Maximizing the Effectiveness and Efficiency of Software Inspections

SEPG 2005

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

How many of your organizations

- **are doing inspections?**
- **are collecting data?**
- **are analyzing the data?**
- **are using the data for corrective actions and process improvement?**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Rigorous Inspections

A review of a work product,

which passes stated entrance criteria,

led by a moderator who is not the author,

that seeks and records defects in that work product,

uses product specific checklists,

uses scenarios and/or other effective reading techniques,

initiates and monitors rework as necessary,

initiates re-inspection based on stated criteria,

passes or fails the work product based on exit criteria,

and adds to the base of historical data.

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Background

Material in this presentation resulted from work done by many different people in many different projects and organizations since 1974

© Software Technology Transition
JohnTHarding@compuserve.com

Inspection Effectiveness

- Percentage of the defects in the work product being identified
- Teams doing rigorous Inspections can typically start around 50%
- Can go as high as 90+%

© Software Technology Transition
JohnTHarding@compuserve.com

Inspection Efficiency

The business value

The hours per defect

© Software Technology Transition
JohnTHarding@compuserve.com

Why Do Rigorous Inspections?

A business decision

To remove defects at the point of lowest cost

- **Requires "cost data" for each defect removal activity**
  - **Use hours per major defect for inspections and test**
  - **If no hard data use an approximation**

    a Delphi type of data gathering

    divide total test activity effort hours by defects found

You need to make the comparison visible to support this decision

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Minimize “Bureaucracy”

When defining an inspection process

- **Keep everything as simple as possible**
- **Do not collect any data you will not use/analyze**
- **Ensure people understand how the data will be used/analyzed before collecting it**
- **Ensure that techniques critical to effective inspections are properly included (e.g., how to read, mapping to predecessor work product)**

This will ensure a reasonable degree of efficiency

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Where Should You Initially Focus?

Focus on effectiveness first

- **Why make a process which is less effective than desired more efficient?**

But where will many managers want to focus first?

© Software Technology Transition
JohnTHarding@compuserve.com

Effort To Inspect 1kloc of Code

Assume

- **Modification to existing code and uplift factor is 5**
- **Optimum rate is 100 lines of executable code per hour**

Effort required

- **1kloc + 5 kloc = 6 kloc to inspect**
- **Preparation time is 60 hours**
- **Inspection time is 60 hours**
- **Total effort required for each participant is 120 hours!**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Key Enablers For Rigorous Inspections

Champion

- **Volunteer not a draftee!**

Metrics Policy

- **Sample at end of presentation**
- **Must have an Ombudsman**

Capacity plans

Training

Data definitions

- **Especially for loc and major defect**

Checklists

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Areas for focus

Planning and scheduling inspections

- Capacity planning and schedule refinement
- Just in Time (JIT) code inspections
- Selection criteria

Performing Inspections

- Mapping
- Reader role and use of scenarios
- Product and domain specific checklists

Data analysis and use

- Stages to go through
- Some key analysis and uses
- Some key data elements and granularity needed

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Planning and Scheduling

© Software Technology Transition
JohnTHarding@compuserve.com

Capacity planning and schedule refinement

First ensure time for activities is allocated into overall plan

- **Use data to determine amount of time to block out**
- **How much will be written?**
- **What % is the target for inspecting?**
- **If it is a modification of existing software what is the uplift factor?**
- **If you do not have optimal rate data for you project**
  - **Start with reasonable rates**

    Requirements: 3 to 12 pages per hour

    High Level design: 3 to 8 pages per hour

    Low Level Design: 100 to 200 pseudo code lines per hour

    Code: 100 to 150 lines of code per hour
- **How many attendees? Who? Allocate the time!**

As you get close to completing various pieces continuously schedule specific inspections using the selection criteria

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Inspection timing and unit test

Do you inspect before or after unit test?

