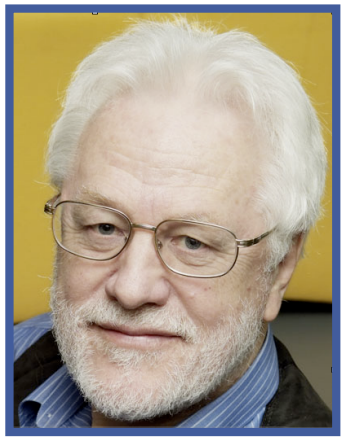


Tom Gilb

Inspection Facts:

to help you make good decisions to do Software Inspections and Reviews properly

- JUSE Tokyo September 2008
- Tutorial: 60min x 2 (continuous sessions before and after lunch)
- by Tom Gilb

•Sources: Gilb, Competitive Engineering (2005),

•Gilb and Graham, Software Inspection (1993):

•Japanese Edition

kyoritsu-pub.co.jp

Management Data About Inspections and Reviews

- **Main Objective:**
  - **to prepare management**
    - **to make decisions**
    - **regarding the evolution of their organization**
      - **in the direction of *zero defects***
      - **using the classical Software Inspection Process.**

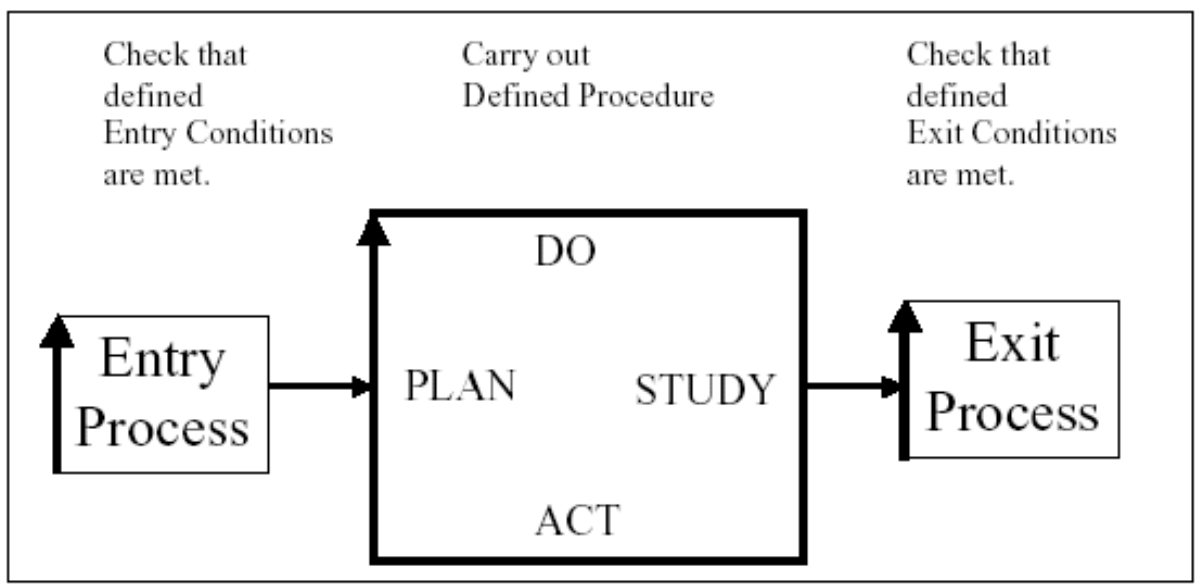
-

Why should Management be interested in having Software Inspections?

- Not to ‘clean up’ bad work.
  - because this is ineffective (50%)
  - and there are more cost effective alternatives to the ‘real goal’ (quality)

- Inspection *should*
  - Measure the level of major defects in work
  - use this measured level to exit good work
    - And to fail, exit to next process, for bad work
  - and thus *motivate* people to learn to do good work
    - Meaning “to follow our official standards” (rules).

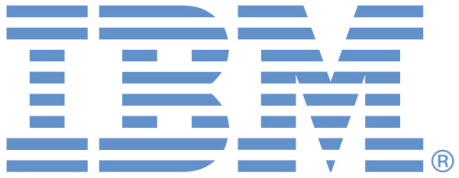
The quantified Exit and Entry controls



- Entry and Exit Condition example:
- Maximum estimated 1.0 Major defects per logical page remaining.
- This was the MOST important lesson IBM learned about software processes (source Ron Radice, co-inventor Inspections, Inventor of CMM)

M1. Real-world case studies of defect reduction, and how this is achieved using Software Inspection.

- Raytheon
- Datastream Primark
- Citigroup
- A-sim project Ericsson
- others from cases



Great Case study book of software change (similar to Raytheon story)

- Craig Kaplan et al
- Secrets of Software Quality
  - 40 innovations from IBM
  - McGraw Hill
- about 1995, maybe out of print but used copies at amazon.com $2.99

- See Gilbs set of Kaplan slides!
  - ckaplan@iqco.com
  - http://www.iqco.com/

Software Process Improvement at Raytheon

- Source : Raytheon Report 1995
  - http://www.sei.cmu.edu/pub/documents/95.reports/pdf/tr017.95.pdf
  - Search "Dion & Raytheon"
- An excellent example of process improvement driven by measurement of improvement
- Main Motor:
  - "Document Inspection", Defect Detection
- Main Driver:
  - "Defect Prevention Process" (DPP)

Cost of Quality over Time: Raytheon 95

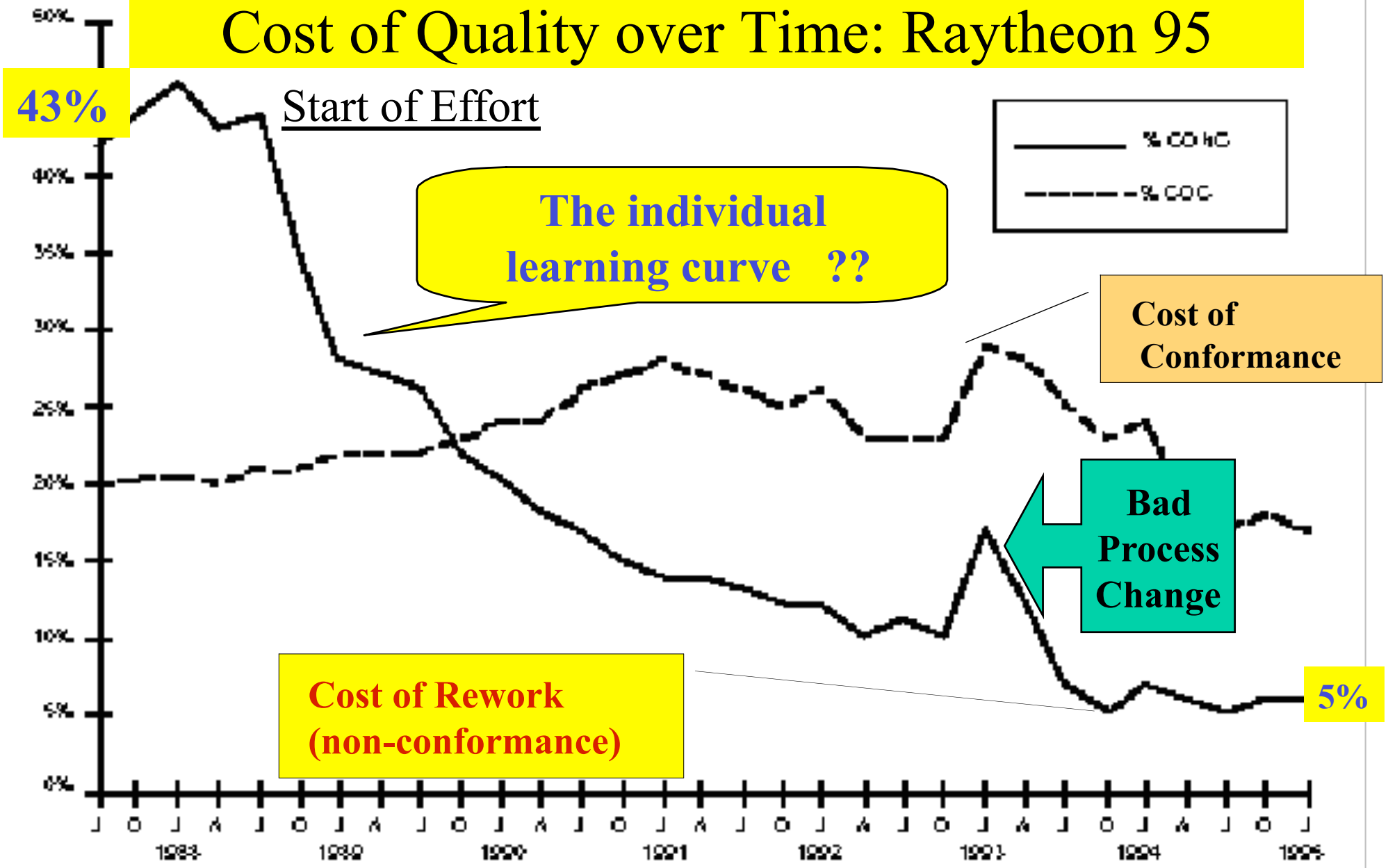


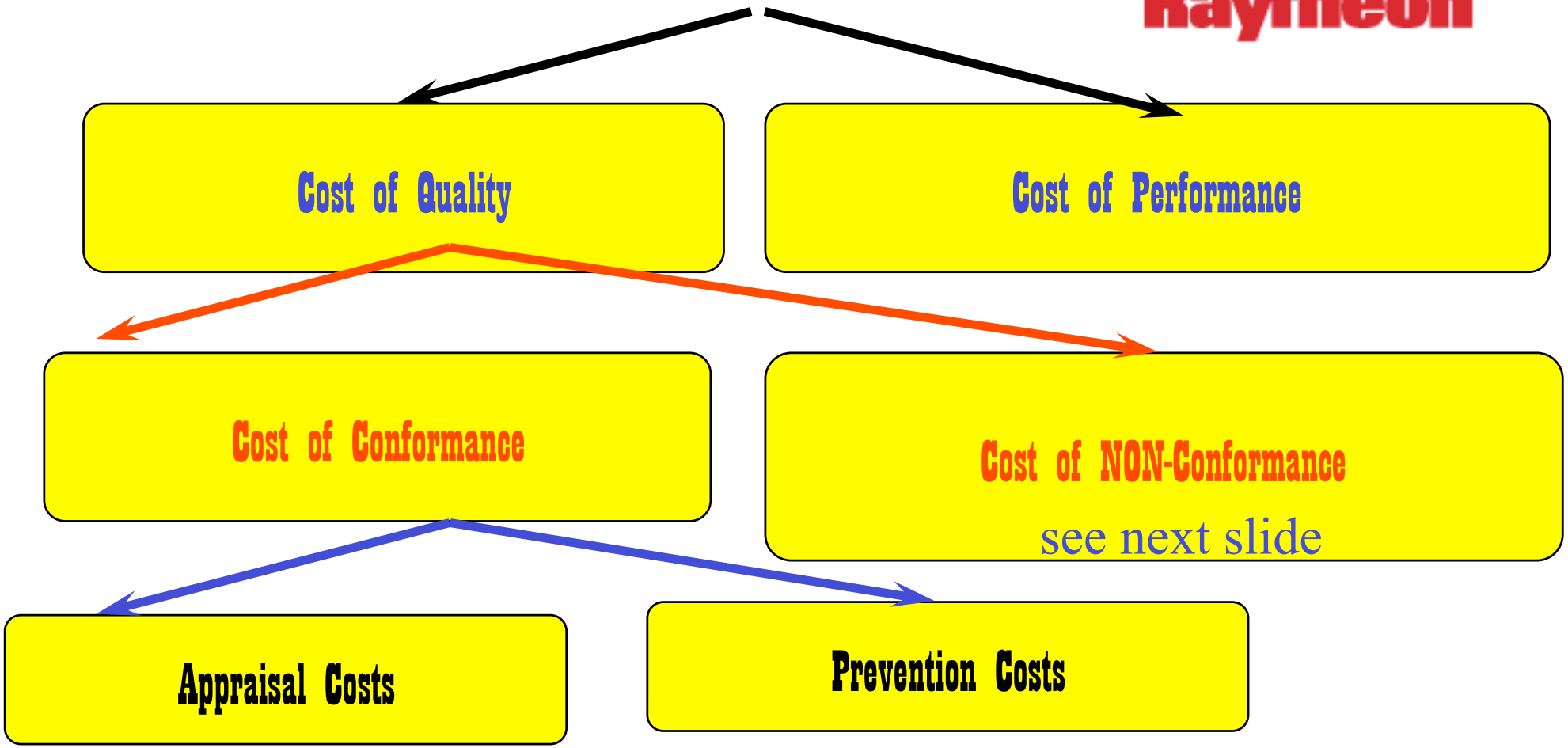
Figure 8: Cost of Quality Versus Time

Rework Cost: Making a plan

- **Ambition:**
  - **reduce by-half wasted development effort due to avoidable errors, if process improved.**
- **Scale:**
  - **% of total effort which is applied to handling {identifying, correcting, re-testing, reissuing} avoidable errors.**
- **Past [Our test process, 2008]**
  - **45%**
- **Goal**
  - **[Us, 2009 end] 30%,**
  - **[End 2010] 20%,**
  - **[End 2011] 10%,**
  - **End 2011] 5%**

