

# Metrics – Thinking in N-Dimensions

Silverpath Technologies Inc.  
Trevor.Atkins@silverpath.com

*Thinking  
Through  
Testing*

# What Are Metrics?

- ❖ *“Means by which software engineers measure and predict aspects of processes, resources, and products that are relevant to the software engineering activity.”*
  - Software Program Managers Network (SPMN)
- ❖ *“Software quality metric: A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality.”*
  - IEEE Standard for a Software Quality Metrics Methodology IEEE Std 1061-1998

# Goals of a Metrics Programme

- ❖ Reduce software lifecycle costs by improving process effectiveness and customer satisfaction
- ❖ Provide a quantitative basis for evaluating and making decisions about software quality in a timely manner
- ❖ Identify and increase awareness of quality requirements and goals
- ❖ Provide feedback on the metrics programme itself and validate the set of metrics being tracked

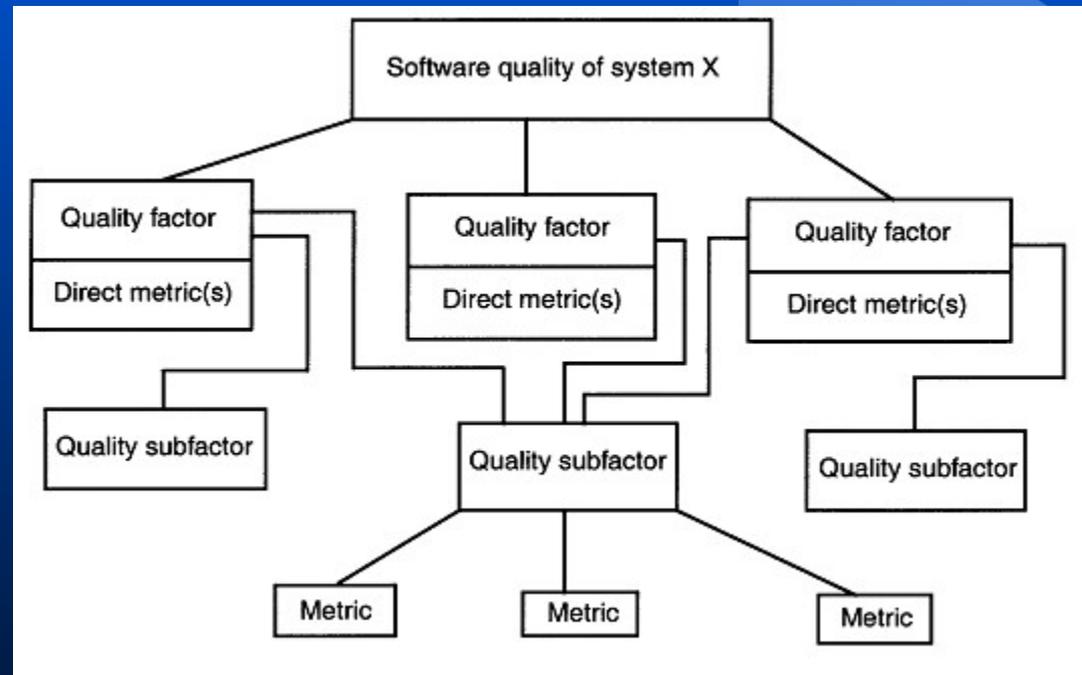
Remove ambiguity, Reduce uncertainty  
Raise confidence in decision-making

- ❖ *“Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys.*
- ❖ *For example, the specific parameters of customer satisfaction in software monitored by IBM include the CUPRIMDSO categories (capability / functionality, usability, performance, reliability, installability, maintainability, documentation / information, service, and overall);*
- ❖ *for Hewlett-Packard they are FURPS (functionality, usability, reliability, performance, and service).”*

– Stephen H. Kan, “Metrics and Models in Software Quality Engineering”

- ❖ Quality attributes
- ❖ Quality factors
- ❖ Quality subfactors
- ❖ Metrics
- ❖ Measures
- ❖ Indicators