There are two questions which can help the technical people decide the right answer for their product/project

- **What are the hours per defect for Inspection and unit test?**
  - **Which is cheaper?**
- **If unit test is lower what are the attributes of the defects found in Inspections and unit test?**
  - **If unit test finds code problems and Inspection finds both code and design problems which should be done first?**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Just in Time (JIT) code inspections

- Do capacity planning
- Block out time each day for preparation and the inspection meeting
- Select code to be inspected 1-5 days prior to its being completed
- Prepare for the Inspection the afternoon it is completed
- Hold the inspection meeting the next morning
- Works best when code is being written in incremental “chunks”

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

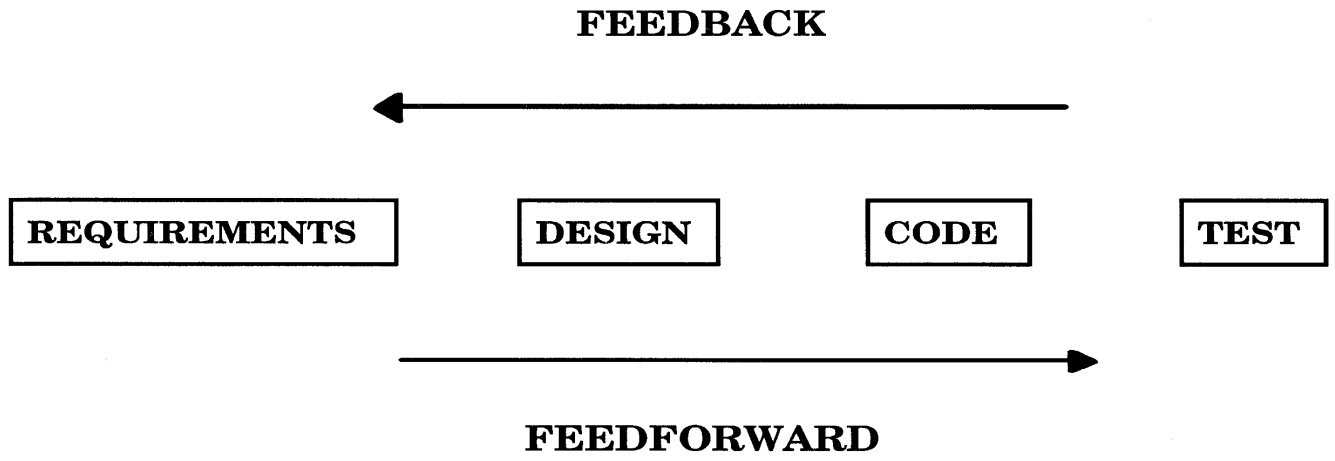
© Software Technology Transition
JohnTHarding@compuserve.com

Selection criteria

If you can not inspect everything consider two separate criteria

- Critical function, especially from an end user perspective
- Defect density; go where the bugs are

Consider using the notion of “feed forward” along with historical defect density



Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Performing Inspections

© Software Technology Transition
JohnTHarding@compuserve.com

Mapping

There are two types of defects

- **Something does not work**
- **Something works but is not what was intended or needed**

By using documents or other appropriate work products created earlier in the life cycle the Inspections can

- **Ensure that the refinement accurately reflects what was intended**
- **Ensure that any necessary interpretation was consistent and accurate**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Reader role and use of scenarios

The reader would lead the team through the material during the Inspection meeting

The material can be reviewed

- **Sequentially**
- **Functionally**
- **Driven by design flows**
- **Driven by test scenarios or use cases**
- **The material can be**
  - **"Read" directly (when justified)**
  - **Paraphrased (most typical approach)**
  - **Both**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Product and domain specific checklists

- Do all products have the same types of defects?
- Use checklists to help ensure consistency across Inspections
  - Try to limit checklists to 1 page
- List common or frequently occurring problems and ensure that they are always checked
  - Need to be specific for your products and your environment
- List any specific critical items for your project or domain for the inspectors to check for
- Direct relationship with data analysis (e.g., Pareto charts)

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Data Analysis and Use

© Software Technology Transition
JohnTHarding@compuserve.com

Stages to go through

1 of 2

Have clear definitions for key data elements

- Size, preparation time, Inspection time, Major and Minor defects, number of inspectors

Identify inconsistencies in recorded data and take corrective actions; do this as you start to collect data…do not wait!!