Project Cost

Raytheon



Costs of Non-conformance Items

Raytheon

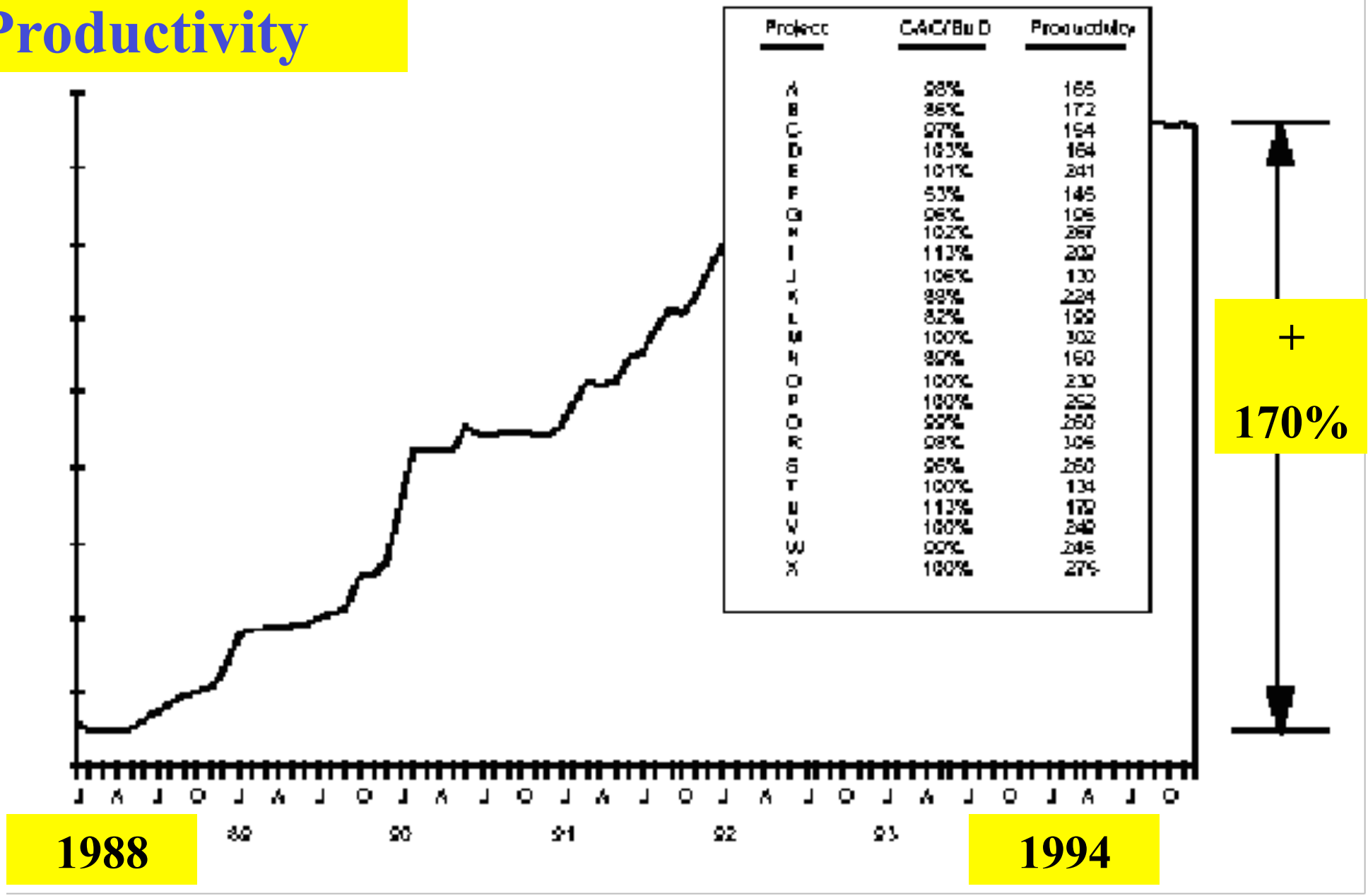
- Re-reviews
- Re-tests
- Fixing Defects (code, documentation)
- Reworking any document.
- Engineering Changes
- Lab Equipment Costs of Retests

- Updating Source Code
- Patches to Internal Code
- Patches to Delivered Code
- External Failures
- from Crosby's Model according to Raytheon95 Fig. 7

Raytheon 95 Software Productivity 2.7X better

Raytheon

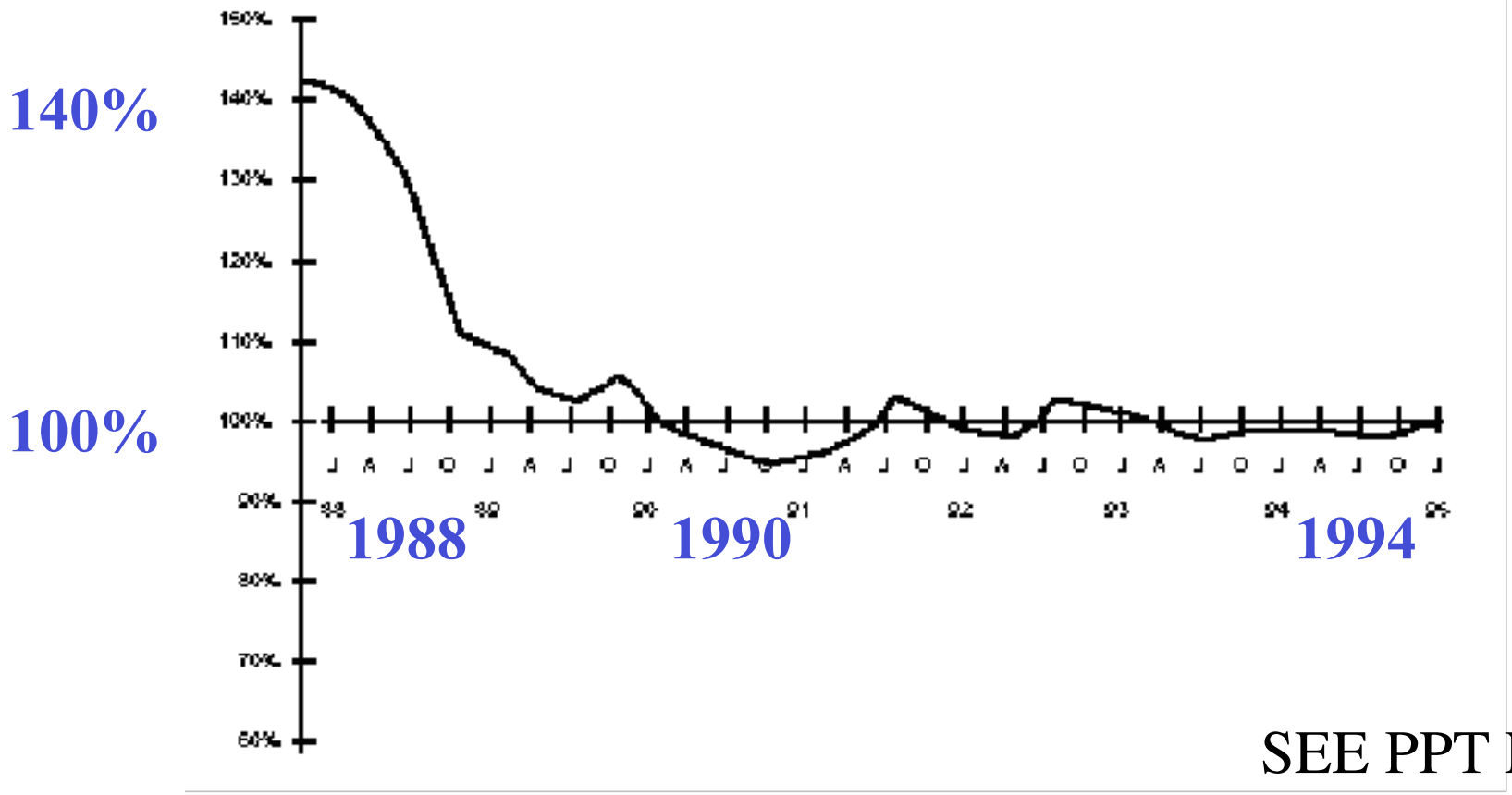
Productivity



Achieving Project Predictability: Raytheon 95

Raytheon

Cost At Completion / Budget %



SEE PPT NOTE FOR DEFINITION.

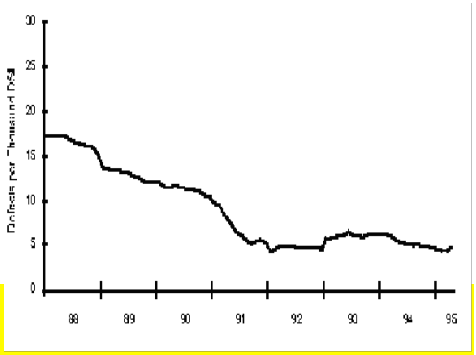
Examples of Process Improvements: Raytheon 95

- **Process Improvements Made**

Raytheon

- **Erroneous interfaces during integration and test** -
  - Increased the detail required for interface design during the requirements analysis phase and preliminary design phase - Increased thoroughness of inspections of interface specifications
- **Lack of regression test repeatability -**
  - Automated testing - Standardized the tool set for automated testing - Increased frequency of regression testing
- **Inconsistent inspection process -**
  - Established control limits that are monitored by project teams - Trained project teams in the use of statistical process control - Continually analyze the inspection data for trends at the organisation level
- **Late requirements up-dates** -
  - Improved the tool set for maintaining requirements traceability - Confirm the requirements mapping at each process phase
- **Unplanned growth of functionality during Requirements Analysis**
  - - Improved the monitoring of the evolving specifications against the customer baseline - Continually map the requirements to the functional proposal baseline to identify changes in addition to the passive monitoring of code growth - Improved requirements, design, cost, and schedule tradeoffs to reduce impacts