## ❖ A Metrics Framework



IEEE Standard for a Software Quality Metrics  
Methodology IEEE Std 1061-1998

# Some Common Metrics?

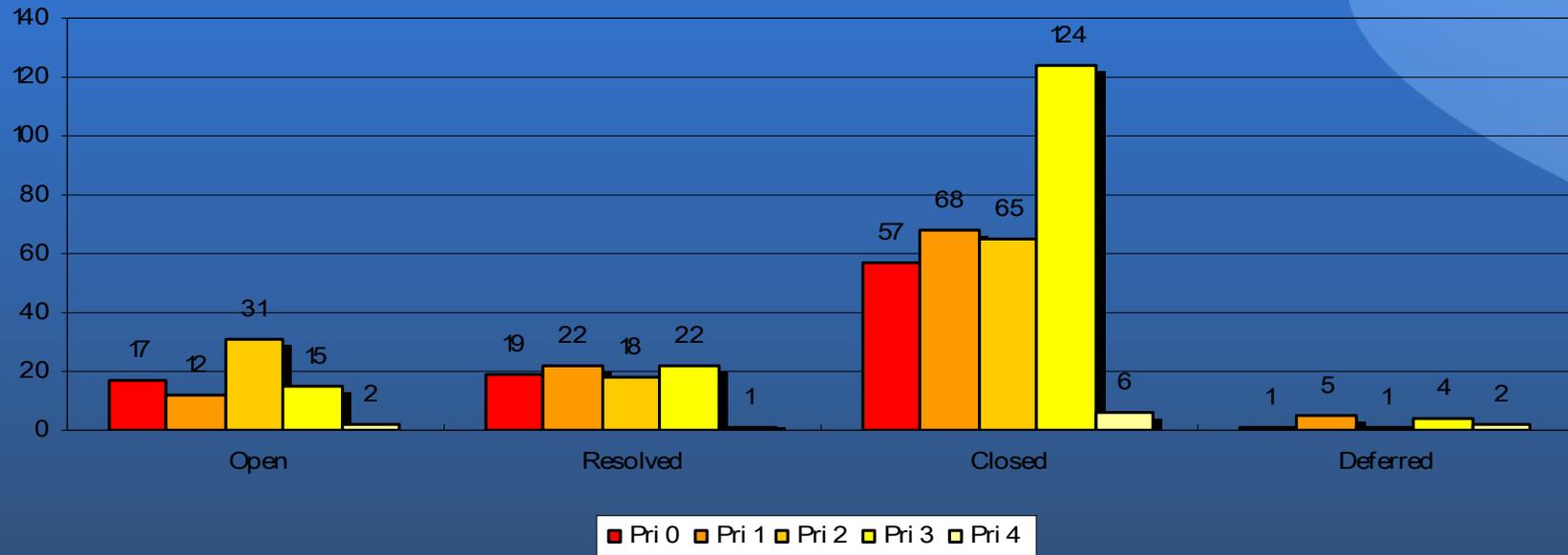
- ❖ Give examples for:
  - ❖ Product quality
  - ❖ Product performance
  - ❖ Schedule and progress
  - ❖ Resources and cost
  - ❖ Development process
- ❖ What questions are being answered?
- ❖ What decisions are being made?
- ❖ Should help answer:
  - ❖ How are we doing?
  - ❖ Do we have to change?
  - ❖ What are we accomplishing?
  - ❖ Are we doing what needs to be done?
  - ❖ Where can we be more efficient?
  - ❖ Where do we need training to do better?

Recommend reading:

“Metrics and Models in Software Quality Engineering”,  
2nd Edition, by Stephen H. Kan