- Requires that you use the data you collect!

Identify the key ratios you want to use and train people on how to calculate and use them

- Amount of material per hour (e.g., loc/hour) for preparation and the Inspection meeting
- Defect density; major defects/size (e.g., defects/page)
- Hours/major defect; can be considered the "cost"
  - As long as this is less than test doing Inspections makes good business sense!

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Stages to go through

2 of 2

Analysis and use

- **Common sense!**
- **Use basic data and ratios for planning**
- **Use basic data and ratios for tracking**
  - **Compare to prior similar projects**
  - **Compare to industry data**
- **Use simple scatter charts for optimization**
- **Use Inspection data to forecast test and customer defects**
- **Put inspections (as appropriate) under SPC**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Who does the analysis?

- The technical people must be involved
- Who else can explain what the numbers really mean?

© Software Technology Transition
JohnTHarding@compuserve.com

Inspection optimization

Consider optimizing inspections before stabilizing the process (SPC)

Start with scatter charts; if appropriate use linear regression

Ensure appropriate rationale sub grouping

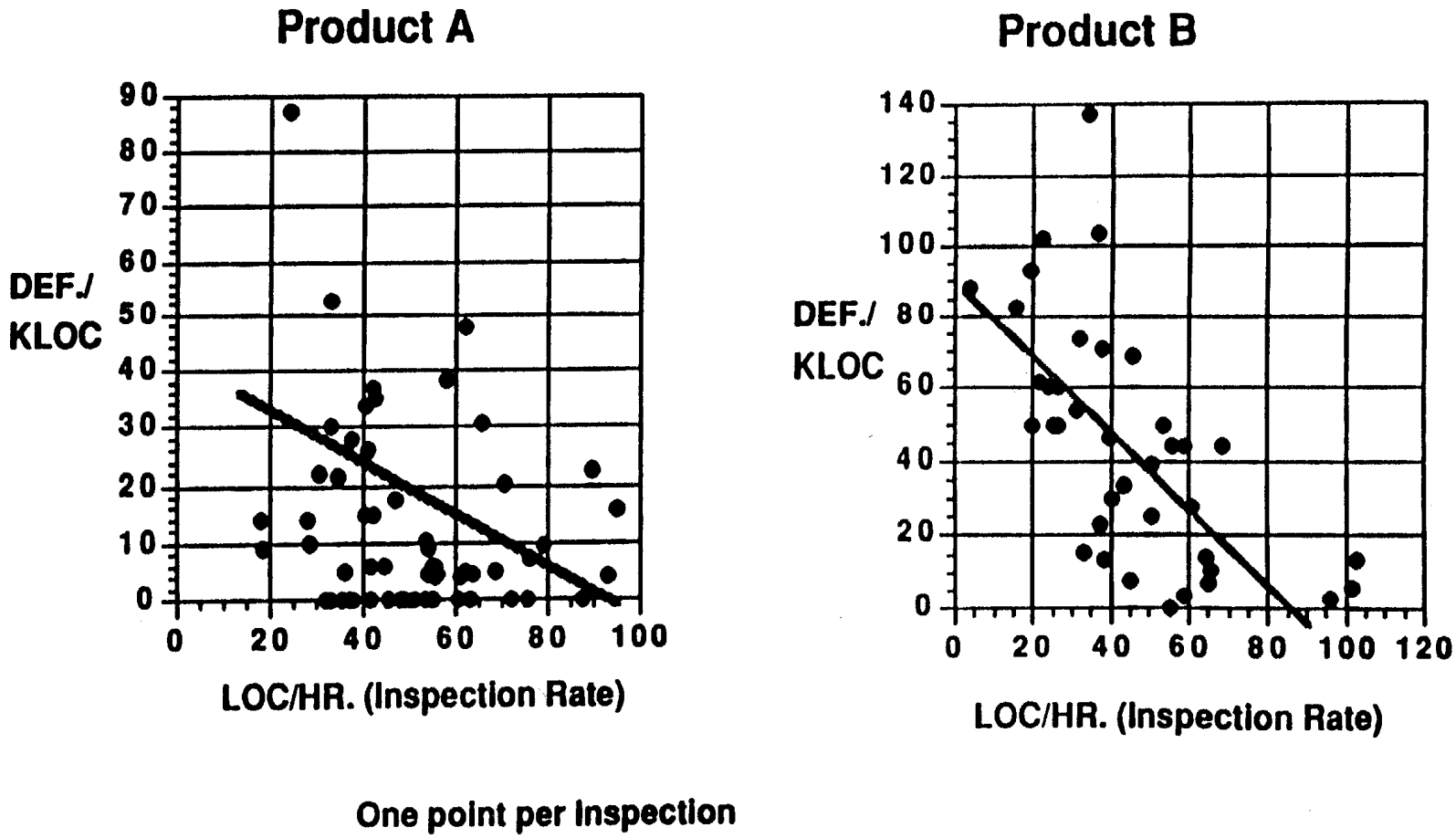
Some typical scatter charts

- **LOC/hour vs. Major defects/KLOC**
- **LOC/hour vs. Hours/Major defect**
- **Segment these two charts by size**
- **Segment these two charts by number of inspectors**
- **Look for other relationships**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Correlation Analysis



One point per Inspection

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

LET'S USE CODE INSPECTION

DATA TO HELP US UNDERSTAND

THE NUMBER OF DEFECTS

REMAINING IN THE PRODUCT

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Project Information Available

Size of product

- **10,000 lines of code**

Line of code inspected

- **2,000**

Major Defects found

- **50**

Inspection data

- **consistent and reasonable rates**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Industry Information Available

Inspection effectiveness can go up to 90%

- **Typically requires several years of evolution**

Groups starting to do consistent inspections typically start at around 50% effectiveness

- **Rarely over 60% for groups just starting**
- **Rarely are they ever less than 40%**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Let's Assume We Are 50%

Size of product

- 10,000 lines of code

Line of code Inspected

- 2,000

Major Defects found

- 50

Inspection defect density is

- 25 defects/kloc

Product defect density is 50 defects/kloc

- 25 x 2 because we are assuming 50% effective

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

How Many Defects Remain?