Overall Product Quality: Definition at Raytheon

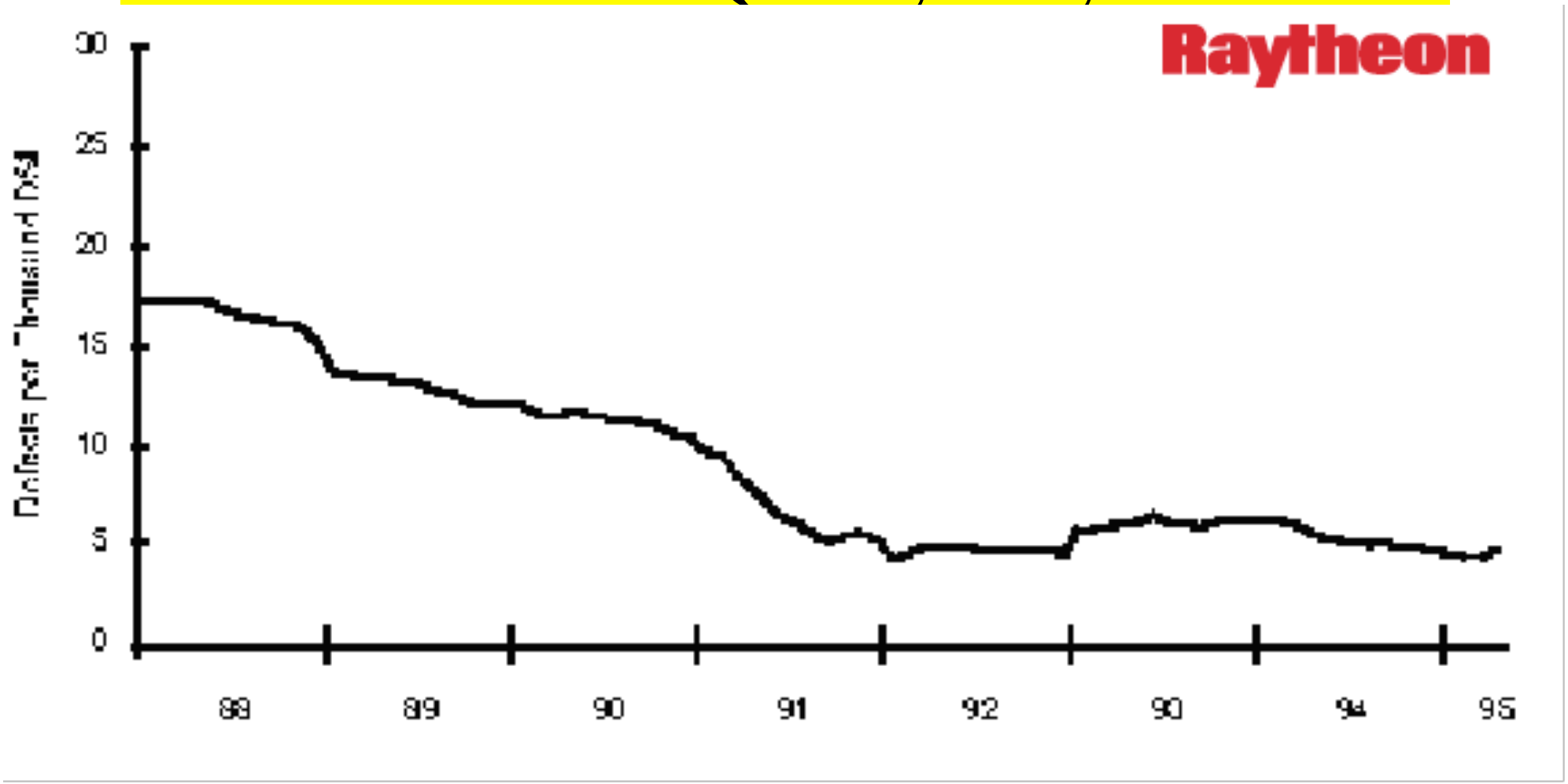


- Overall Product Quality

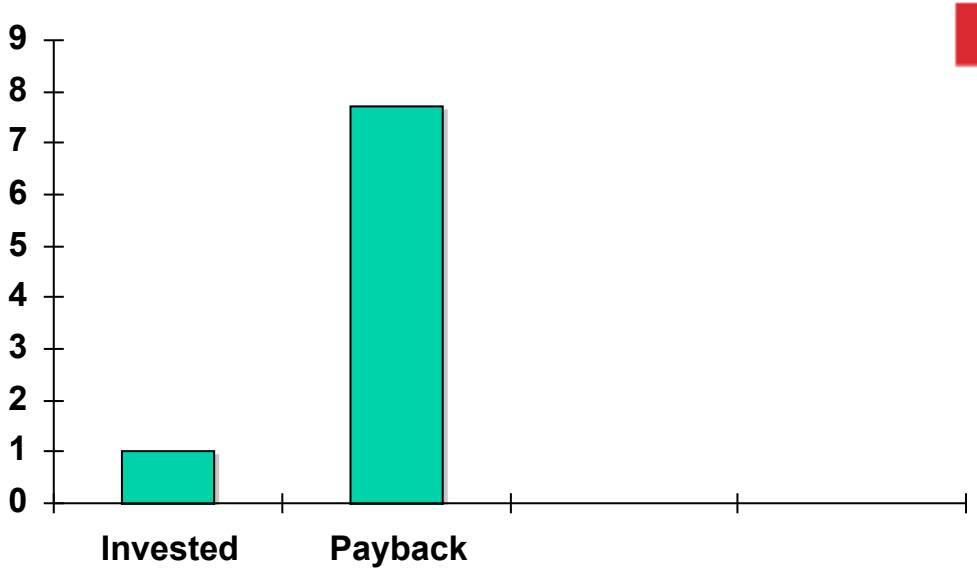
Raytheon

- The primary measure used to assess overall product quality is the **defect density** in the final software products.
- We measure this factor in **"number of software trouble reports (STRs) per thousand lines of delivered source code** (STRs/KDSI)" on an individual project basis.
- The project defect densities are then combined to compute the monthly weighted average (using the same approach as the cost of quality described above) thus yielding a time -variant plot of our overall product quality measure.
- As shown in next slide, data collected over the period of the initiative shows an ***improvement*** from an average of **17.2** STRs/KDSI to the current level of **4.0** STRs/KDSI.

Overall Product Quality: Raytheon 95



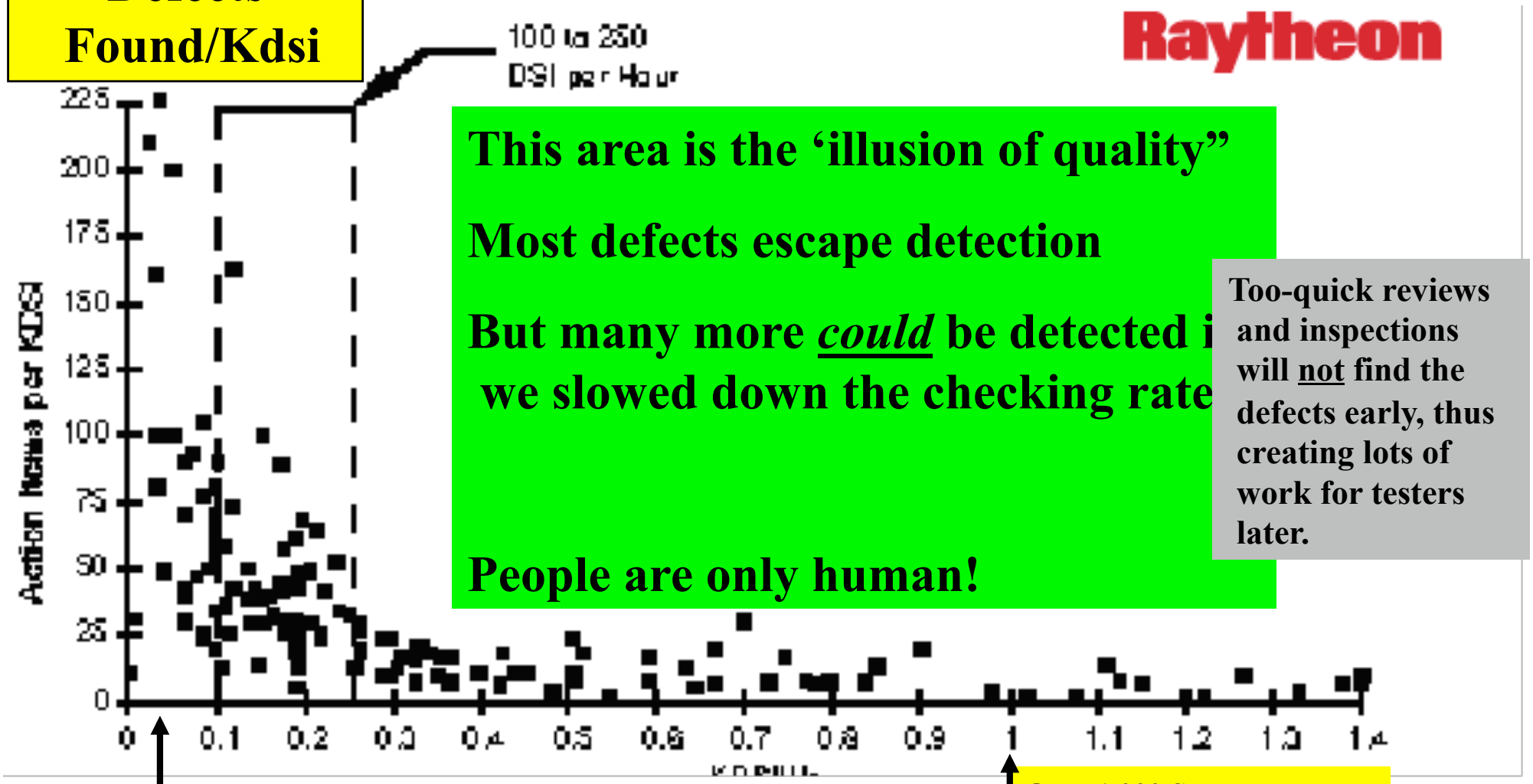
Return On Investment at Raytheon about $10,000 per programmer/year



- $7.70 per $1 invested at Raytheon
- Sell your improvement program to top management on this basis
- Set a concrete target for it
  - Goal [Our Division, 2 years hence] 8 to 1

Fault Density versus Checking Rate: Raytheon 95

Defects Found/Kdsi



Raytheon

This area is the 'illusion of quality"

Most defects escape detection

But many more *could* be detected i we slowed down the checking rate

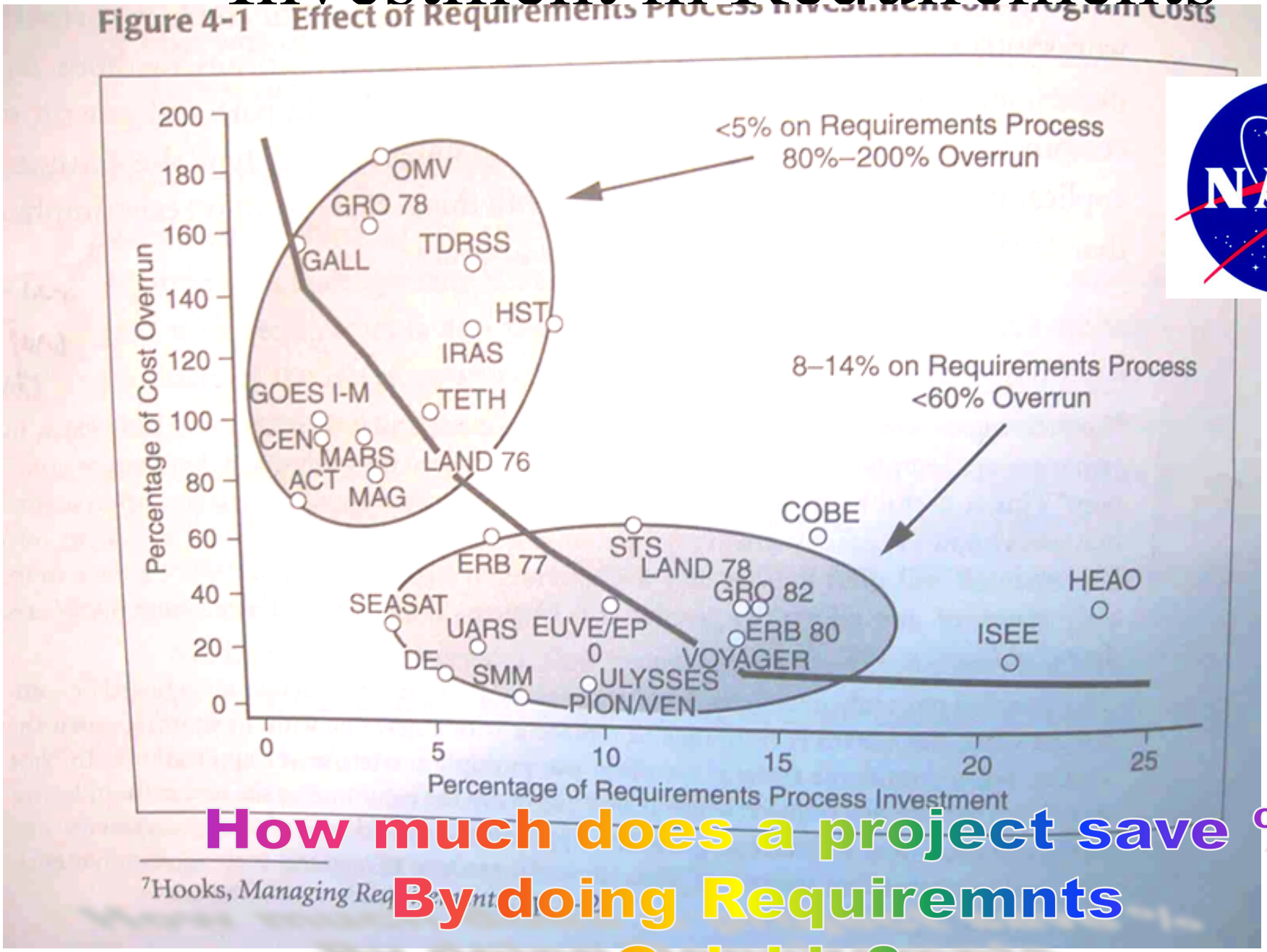
People are only human!

