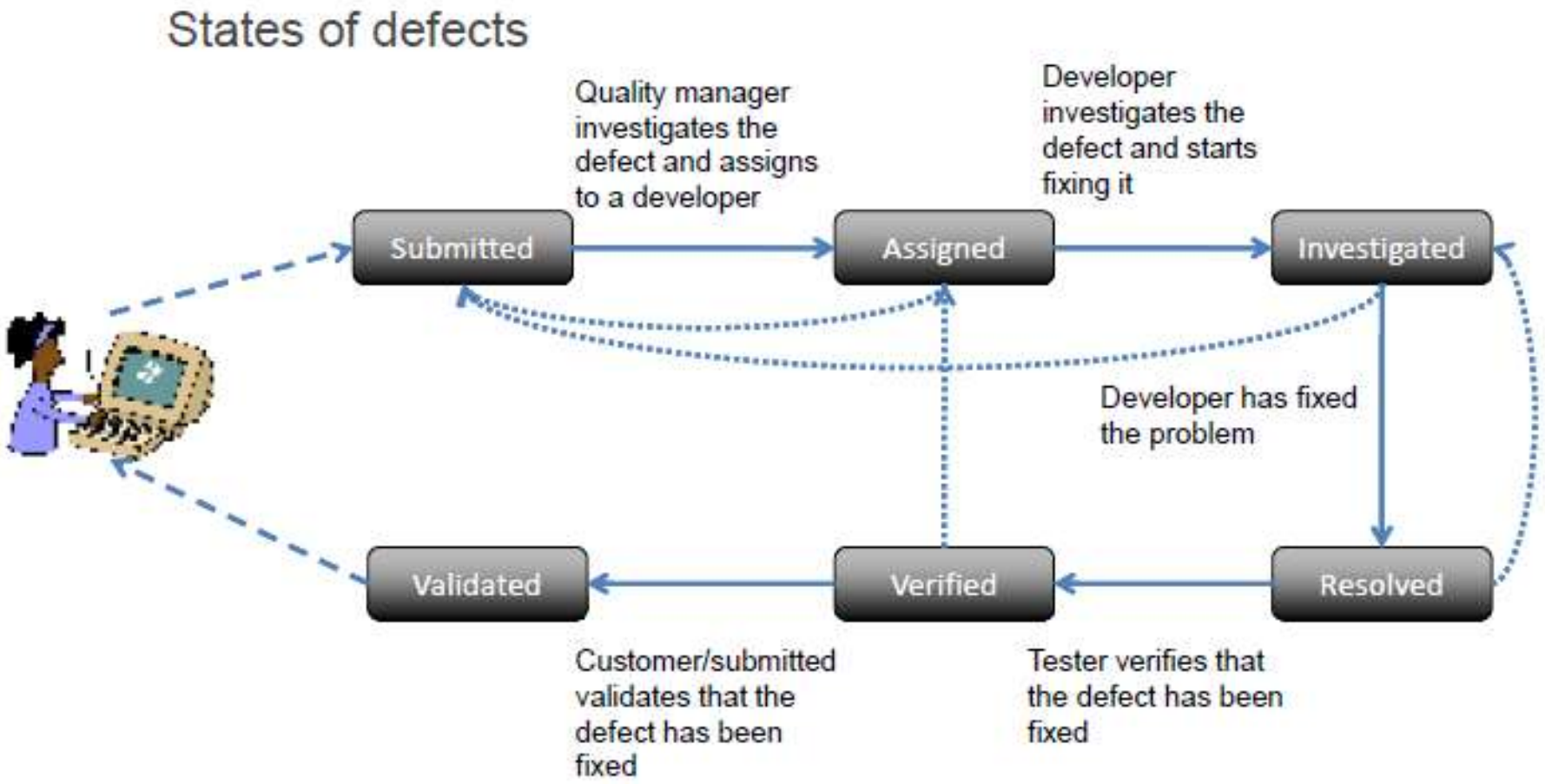
- How would you evaluate the classification from the previous slide?
- Is it unambiguous? Was it clear to which category each defect belong?
- Is it overlapping?
- Is it well defined?

- Given answer: [filled in-class]

During the development, a defect can be in several states

- This list is based on the list of states from Rational Clear Quest (http://www-01.ibm.com/software/awdtools/clearquest/):
  - Submitted (S)
  - Assigned (A)
  - Investigated (I)
  - Resolved (R)
  - Verified (V)
  - Validated (Z)
  - Closed (C)
- The important thing is that these states change forward (and sometimes backwards) during the development cycle

States of defects



Orthogonal Defect Classification (ODC)

- IBM defines ODC as:
  - a scheme to capture the semantics of each software defect quickly; it is the definition and capture of defect attributes that make mathematical analysis and modelling possible
- is “ORTHOGONAL”

Fundamentals of ODC

- The goal of ODC is to classify defects in order to improve:
  - the time it takes to pinpoint the solution to the problem
    - measured by: Time from S -> I
  - correctness of the solution
    - measured by: e.g. number of new defects introduced by this fix (fixing one defect might cause several others!)
  - time to provide the solution
    - Measured by: Time from I -> V
- categories:
  - Opening: when the defect was **detected**
  - Closing: when the defect was **investigated**

Opener Section (These attributes are usually available when the defect is opened.)		
Defect Removal Activities	**Triggers**	**Impact**
Design Rev, Code Inspection, Unit test, Function Test, System Test.	1. Design Conformance 2. Logic/ Flow 3. Backward Compatibility 4. Lateral Compatibility 5. Concurrency 6. Internal Document 7. Language Dependency 8. Side Effect 9. Rare Situations 10. Simple Path 11. Complex Path 12. Coverage 13. Variation 14. Sequencing 15. Interaction 16. Workload/Stress 17. Recovery/Exception 18. Startup/Restart 19. Hardware Configuration 20. Software Configuration 21. Blocked Test (previously Normal Mode)	1. Installability 2. Serviceability 3. Standards 4. Integrity/Security 5. Migration 6. Reliability 7. Performance 8. Documentation 9. Requirements 10. Maintenance 11. Usability 12. Accessibility 13. Capability