Size of product

- **10,000 lines of code**

Major Defects found

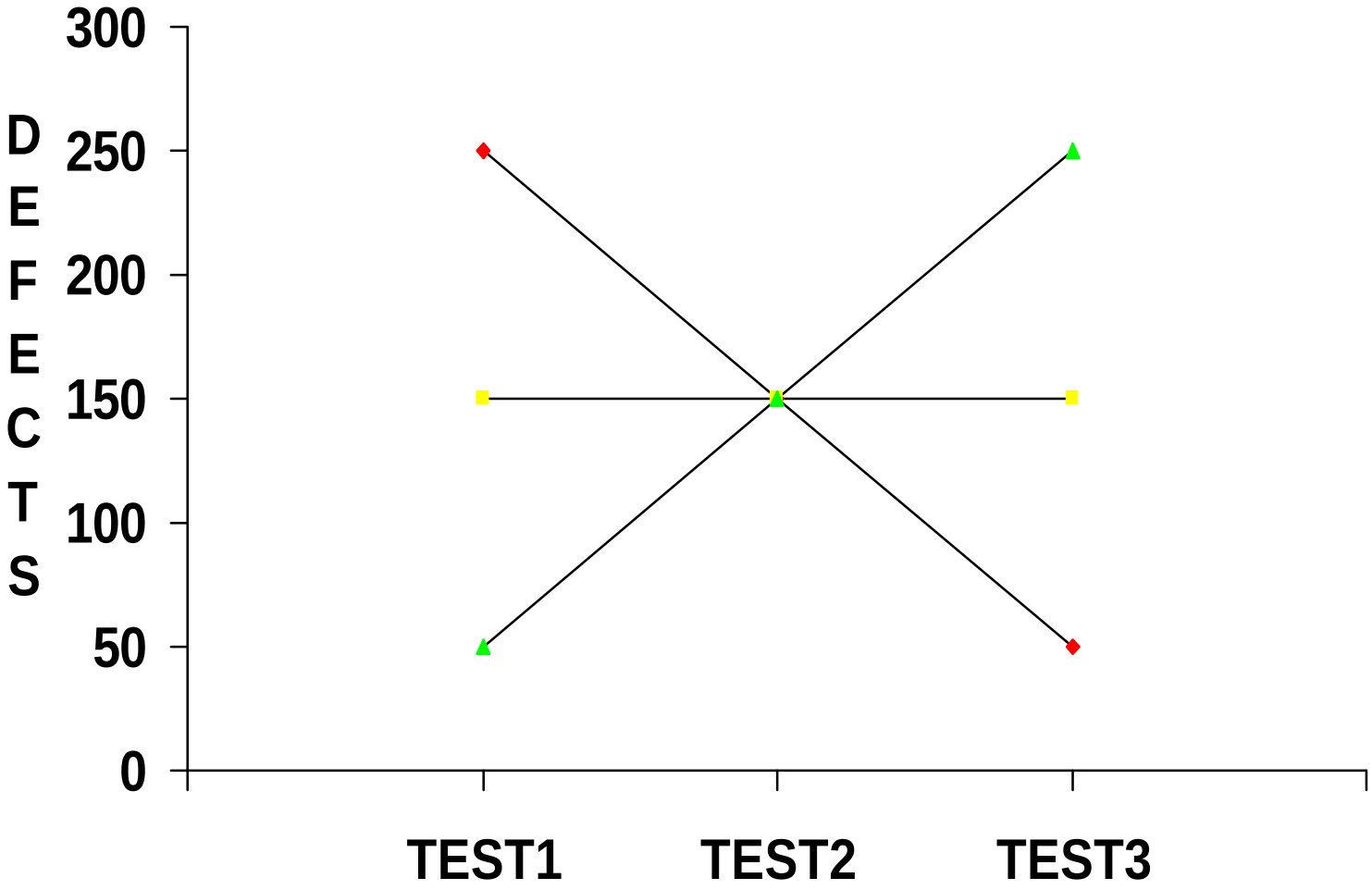
- **50**

There are 450 defects remaining

- **10 kloc x 50 defects/kloc = 500 defects**
- **but we found 50 already in Inspections (for the 2,000 lines inspected)**