Too-quick reviews and inspections will not find the defects early, thus creating lots of work for testers later.

Real Optimum Checking Rate

Over 1,000 Statements Checked per hour by a single checker

Investment in Requirements



How much does a project save % By doing Requiremnts Quickly?

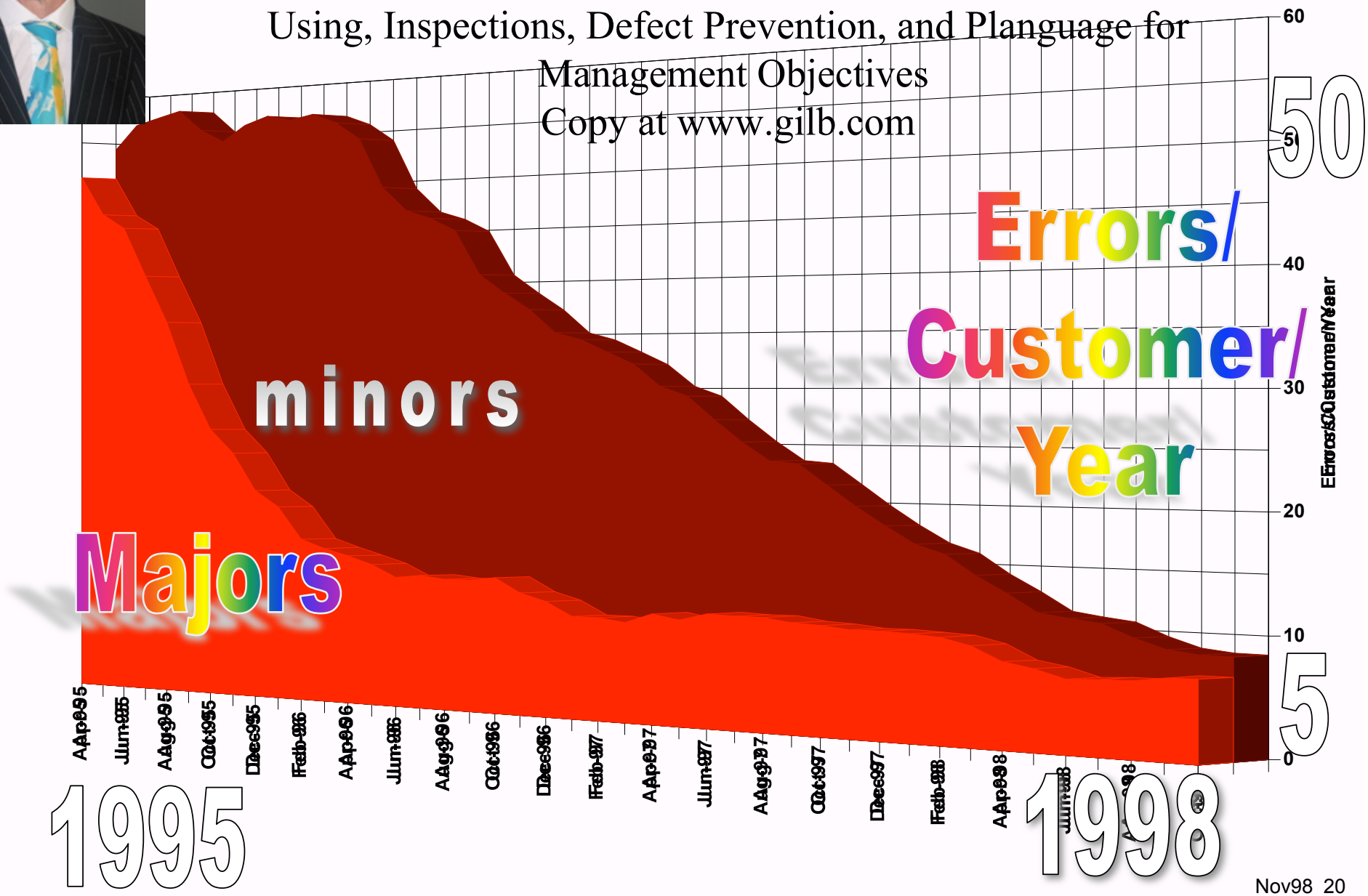
Improving the ***Reliability*** Attribute

Primark, London (Gilb Client)

see case study Dick Holland, "Agent of Change" from Gilb.com

Using, Inspections, Defect Prevention, and Planguage for Management Objectives

Copy at www.gilb.com



Defect Rates
in 2003 Pilot Citigroup, London, Gilb Client
Spec QC/Extreme Inspection + Planguage Requirements

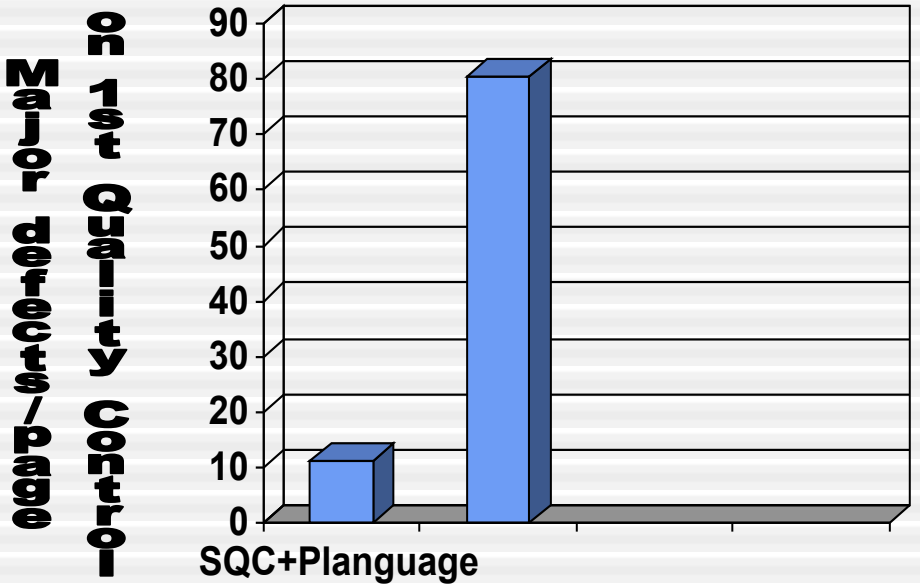
Across 18 DV (DeVelopment) Projects using the new requirements method, the average major defect rate on first inspection is 11.2.

4 of the 18 DV projects were re-inspected after failing to meet the Exit Criteria of 10 major defects per page.

A sample of 6 DV projects with requirements in the 'old' format were tested against the rules set of:

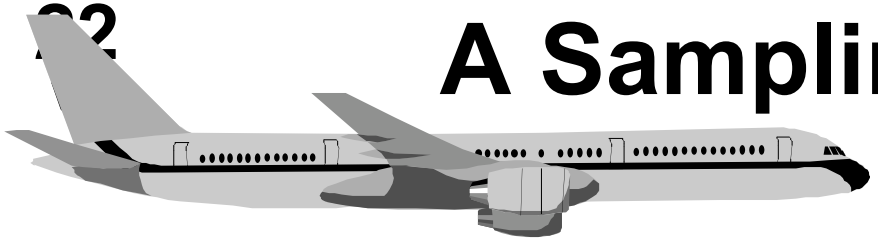
- The requirement is uniquely identifiable
- All stakeholders are identified.
- The content of the requirement is 'clear and unambiguous'
- A practical test can be applied to validate it's delivery.

The average major defect rate in this sample was 80.4.



citigroup

A Sampling Case Study



- **1986 Sweden -> Germany**
  - **Air traffic control trainer system for export**
  - **80,000 pages contracted documentation before code**
  - **40,000 pages already written**
  - **Project seriously late already (customer informed)**
  - **About 7 management signatures approving the 40,000 pages (pseudocode for coders)**
  - **Inspection of a sample of three pages**
    - **chosen by random numbers**
    - **declared to be representative**
    - **19 Major defects found in half day inspection by the 7 managers**
    - **director checks the defect log and confirms**

ERICSSON

The experience of Industry.

- Using Inspection to cleanup bad code
  - Is uneconomic, too late
- Using Inspections to go through entire large documents
  - Is uneconomic (does part of the job at a cost we are not willing to pay)
  - Is ineffective
    - 3% [usual malpractice!] of defects actually there
    - to 30% [reasonable practice] of defects actually there
- at least 50% of the bug problem is due to bad specifications given to programmers <- Bell Labs, TRW (65%)

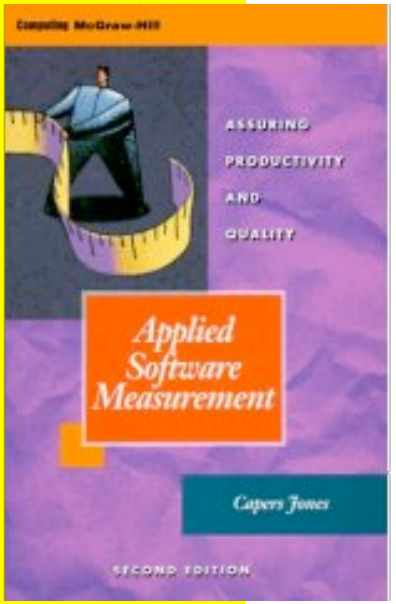
Defect *Removal* Effectiveness Inspections and Tests

24

Capers Jones,
http://www.spr.com/

Table 9: Software Defect Removal Effectiveness Ranges (Capers Jones)

Defect Removal Activity	Ranges of Defect Removal Effectiveness
Informal design reviews	25% to 40%
Formal design inspections	**45% to 65%**
Informal code reviews	20% to 35%
Formal code inspections	**45% to 70%**
Unit test	**15% to 50%**
New function test	**20% to 35%**
Regression test	**15% to 30%**
Integration test	**25% to 40%**
Performance test	**20% to 40%**
System test	**25% to 55%**
Acceptance test (1 client)	**25% to 35%**
Low-volume Beta test (< 10 clients)	**25% to 40%**
High-volume Beta test (> 1000 clients)	**60% to 85%**

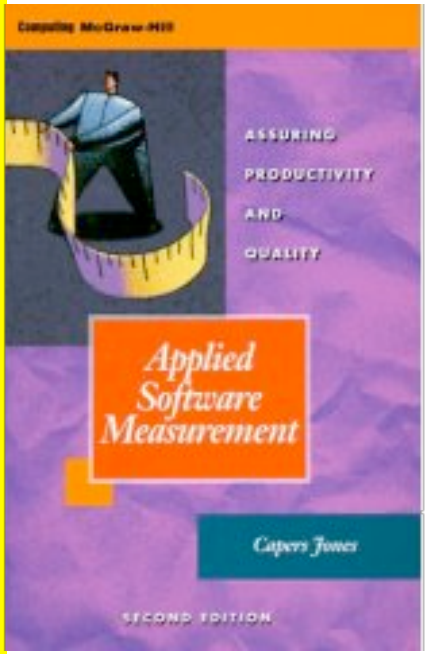


24

Capers Jones

No 'Silver Bullet' Solution Machine Guns Kill Defects

- "It is obvious that no single defect removal operation is adequate by itself.
- This explains why
  - "best in class" quality results can only be achieved from
    - synergistic combinations of
      - defect prevention,
      - reviews or
      - inspections,
      - and various kinds of test activities.
- Between eight and 10 defect removal stages are normally required to achieve removal efficiency levels > 95%".
    - Jones, Capers; Applied Software Measurement; McGraw Hill, 2nd edition 1996; ISBN 0-07-032826-9; 618 pages.



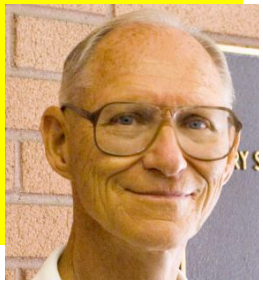
26

When do Defects occur? Upstream!