Closer Section (These attributes are usually available when the defect is fixed.)				
Target	**Defect Type**	**Qualifer**	**Age**	**Source**
Design/Code	1. Assign/Init 2. Checking 3. Alg/Method 4. Func/Class/Object 5. Timing/Serial 6. Interface/O-O Messages 7. Relationship	1. Missing 2. Incorrect 3. Extraneous	1. Base 2. New 3. Rewritten 4. ReFixed	1. Developed In-House 2. Reused From Library 3. Outsourced 4. Ported

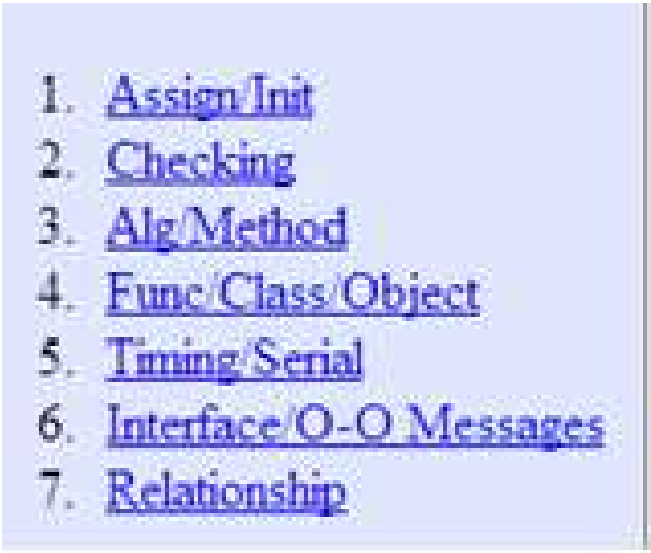
Exercise

- For the list of the following defects, please provide the classification of the *Defect Type* and explain your classification.

- Defect 1: *User screen should contain a link to the help. It was found in the test that it did not contain that link. The developer added the link to the user screen.*

- Defect 2: *Algorithm calculating the tax return returned an incorrect tax return value. After investigation it was found that the tax return rate was calculated based on the data from 2008 instead of 2009.*

- Defect 3: *The parameter of CDefectTypeClass.getID(intidType) was 0 (uninitialized) when called from CExerciseClass.classifyDefects when running the test case 081222_99_09*



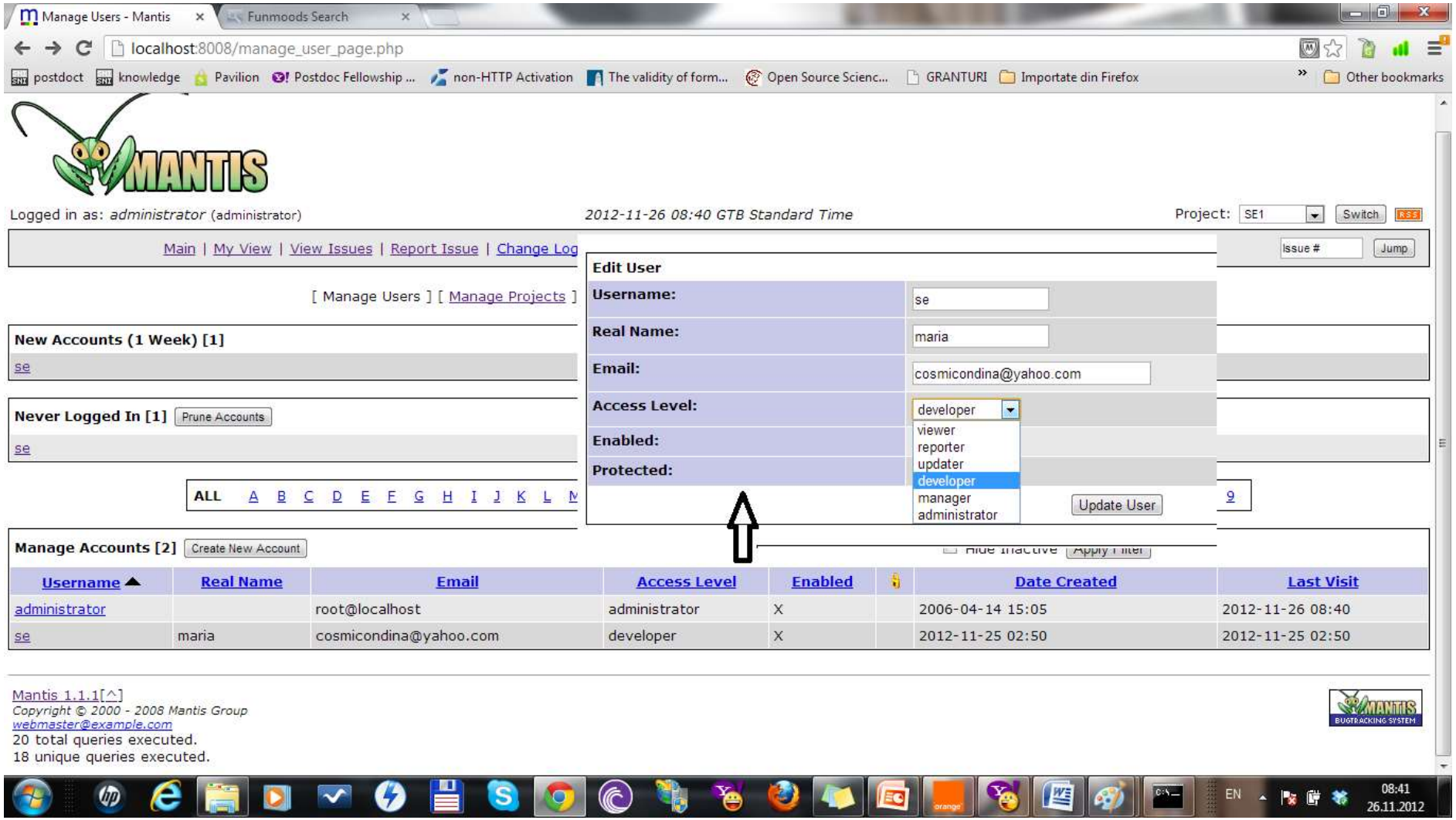
Other Schemes

- https://gupea.ub.gu.se/bitstream/2077/32048/1/gupea_2077_32048_1.pdf
- http://www.mysmu.edu/faculty/lxjiang/papers/wcre12defects.pdf
- …..