# Useful or Not?

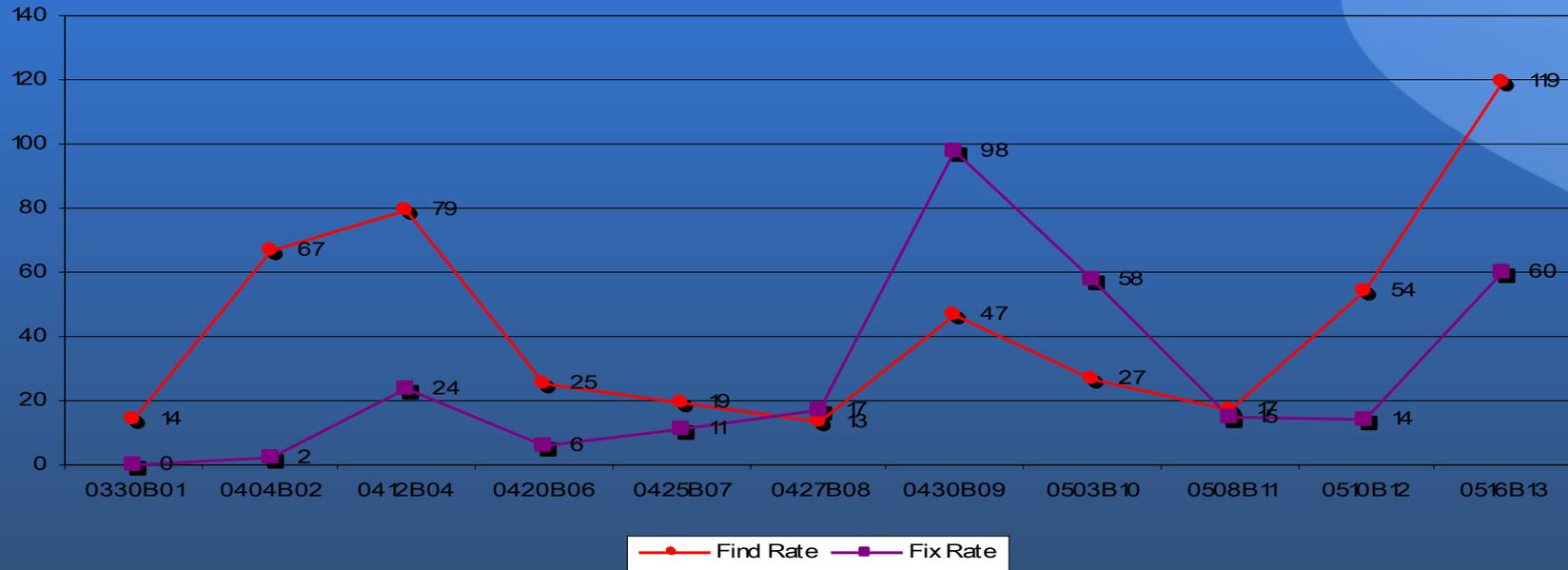
Number of Defects by Status and Priority



- ❖ What is the context? Where are we in the project? What events have happened to cause this picture? What is the change over time?
- ❖ To be useful, we need to know what questions/decisions are intended to be supported by this graph **and we need to tell the story that goes with the picture!**
- ❖ Consider: if you were driving a car is 'd', 'v', or 'a' the most useful thing to measure ...maybe you need all three? What about thresholds? What should 'a' be as you approach Point B (what if it is a wall?)