62% Design was source

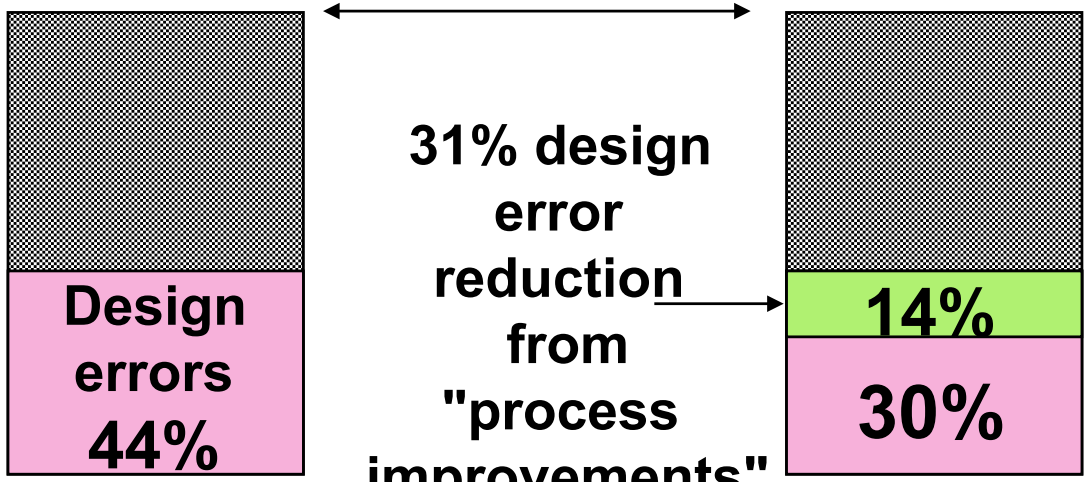
Code 38%

- **Four US Command & Control Systems studied**



Source of data is TRW Series, North-Holland Publishers, "Software Reliability" (B. Boehm)

100% of customer reported problems



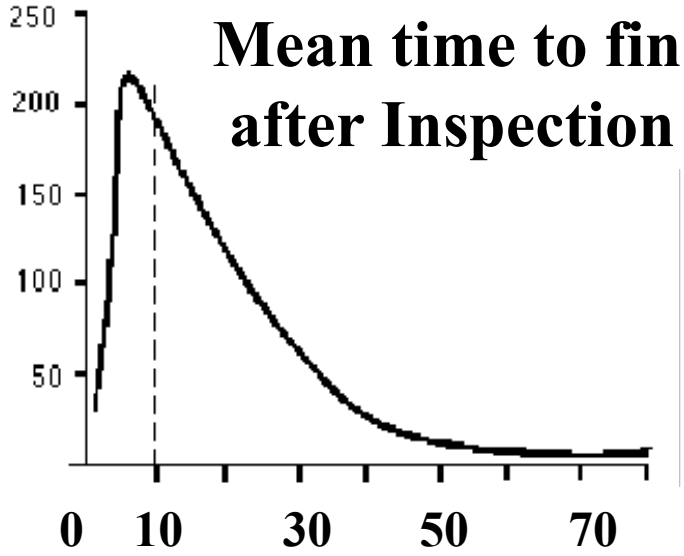
Source: J. L. Pete Pence and Samuel E. Hon III, Bellcore Piscataway NJ
Building Software Quality Into Telecommunications Network Systems, Quality Progress, Oct. 1993 pp. 95-97
"a recent defect analysis on switching system software, a supplier determined (this data).

- All development and maintenance processes
  - – can apply defect prevention technology
  - – all *should* do it too!
- The *earliest* processes need to be invested in *first*
  - – Start upstream (like requirements, contracts)
- Attacking **code** is too late
  - – The damage is already done upstream!
- The answer also depends heavily on
  - – whether you are using Evolutionary project management
    - Then you can build defect prevention into each evo cycle
    - And you can
      - – improve both people and processes,
      - – and prove measurably that you have really succeeded,
      - – early and continuously, in the field
  - – or Waterfall project management
    - In which case you will probably have to learn in one project
    - And apply learnings in future projects (not smart!)

The downstream alternative cost of quality at a Defence Electronics Factory (all types of documents for electronics).

28

Number of defects of the 1,000 sampled Majors



Mean time to find and correct a Major after Inspection was 9.3 Hours.

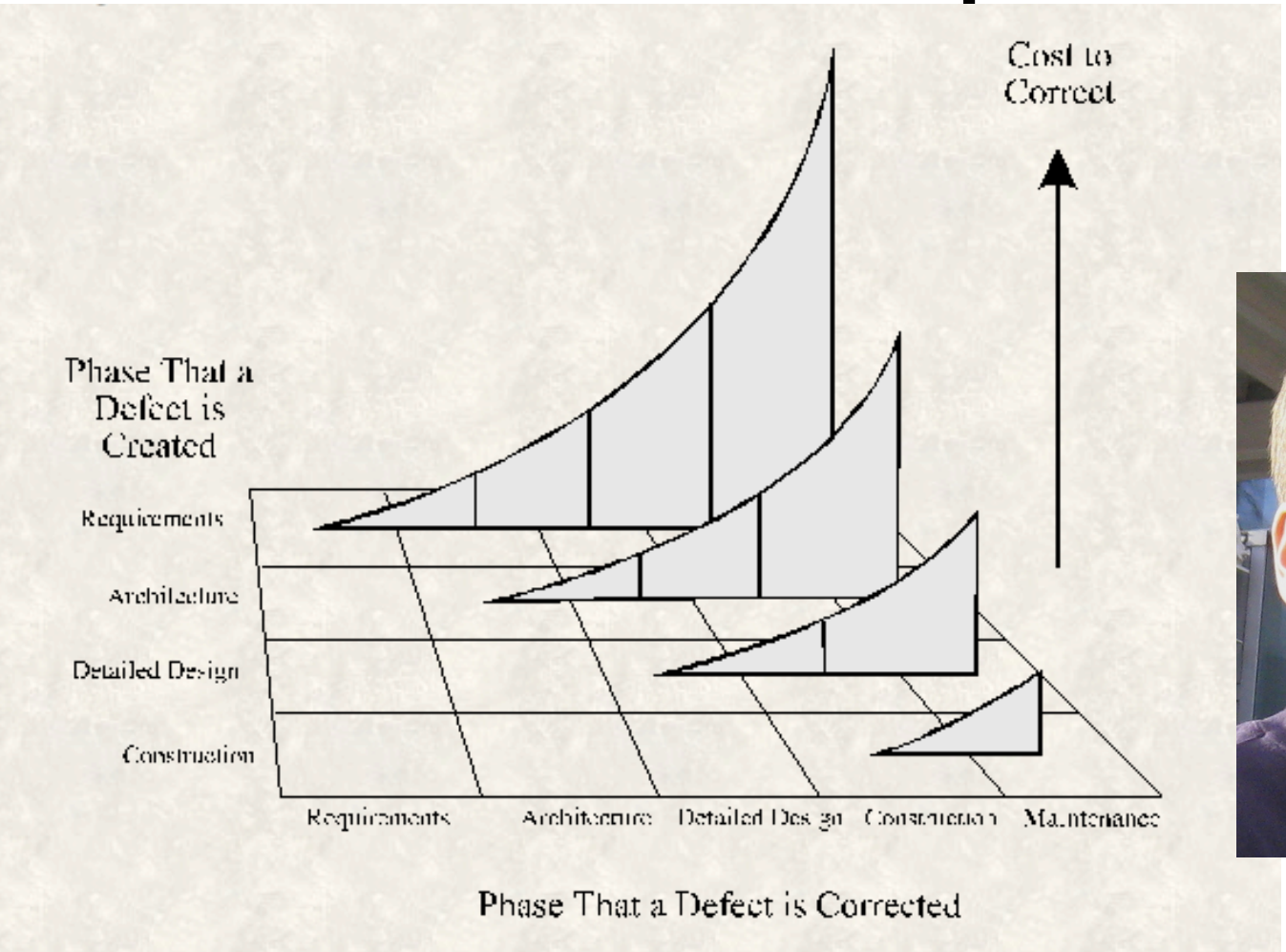
It cost about 1 hour to find and fix a Major using Inspection

Estimated hours to find and correct in test or in field

Source: Trevor Reeve, Case Study Chapter in "Software Inspection", Gilb client Philips MEL became "Thorn EMI", then Racal. Crawley UK. 1999 Raytheon?

Correction Costs Explode

29



Increase in defect cost as time between defect creation and defect correction increases. Effective projects practice "phase containment"Ldetecting and correcting defects in the same phase theyIre created.

http://www.stevemcconnell.com/articles/art08.htm

30

Generic Cost Explosion

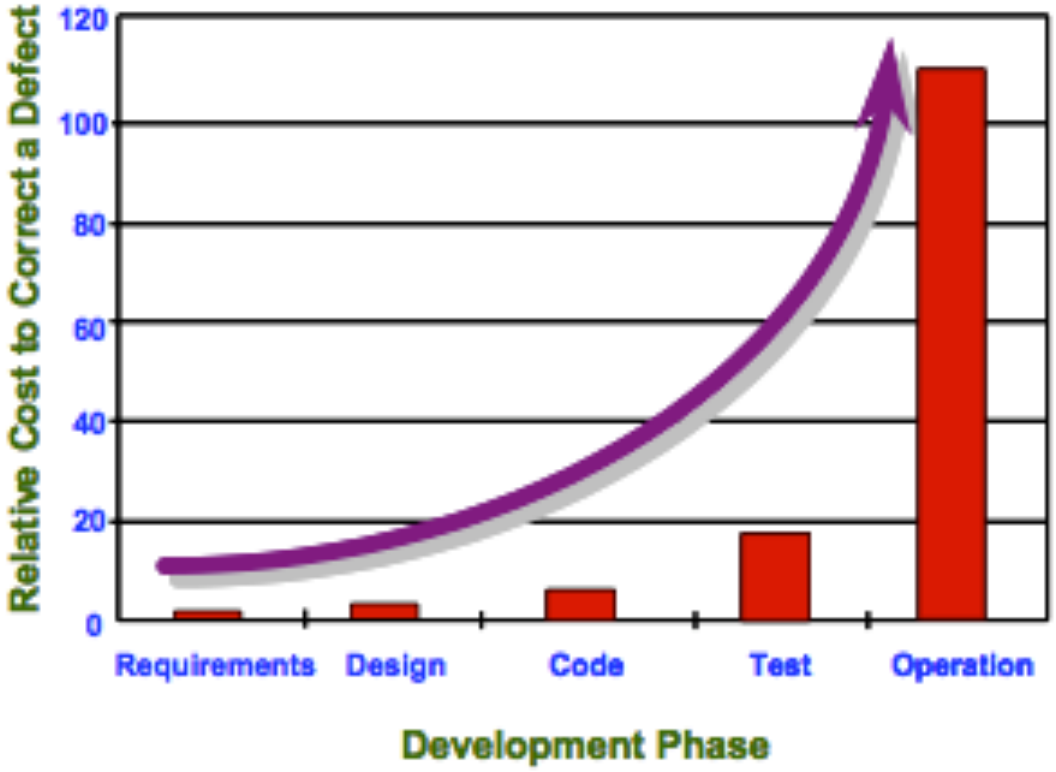


Figure 1 - Increasing cost of defect correction

- **It pays heavily to attack defects upstream**
- **(see references in ppt note)**

Errors Inserted vs Errors Found

31

Table 5-3. Example of the Frequency (%) of Where Errors Are Found, in Relationship to Where They Were Introduced

Where Errors are Introduced (%)	Where Errors Are Found					
	Requirements Gathering and Analysis/ Architectural Design	Coding/ Unit Test	Integration and Component/ RAISE System Test	Early Customer Feedback/ Beta Test Programs	Post-product Release	Total
Requirements Gathering and Analysis/Architectural Design	3.5	10.5	35	6	15	70
Coding/Unit Test		6	9	2	3	20
Integration and Component/RAISE System Test			6.5	1	2.5	10
Total	3.5	16.5	50.5	9	20.5	100%

RAISE: Reliability, Availability, Install Serviceability, and Ease of Use

The Economic Impacts of Inadequate Infrastructure for Software Testing **Final Report** May 2002

Note: this report has large scale FINANCIAL SERVICES DATA!