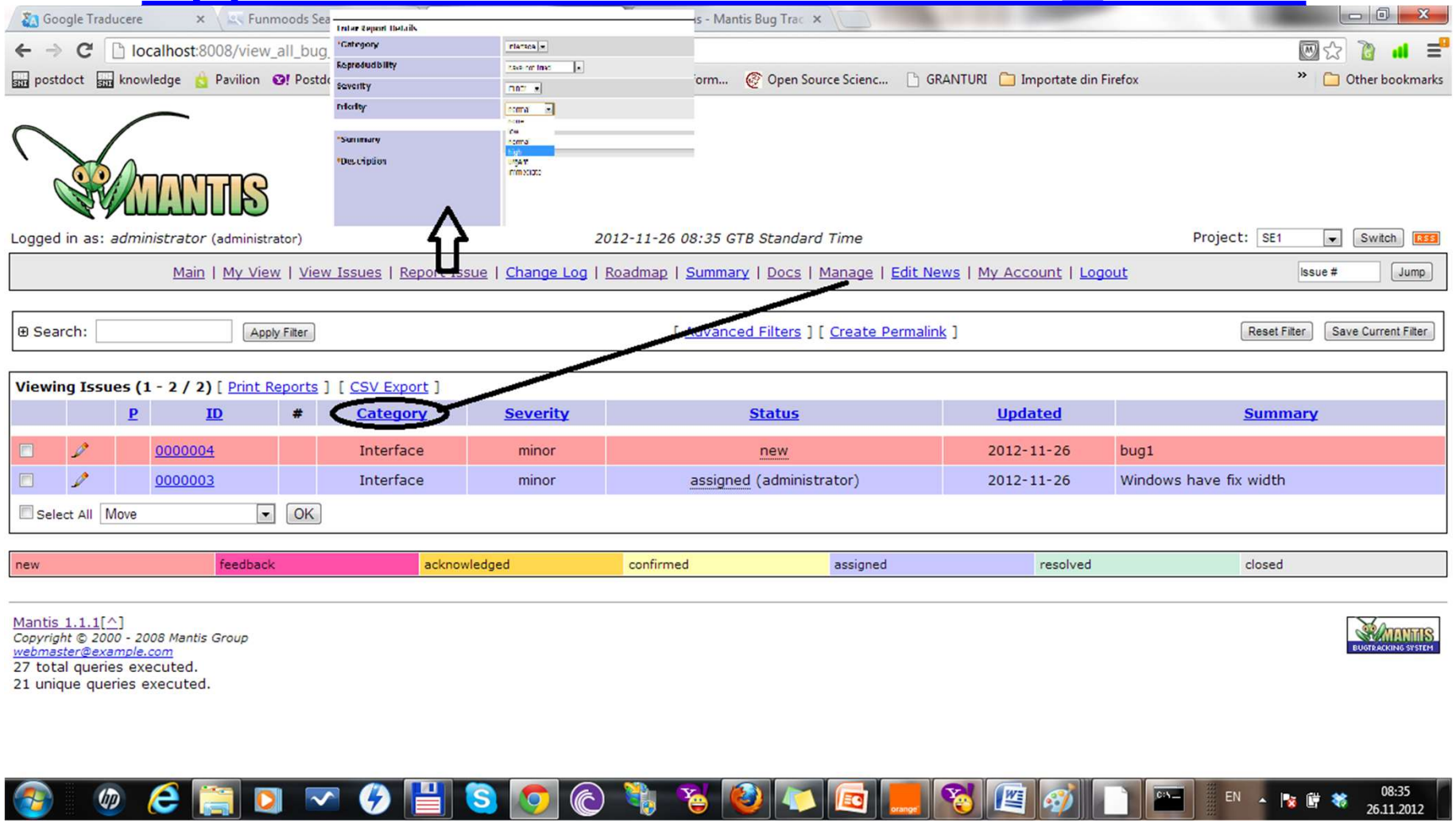
Bug Tracking Tools

- IBM Rational ClearQuest
- IBM Rational Quality Manager
- BugZilla
- Mantis
- ….

Instant Mantis



http://www.mantisbt.org/wiki/doku.php/mantisbt:instantmantis_install



Root Cause Analysis (RCA)

- RCA is used to find the root causes of defects – what is the cause of the failure
- RCA can be done using a number of methods:
  - Barrier analysis
  - Causal factor tree analysis
  - Change analysis
  - Failure mode and effects analysis
  - Fault tree analysis
  - 5 Whys
  - Ishikawa diagram
  - Pareto analysis
- More: https://www.dropbox.com/s/ojfz18ncqq3xqjj/Reading4.pdf?dl=0

5 Whys: Example

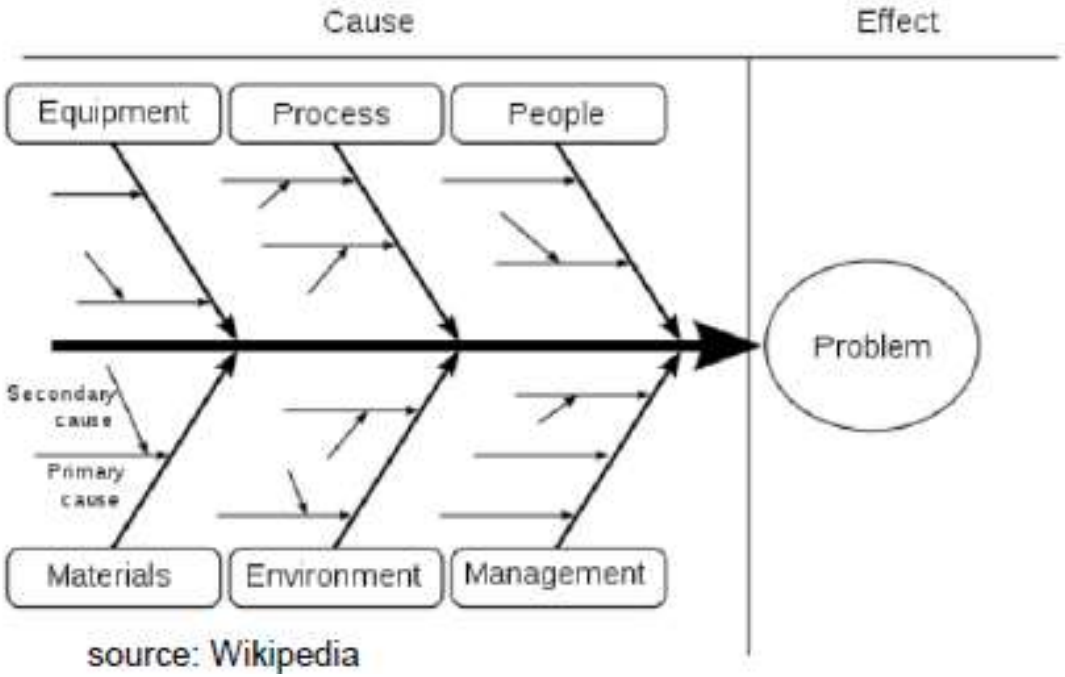
- Problem Statement: You are on your way home from work and your car stops in the middle of the road. (from www.isixsigma.com)