# Worth, or Needs, 1000 Words?

### Defect Find Rate Vs. Fix Rate

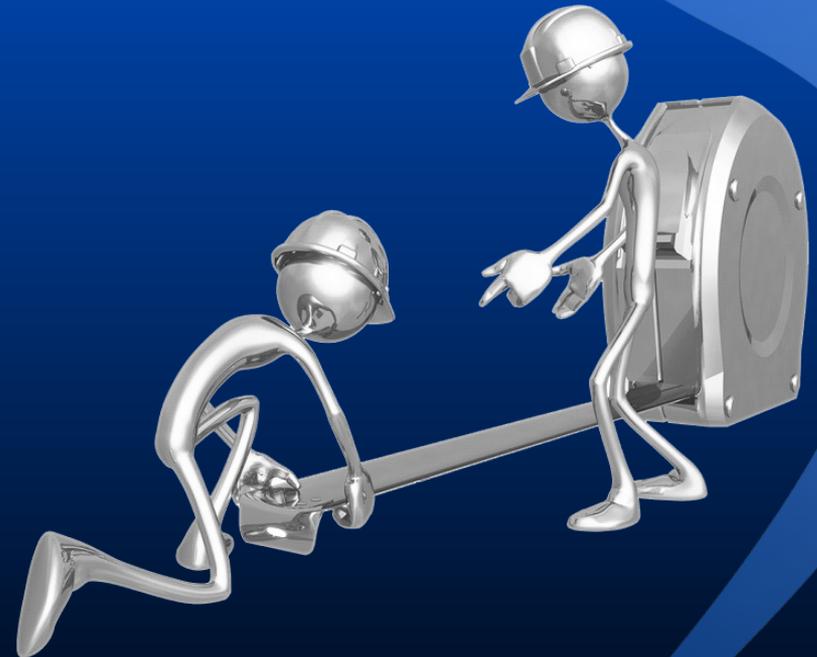


- ❖ This is a velocity type of graph that can tell a great story. **But it is open to (false) conclusions. Don't assume – Ask questions! Get the real story...**
- ❖ Can you see where Dev just closed a whole lot of bugs? (nope, didn't fix them) See the time between builds is not uniform so looks like numbers fell. Also, the May 8<sup>th</sup> build was only smoke tested. Finally, the numbers increase at the end of the graph because of change in the test execution vs. test case documentation effort ratio.
- ❖ **Danger:** collecting data without capturing the context and then presenting it “as is”...

# Measuring the Extent of Testing

How much testing have  
you got done?

How much more testing  
do you have to do?



Are we able to answer this question?

What is your answer based on?

# Right-click → Context Menu

- ❖ Consider these answers:
- ❖ Makes us wonder:

We've tested 60% of lines of code	How many requirements covered? What ratio of test cases executed? How many defects logged?
We've tested 80% of the test cases	How many requirements does that cover? What's left? # defects?
We've discovered 354 defects	How much functionality was tested? Are we stable? Severities? Clustering?
We've tested for 176 hours	How many test cases are done? How many remaining? What priorities? Are you on schedule?

- ❖ By themselves, **each answer is 1-Dimensional** – ambiguous – demanding more questions rather than giving answers
- ❖ If taken together, however, **a multi-dimensional story** could be told...

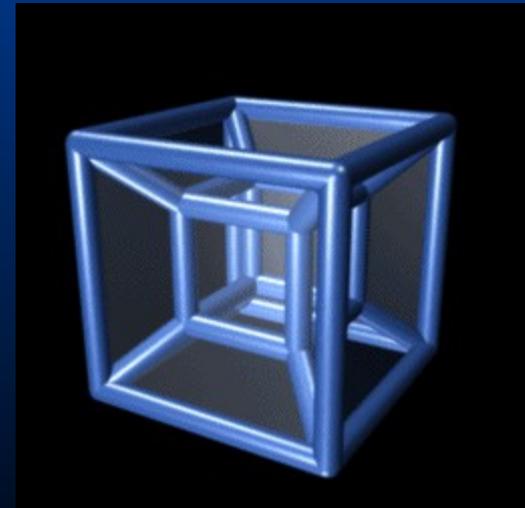
# Thinking in N-Dimensions

- ❖ 1 Dimension
- ❖ 2 Dimensions
- ❖ 3 Dimensions
- .
- .
- N-Dimensions

- ❖ How many correlated metrics do you need to support the decisions you need to make?
- ❖ Use 3 or more metrics for an increasingly complete version of the real story

- ❖ *“How many dimensions are needed to describe the universe is still an open question.”*

<http://en.wikipedia.org/wiki/Spacetime>



<http://en.wikipedia.org/wiki/Tesseract>

# Tips for Choosing Metrics

- ❖ Select practical metrics
  - ❖ Combine metrics to support answering real world questions and making timely decisions
  - ❖ Look for “quick wins” when getting started; what do you already collect?
  - ❖ Don’t rely on new metrics to save the day the first time out
- ❖ Metrics do change behaviour...
  - ❖ But not always the way you may hope
  - ❖ Heisenberg’s Uncertainty Principle
- ❖ Build a data/OLAP cube
  - ❖ Collect the data now for later analysis and use
- ❖ Read “Establishing Effective Metrics”
  - ❖ Guide to IEEE outline for defining a metric in detail



# Thinking Through Testing



For our latest updates:

- ❖ Visit [ThinkTesting.com](http://ThinkTesting.com)
- ❖ Follow [@ThinkTesting](https://twitter.com/ThinkTesting)

*Discussing "right-fit"  
approaches for  
software testing*

We are always sharing our ideas on crafting “right-fit” approaches to software testing. We are sure you will find something you can apply to your own projects and organizational environment.

*Thinking  
Through  
Testing*