National Institute of Standards and Technology

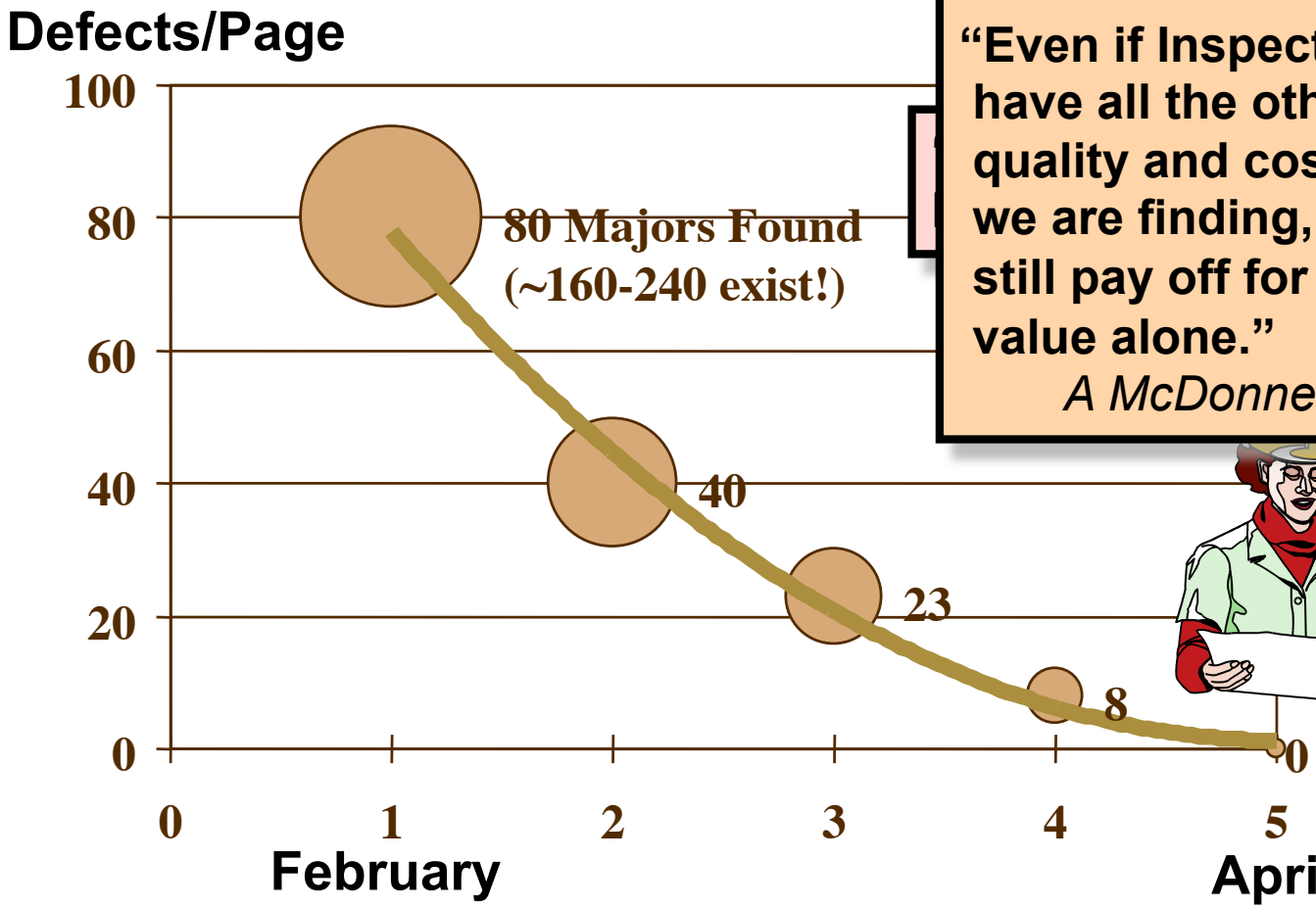
http://www.nist.gov/director/prog-ofc/report02-3.pdf

M5. The different classes of review and inspection of any type of document.

- Main inspection purposes
  - Traditional:
    - 4 people read all pages,
      - far too quickly,
      - searching for all types of defects.
    - Found defects are corrected and inspection is finished
      - (leaving most (yet unfound) defects in place, unfixed)
      - To be discovered in test of field use
  - Agile Inspection:
    - 2 people inspect a random representative sample (a page or 3).
    - Correct total major defects (rule violations) are determined.
    - Exit level is between 1 major (high maturity) to 10 majors/page remaining
    - People are 'forced' to actually learn and practice healthy rules for specification (they reduce bad practice by 50% per learning curve)
-

Positive Motivati
Personal Improveme

> **"We find an hour of doing Inspection is worth ten hours of company classroom training."**
>
> *A McDonnell-Douglas line manager*
>
> **"Even if Inspection did not have all the other measurable quality and cost benefits which we are finding, then it would still pay off for the training value alone."**
>
> *A McDonnellDouglas Director*



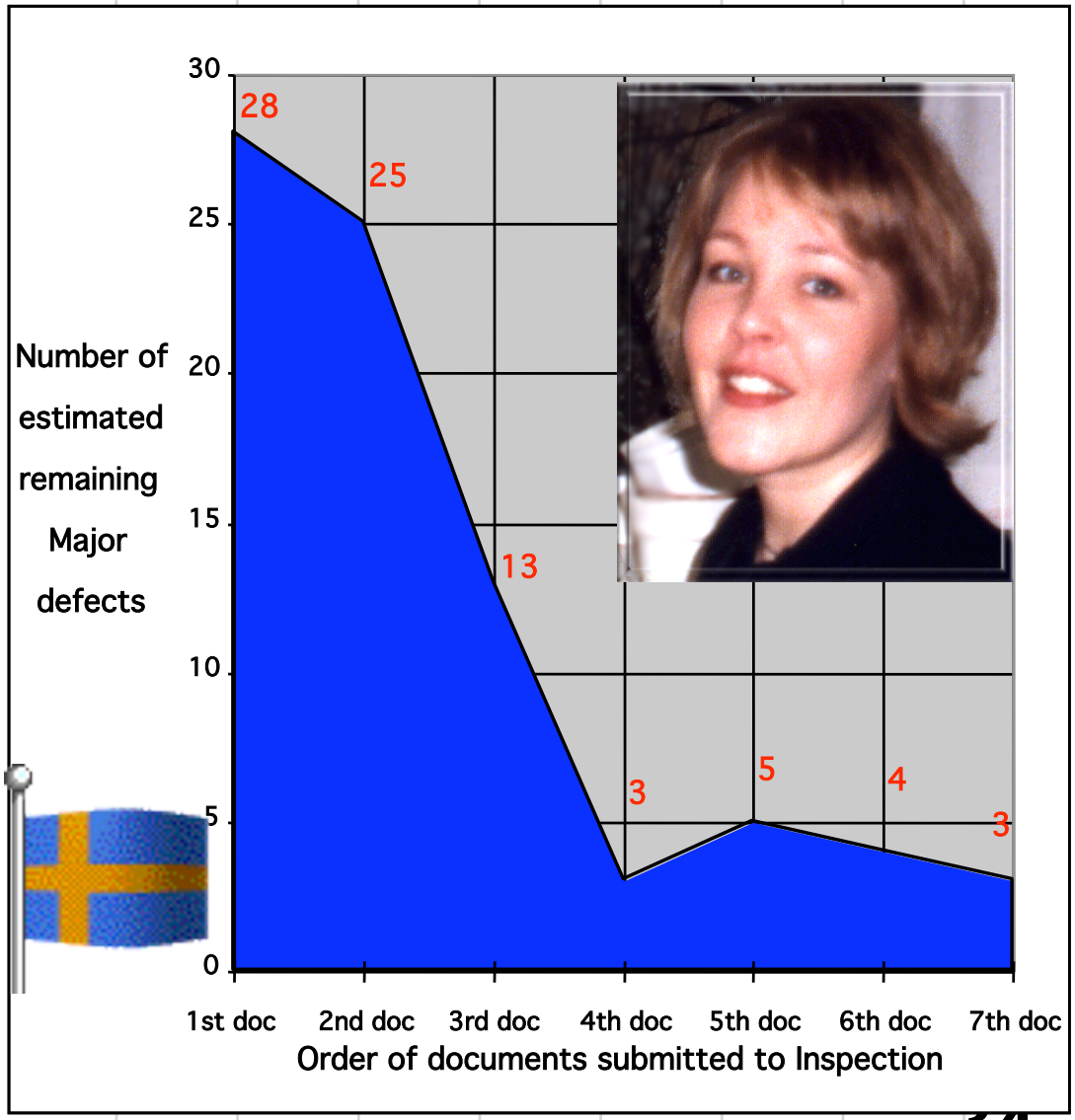
Individual learning Curve

34

- **Individual Learning Curve**
  - **The speed which the individual learns to follow the Rules,**
  - **As measured by reduced Major Defects found in Inspections**
    - **Notes:**
      - **Faster, earlier and more dramatic than "process improvement"**
      - **Never mentioned in literature as a measurable**

Marie Lambertsson's Learnability Curve,

Ericsson, Stockholm, 1997



See also the Raytheon Learning Curve

Main Inspection process types

- Cleanliness
  - –Determines if the spec/code are ‘well written’, intelligible to readership, complete, clear, unambiguous, testable

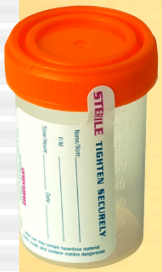
- Suitability
  - –Determines if the content of the spec is good, useful, profitable, efficient in relation to main objectives of the project, and of the spec type (e.g. architecture, code, test plans)

Cleanliness --> Suitabilty

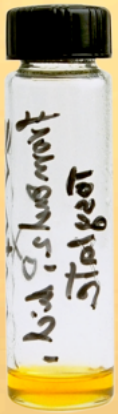
- Suitability reviews should not be conducted until we have exit of the spec for ‘Cleanliness’
- The two types of reviews should never be conducted simultaneously (illogical act)
- The ‘rules’ used to conduct the inspections will determine which type of review is taking place
  - Rule C: must be clear, complete, unambiguous
  - Rule S: must help meet our quality objectives, Cost-effectively

A Sample page Marked By Checker
2 General Rules = 153 majors/Page density

Sample 1



Sample 2



1.1 Look and feel

1.1.1 No know

- The user will not be required to have ANY operating system knowledge or Unix skills

1.1.2 MS-Windows user interface

1.1.3 MS-Windows concepts

- The system will make full use of the MS-Windows user-interface concepts such as

1.1.4 Hot keys

- The user will ... customize the ... user will be able to

2 General Tools: Visualization (1.5) m-

The section will cover:

- Geographic display (2.1)
- Data viewer (**Error! Reference source not found.**)
- 2D graphs (**Error! Reference source not found.**)
- Distribution graphs (**Error! Reference source not found.**)

2.1 Geographic Display Area (1.5) m -

- The display area will be used for display of:
  - Reference information, such as raster and vector imagery
  - Plan information, such as locations of points, lines, exclusion areas etc.
  - Real-time information, such as vibrator status, or spread status

2.1.1 Geographic Display start-up (1) m-

- Upon starting up the display the mapped area will correspond to the one displayed upon last use, or the entire project area if no previous area default is available.
- Upon starting up the display the layer settings (which layers are visible, annotation setting etc.) will be retained.