Ishikawa Diagram: Principles

- Causes are grouped into categories and linked together (primary and secondary causes)
- The list of categories is not definite, the figure shows the "typical"

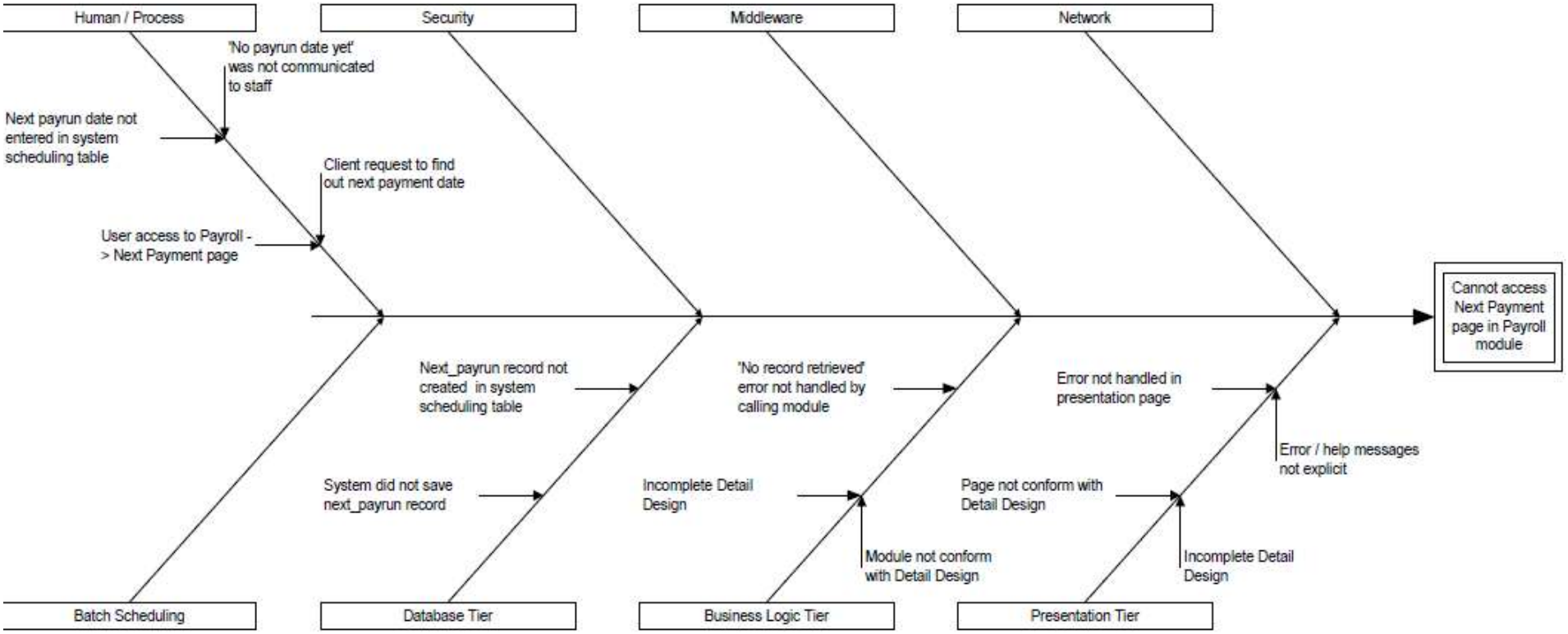
• Relationships are used to backtrack the cause of a particular problem.

• The analysis requires more effort than 5 whys but still no statistics is necessary



source: Wikipedia

Example


Human / Process
Security
Middleware
Network
'No payrun date yet' was not communicated to staff
Next payrun date not entered in system scheduling table
Client request to find out next payment date
User access to Payroll -> Next Payment page
Cannot access Next Payment page in Payroll module
Next_payrun record not created in system scheduling table
'No record retrieved' error not handled by calling module
Error not handled in presentation page
Error / help messages not explicit
System did not save next_payrun record
Incomplete Detail Design
Page not conform with Detail Design
Module not conform with Detail Design
Incomplete Detail Design
Batch Scheduling
Database Tier
Business Logic Tier
Presentation Tier