© Software Technology Transition
JohnTHarding@compuserve.com

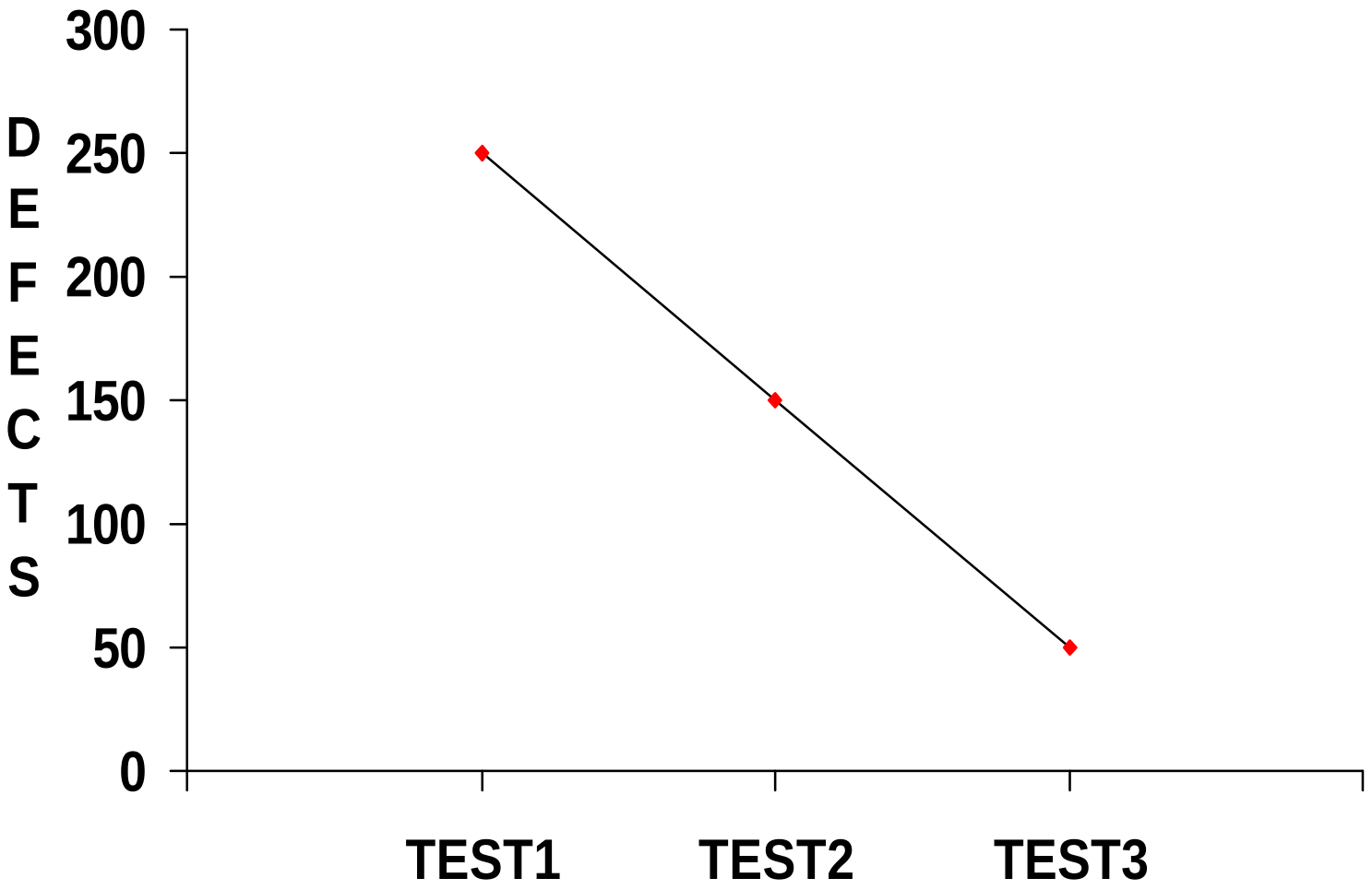
How Do You Want To Find Defects During Test?



Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Target Defect Count By Test Activity



Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Create Defect "Ranges"

What we just did was the 1st step

Go back and do this for 40% and then 60%

- **this gives you an upper and lower range**

Ask yourself if the defect density of the code which was inspected is proportional to the defect density of the product

- **if not then you need to do each calculation separately for the low density and high density parts of the product; this is explained in the paper**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Some key data elements and granularity needed

Size

Major defects

© Software Technology Transition
JohnTHarding@compuserve.com

SIZE (e.g., Lines of Code)