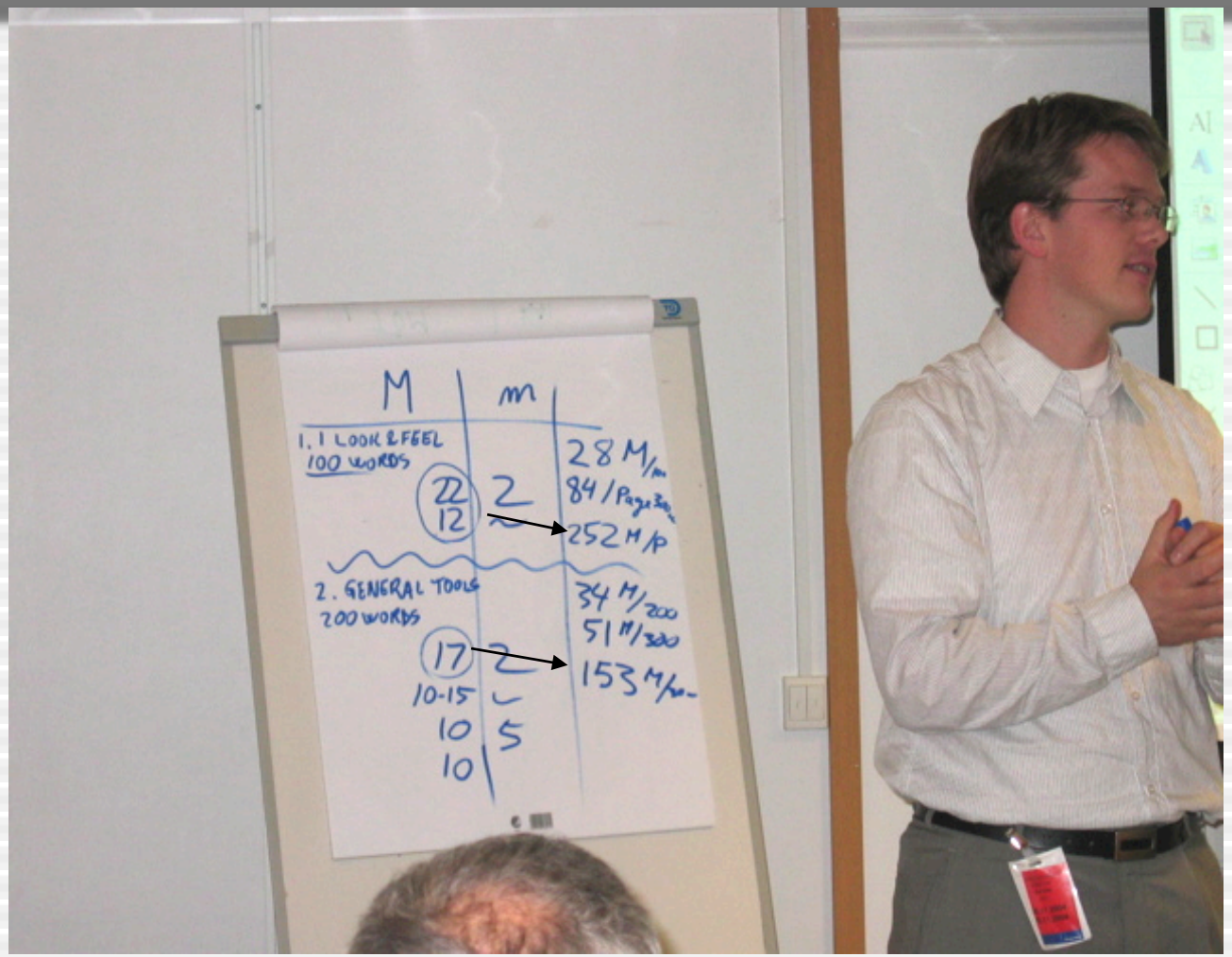
2.1.2 Layer management (1.5) m -

1.1.3 MS-Windows concepts

The system will make full use of the MS-Windows user-interface concepts such as Wizards to lead the user through user-defined

It will be quick and simple to add and remove

Sample Major Defect --> Extrapolations Done
= 153 Majors/Page and 252 Majors/Page
from Samples of Real requirements
determination done by responsible managers, 2004



Rewrite of a real Defective ‘Requirement at Gilb Client 2004

- 1.1.3 MS-Windows concepts
- The system will make full use of the MS-Windows user-interface concepts such as Wizards to lead the user through user-defined

False Requirement (a solution)

Solutions (Designs):
The system will make full use of the MS-Windows user-interface concepts.
Examples: such as Wizards to lead the user through user-defined parameters.

Analysis

Why? Lots of users ask for it. (MS-Windows)
Why? Easy to use. / Intuitive

Usability {intuitiveness, learn, training, mistakes}

Usability.Intuitive
Ambition: after initial training, (one week course, two week field) the user shall not have to refer to the user manual.

Scale: % of defined [Elements] done Correctly, by defined [User], within <5> seconds.

Correctly: defined as: the System responded in a way the user thought the system should do.

System: Defined as: xxx

Record [ISX Sierra, 1994] 95%±5% <- Boss *“as perceived b… the…*
Record [Product = 408] ??%

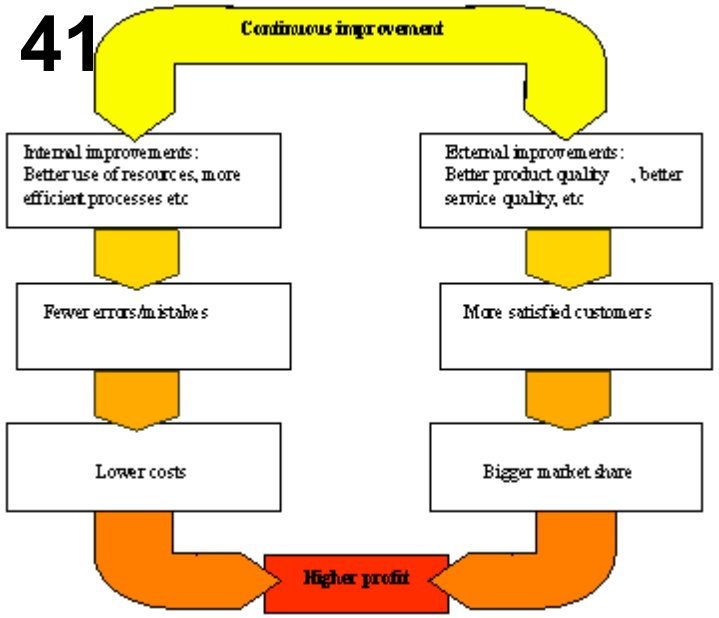
Past [Elements = Finding a menu option, User = Beg…
Tolerable
Goal [Elements = Finding a menu option, User = Beg… Ma… 1th …] 70% … <- the team
Goal [Elements = Finding a menu option, User = Beg…, March 1…th …, at …commercializa…]
90%±5 <- the team

The 'Real' Requirement in Planguage

M2. Defect Detection strategies versus Defect Prevention strategies

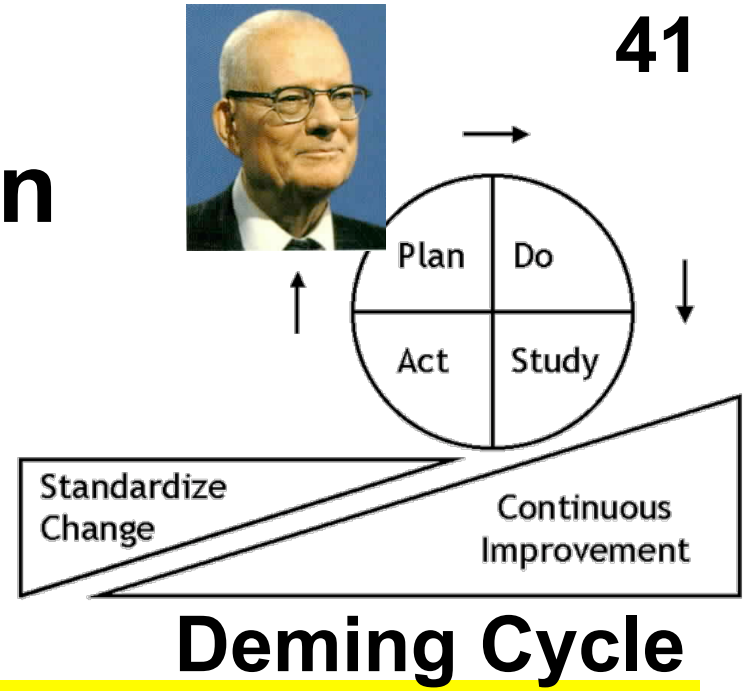
- Defect detection
  - –(inspection, test, customer reports)
  - –Is *ineffective* for getting high bug-freeness into systems
  - –It is better than nothing
  - – Inspection is cheaper than test-and-debug
- Defect Prevention - is at 2 levels
  - – process improvement
    - (CMMI Level 5)
  - – individual capability improvement
    - (50% per motivated cycle)

- Defect prevention is BY FAR the smartest one



Prevention Costs

41

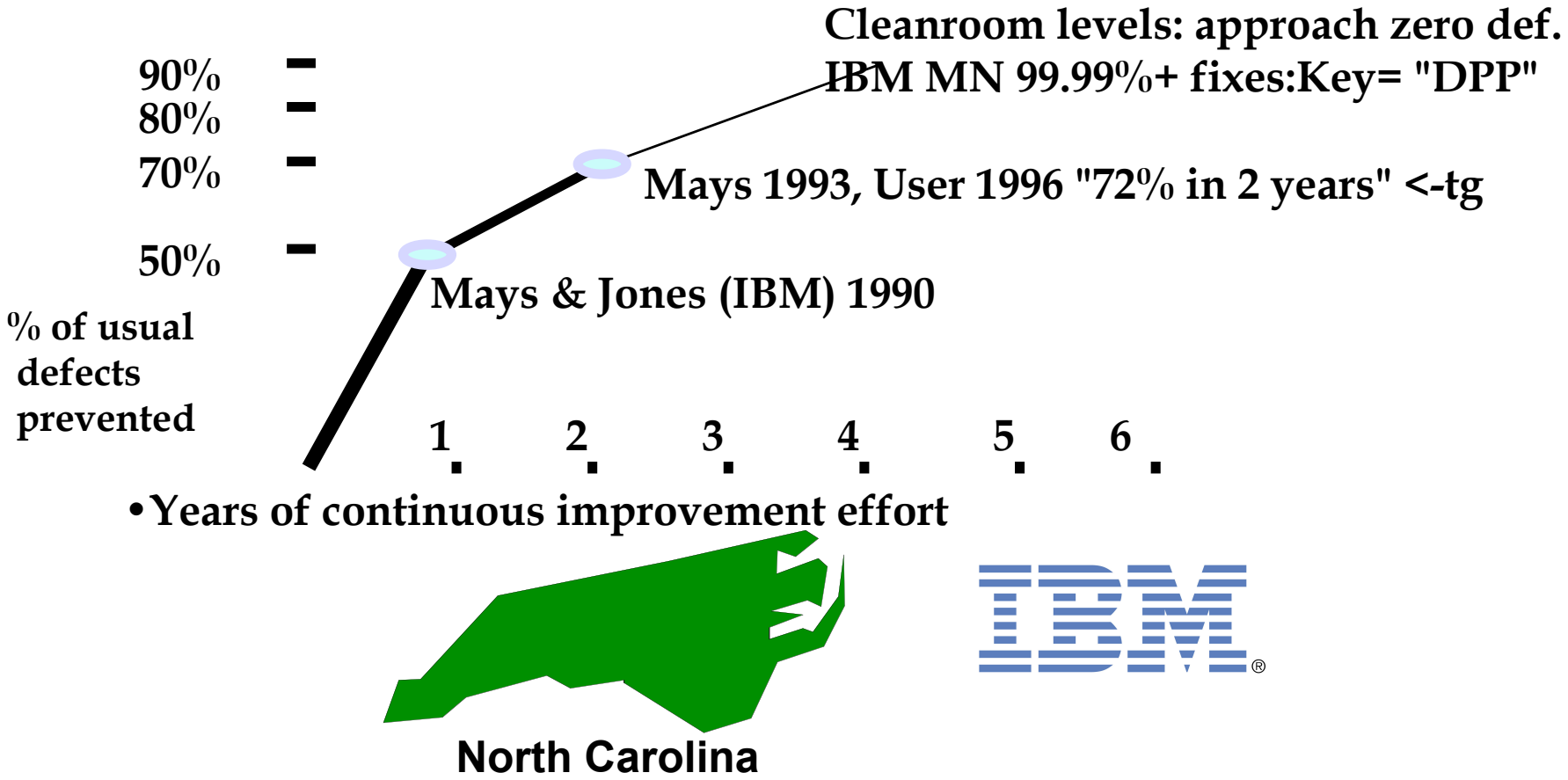


Deming Cycle

- **5.0 %, stable at 5%**
  - **of development costs**
  - **(Raytheon 1993)**
- **0.5 % of development costs**
  - **(Mays 1995)**

42

Defect Prevention Experiences: Most defects can be prevented from getting in there *at all*