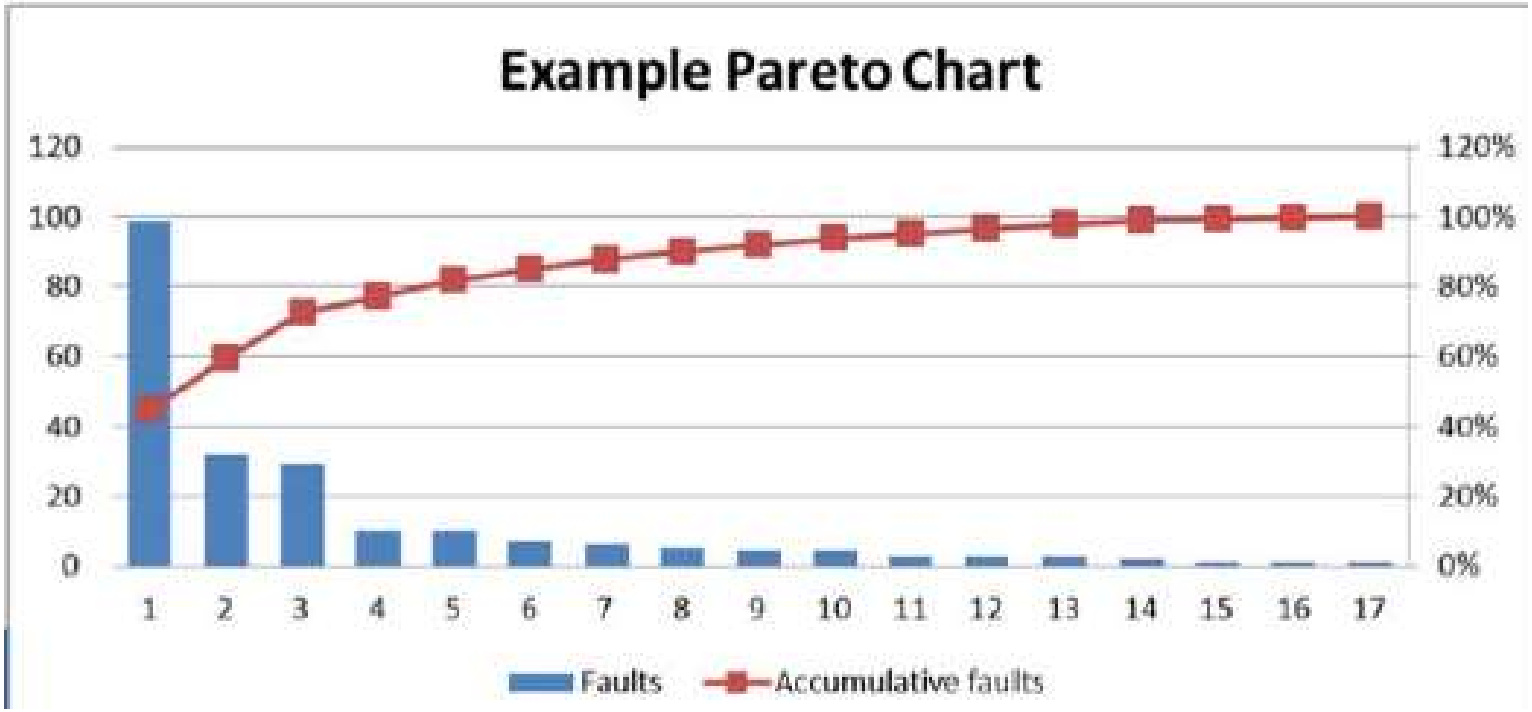
Exercise

- Imagine the following situation: You are a quality manager in a software development project in a new version of the Hyunday i30 car. A new version of the software for the ABS (Anti-Block System) brakes has been delivered and it does not work. Please draw the Ishikawa diagram that would help you do Root Cause Analysis.

Pareto Analysis: Principles

- Pareto analysis is based on the assumption that 20% of the units contributes with 80% of the effect, e.g.:
  - 20% of the modules contain 80% of errors
  - 80% of failures are caused by 20% of the defects
- The basis of Pareto Analysis is Pareto diagram

Pareto Diagram: Example



What are the most “dangerous” defects according to the above diagram?

More: https://www.dropbox.com/s/r66fx4mr3i8urxk/Reading5.pdf?dl=0

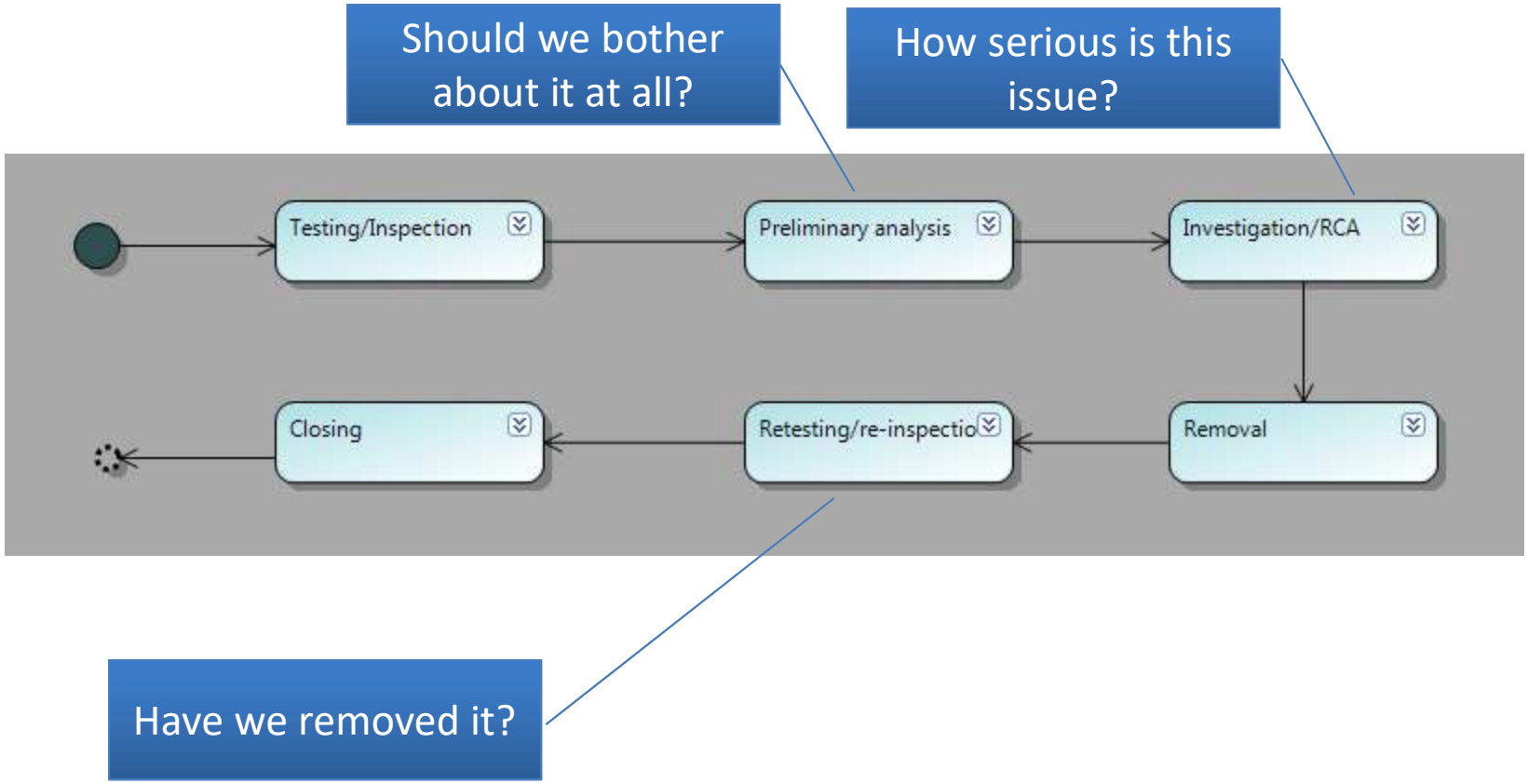
What we've done so far…

- Defined quality management
- Defined the defect
- Looked at different states of the defects…
- It's obvious we make mistakes and we will make even more of them…
- So, what do we do with them?
- We cannot fix all of them, so…?