Keep separate counts

- New and changed
- Existing "surround" code

Start by counting physical lines

- Include Data and Complier directives which change object code

Allows you to understand how much of the new and changed code has been inspected

The amount of "surround code" will vary

- A lot of small changes in many source modules
- One big change
- By product or environment

© Software Technology Transition
JohnTHarding@compuserve.com

Major Defects

Need to be defined so they can be compared to test and customer defects

Count major defects separately

- One count for "surround" code
- Another count for new and changed code

Allows you to:

- Understand where the errors are coming from
- Calculate defect densities for both

They may be quite different

A necessary step to enable true Defect Prevention

- Defects from earlier releases may be due to a process different from the current process!

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

Defect Density for Fixes

Fixes are frequently small

- **Defect density numbers may be very large**
- **Defect density numbers may have large variability**

The real process which needs to be controlled may be the % of fixes which have an error

Consider using % fixes with an error as a measure of your fix defect density

© Software Technology Transition
JohnTHarding@compuserve.com

References

Burr, Adrian and Mal Owen; Statistical Methods For Software Quality; Thomson Computer Press (UK); ISBN 1-85032-171-x

R. G. Ebenau and S. H. Strauss; Software Inspection Process; McGraw Hill, Inc; ISBN 0-07-062166-7

IBM Systems Journal; Volume 15 Number 3;1976

Florac, William A., Robert E. Park, Anita D. Carleton; Practical Measurement: Measuring for Process Management and Improvement; CMU/SEI-97-HB-003 (available on SEI web page)

T. Gilb and Dorothy Graham; Software Inspections; Addison Wesley; ISBN 0-201-63181-4

Robert B. Grady and Deborah L. Caswell; Software Metrics: Establishing A Company Wide Program; Prentice Hall; ISBN 0-13-821844-7

Robert B. Grady; Practical Software Metrics For Project Management And Process Improvement; Hewlett-Packard Professional Books; ISBN 0-13-720384-5

Harding, J.T.; Using Inspection Data to Forecast Test Defects; CROSSTALK; Volume 11 Number 5, May 1998

Kan, S.H.; Metrics and Models in Software Quality Engineering; Addison-Wesley; ISBN 0-201-63339-6

Radice, R.A.; High Quality Low Cost Software Inspections; ISBN 0-9645913-1-6

E. F. Weller; Lessons From Three Years of Inspection Data; IEEE Software; September 1993, pp 38-45

E. F. Weller; Using Metrics to Manage Software Projects; IEEE Computer; September 1994, pp 27-33

E. F. Weller; Practical Applications of Statistical Process Control; IEEE Software May/June 2000, pp 48-55

Web site reference: url to Philip Johnson's FTR website - he has over 200 books/papers identified. http://www2.ics.hawaii.edu/~johnson/FTR/

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition

JohnTHarding@compuserve.com

Sample Metrics Policy

It is the goal of XX to continuously improve in the ways we develop our products and to grow our business. In order to understand our effectiveness in meeting these goals we need to define a set of metrics that will allow us to measure the important aspects of our software development process related to quality, productivity, and costs.

While we increase our use of software metrics and data, it is important to note that:

- **Metrics and data within projects and products will be used to help define and manage to established goals and to evaluate for rates of improvement during the development process. Project and process data will be treated as an asset, as an investment for our future progress, and as a critical factor for our success.**
- **Metrics and data can be used to understand differences between projects or products, to reward what is done well, and to learn what may be required to improve our business.**
- **The collection, recording, and reporting of metrics are intended to measure and to help us improve the processes we use during the development of a product, not to evaluate the performance of individuals. All metrics will be collected and combined such that they are personally non-threatening.**

Software Technology Transition
60 Elm Street
Andover, MA 01810
978-475-5432

© Software Technology Transition
JohnTHarding@compuserve.com

John (Jack) Harding

JohnTHarding@compuserve.com

www.stt.com

© Software Technology Transition
JohnTHarding@compuserve.com