Two groups of defects

- Pre-release defects
  - Defects discovered during testing
  - Defects which need to be fixed before the release of the product
- Post-release defects
  - Defects discovered by the users after the product was released
  - Defects which need to be addresses via updates and service releases

Pre-release defects: Typical process



Pre-release defects: Processes

Important to know that the defects have to be prioritized, e.g.

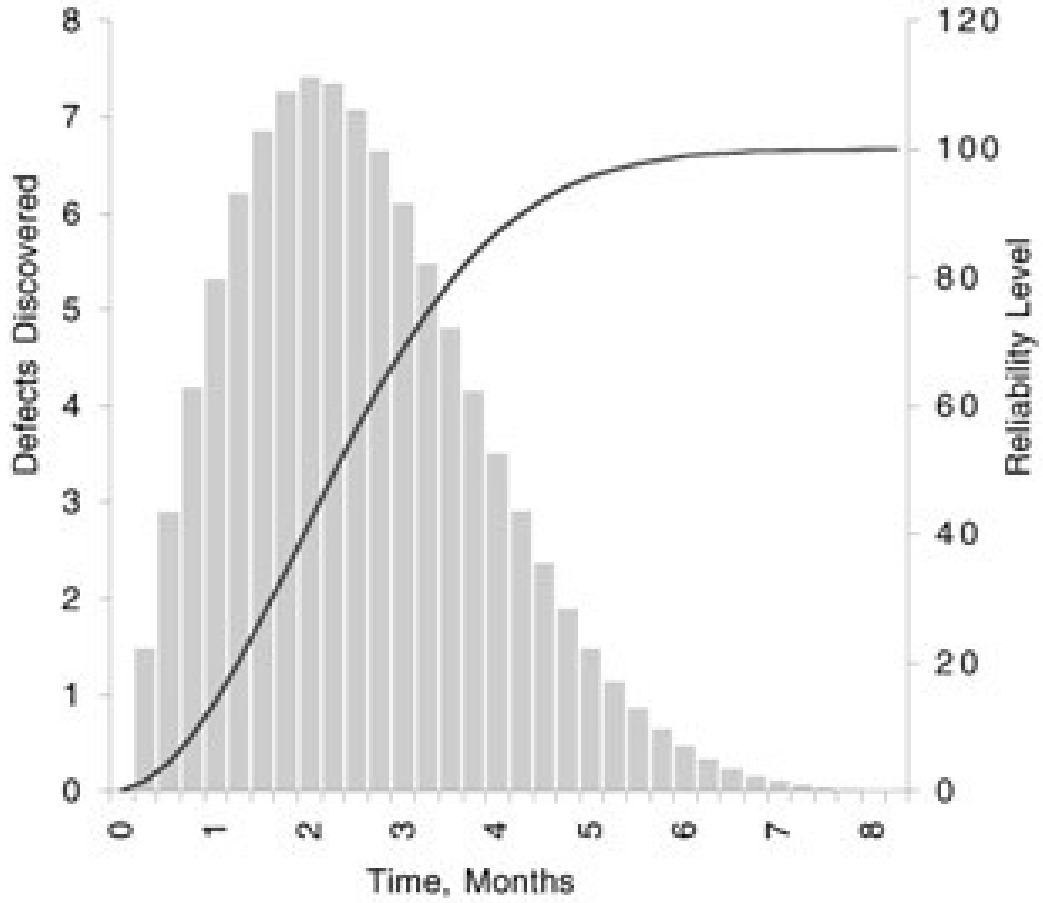
Severity A: Crashing the whole system, impacts the architecture
Severity B: Functionality of the product is affected, fixes do not impact architecture
Severity C: Somewhat affects functionality
Severity D: Affects performance, not functionality
Severity E: Enhancement request; good to have, but not necessary
Severity F: Question

Important that there changes are controlled

It might happen that fixing one defect might introduce so much change that the team leaves it as it is and “contaminates” the problem somehow (e.g. by adding exception handling)

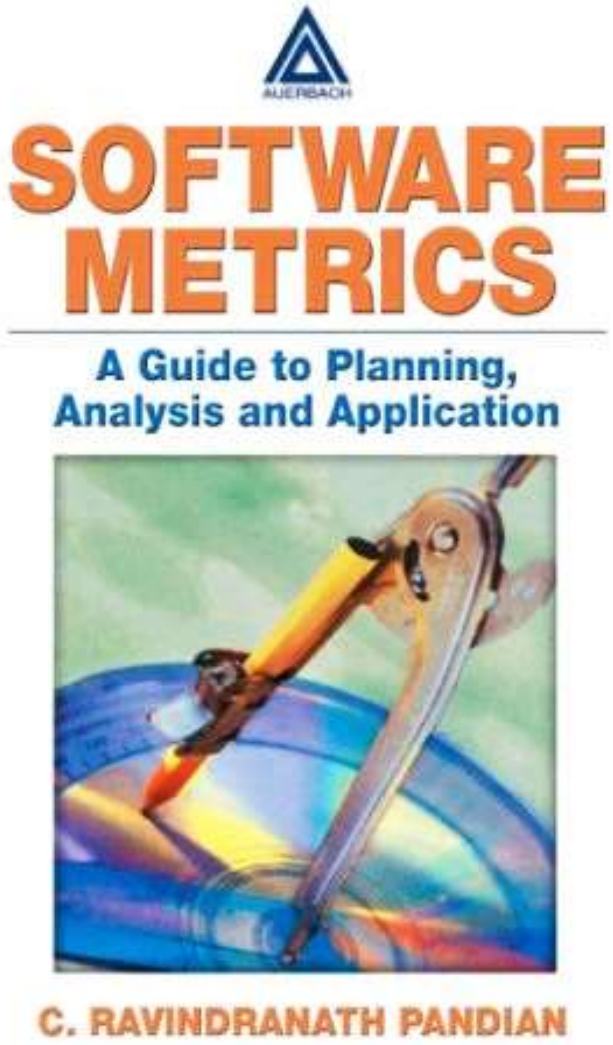
Post-release defects: Reliability theory

- Reliability theory comes from hardware engineering



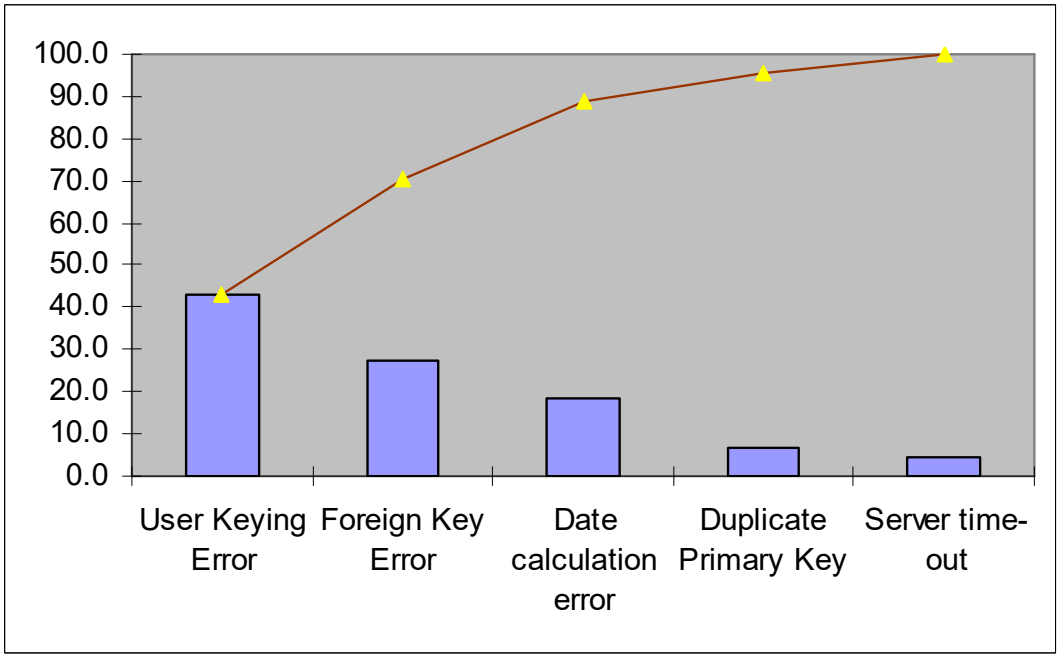
Graphs used during defect management

- Some example of graphs used during defect management:
  - Defect Life Time (control chart, priority wise)
  - Defect Review Effort (control chart)
  - Detection Effort (control chart)
  - Cost of Defect (control chart)
  - Cost (histogram)
  - Defect Life Time (histogram)
  - Review Effort (histogram)
  - Cost of Defect (histogram)
  - Defect Injection Profile, Defect Detection Profile
  - Detection Cost (fix cost, regression line)
  - Severity (detection effort, regression line), Severity (fix effort, regression line)
  - Defect (fix cost, cumulative graph)
  - Defect Arrival Graph, Defect Closure Graph
  - Reliability Growth Graph
  - Defect Severity (reliability bias bar graph)
- Examples: on Reading6 from the provided material



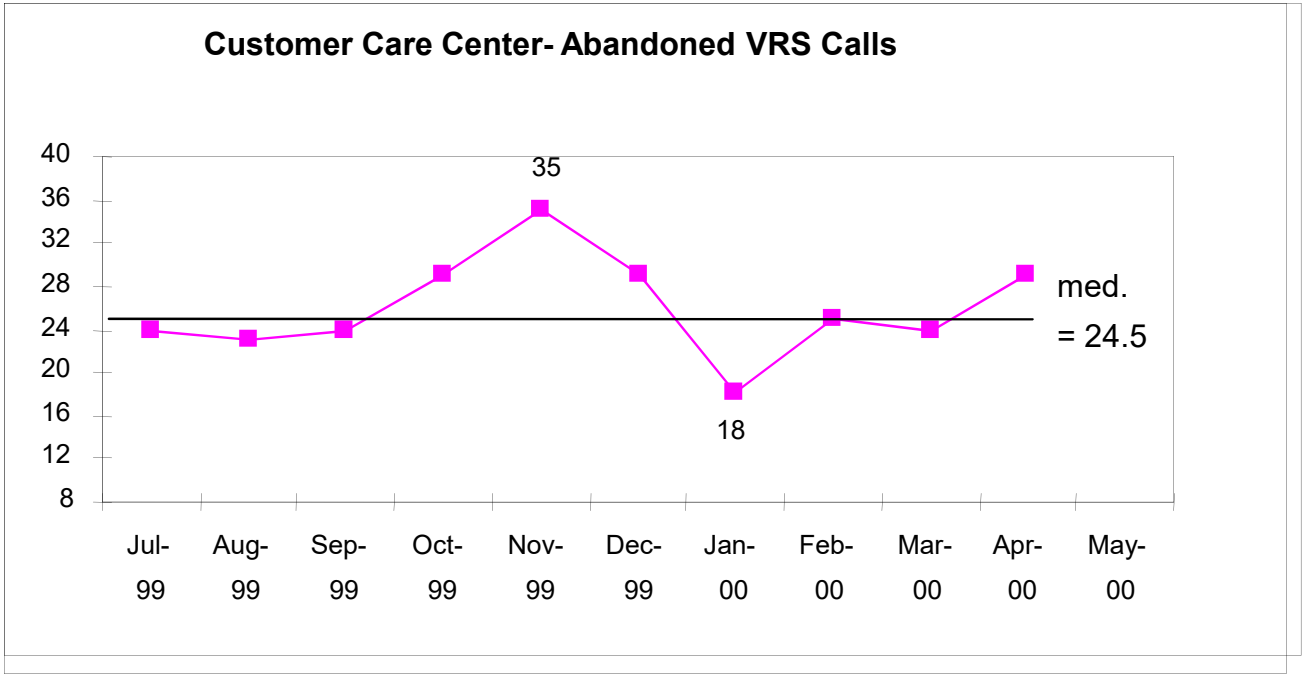
Pareto Chart (again…)

- Define categories
- Collect or classify data
- Calculate totals by category
- Calculate percentages
- Chart the results



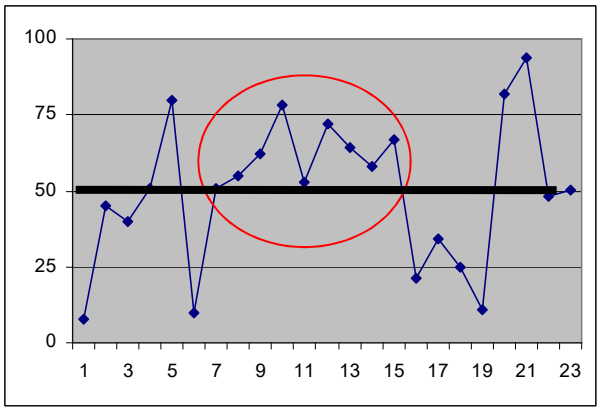
Run Charts

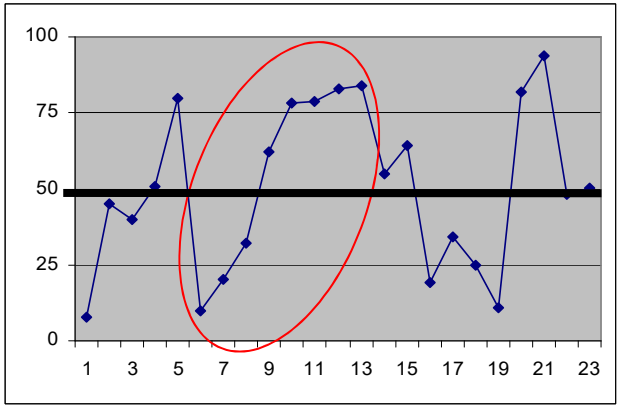
- Plot of a measured variable in time sequence
- Used to detect “assignable causes" of variation:

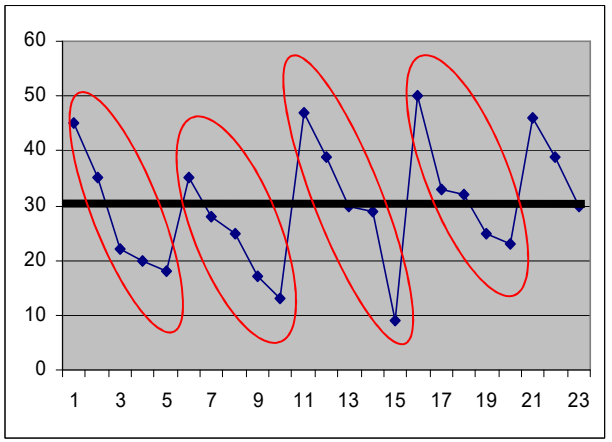


Things to look for:

- Shifts: eight or more consecutive points on one side of the median or target
- Trends: six consecutive jumps in the same direction
- Patterns: a pattern that recurs eight or more times in a row



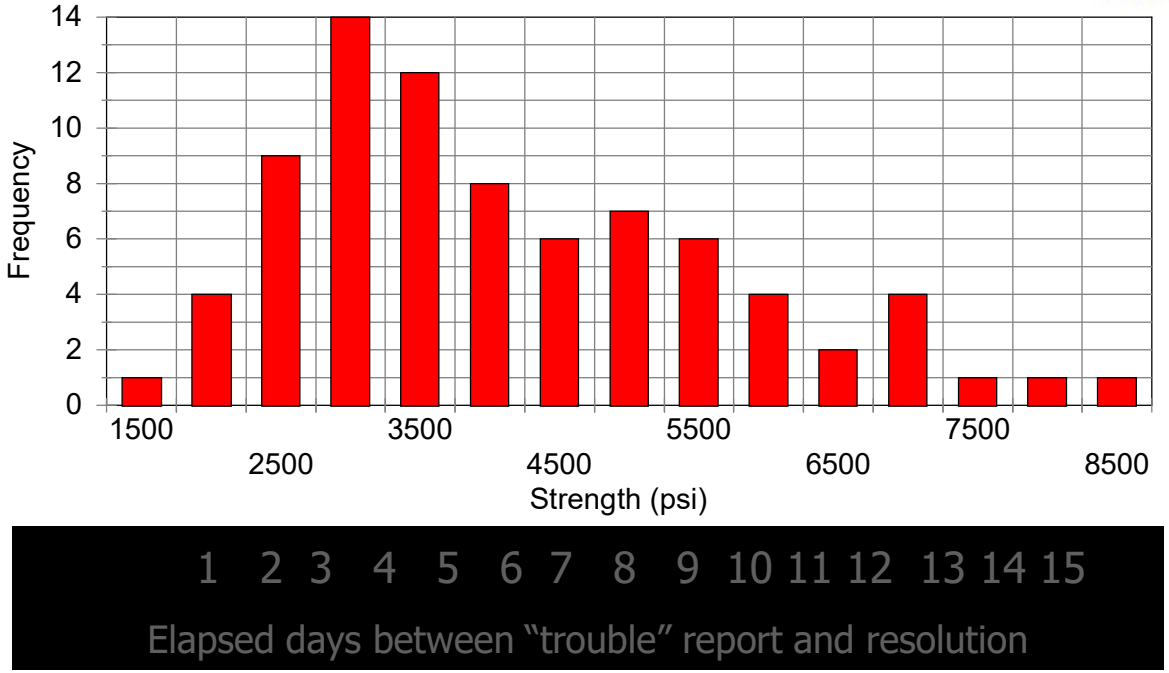




Histograms

- Histogram - a tally of process output falling into specific “categories” of continuous data
- Frequency distribution- an estimate of what the entire population histogram might look like

Defect	Submitted	Resolved	Age
Def1	2010-02-01	2010-02-12	0,030556
Def2	2010-02-01	2010-02-02	0,002778
Def3	2010-02-01	2010-03-01	0,083333
Def4	2010-02-01	2010-03-12	0,113889
Def5	2010-02-01	2010-02-07	0,016667



Defect discovery - Where do the defects come from?

- Techniques for discovering the defects
  - Inspections
  - Walkthroughs
- How to estimate how many defects remain undiscovered
  - Defect density
  - Capture-recapture