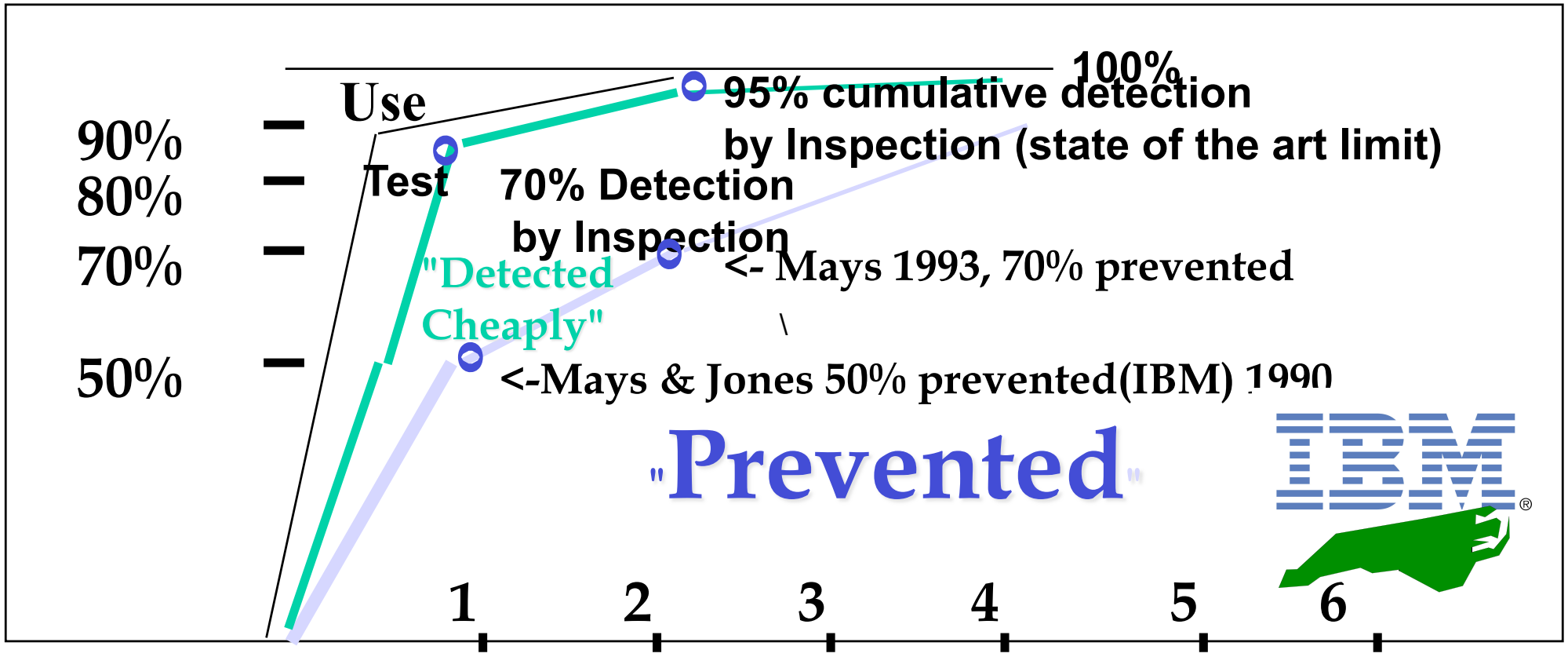


IBM Research Triangle Park Networking Laboratory

42

Prevention + Pre-test Detection is the most effective and efficient

43

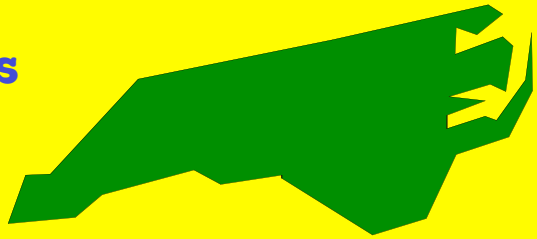


- **Prevention data based on state of the art prevention experiences (IBM RTP), Others (Space Shuttle IBM SJ 1-95) 95%+ (99.99% in Fixes)**
- **Cumulative Inspection detection data based on state of the art Inspection (in an environment where prevention is also being used, IBM MN, Sema UK, IBM UK)**

IBM MN & NC DP Experience

44

- **2162 DPP Actions implemented**
  - **between Dec. 91 and May 1993 (30 months)<-Kan**
- **RTP about 182 per year for 200 people.<-Mays 1995**
  - **1822 suggested ten years (85-94)**
  - **175 test related**
- **RTP 227 person org<- Mays slides**
  - **130 actions (@ 0.5 work-years**
  - **34 causal analysis meetings @ 0.2 work-years**
  - **19 action team meetings @ 0.1work-years**
  - **Kickoff meeting @ 0.1 work-years**
  - **TOTAL costs 1% of org. resources**
- **ROI DPP 10:1 to 13:1, internal 2:1 to 3:1**
- **Defect Rates at all stages 50% lower with DPP**



M4. The role of 'requirements' in defects.

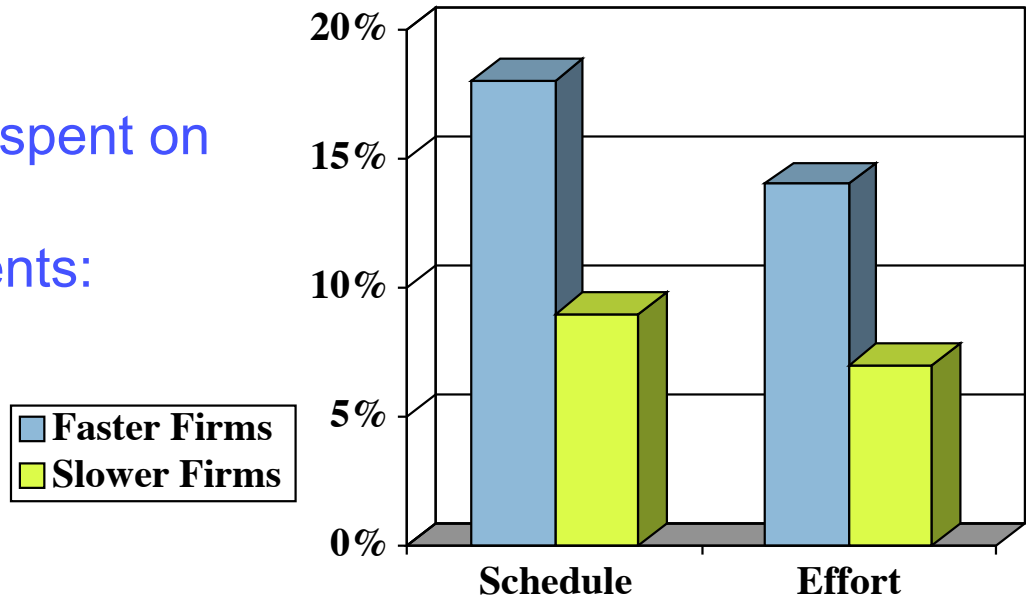
- bad requirements is known as a major cause of defects in code. (Jones, NASA, slides follow)
- most IT shops have extremely high major defects in requirements
  - 100 majors/page ± 80
  - they don't themselves measure, know, or plan to reduce
- Major defects in requirements are directly causal (33% probability) of bugs
  - see GE example slides follow
- To fight this you need to
  - establish rules/standards for requirements
  - train people in following the rules
  - use QC reviews and exit levels to *motivate* people to learn and follow the rules

Why Requirements?

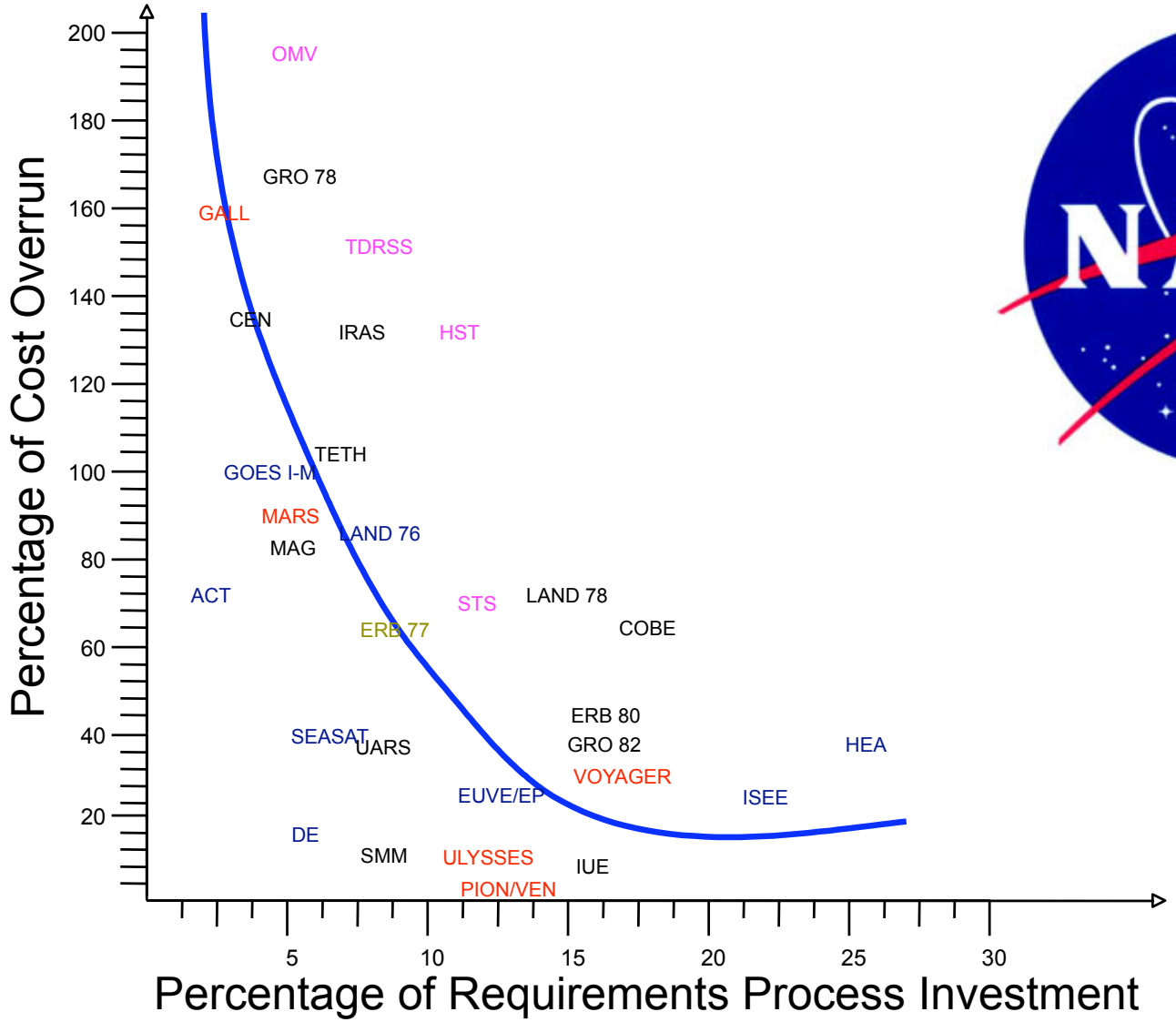
"The firms improving their development speed at the fastest rate actually spend *more* elapsed time and more effort in the customer requirements stage of the project …. These firms spend significantly less time in the testing and integration stage of development.…"

Blackburn, Scudder, et al, in *Improving Speed and Productivity of Software Development: A Global Survey of Software Developers* (1996)

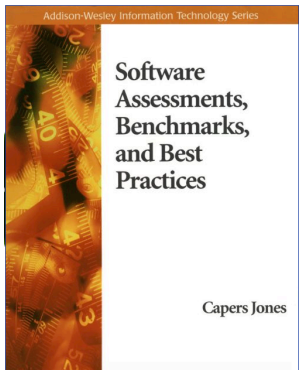
Resources spent on Customer Requirements:



Effect of Investment in the Requirements Process on % program Cost Overrun



"Successful projects do planning very well indeed"



Delayed or cancelled projects, however, almost always have planning failures.

The most common planning failures include

(1) not dealing effectively with **changing requirements**;

(2) not anticipating staff hiring and turnover during the project;

(3) not allotting time for **detailed requirements analysis**; and

(4) not allotting sufficient time for inspections, testing, and defect repairs.

Capers Jones

http://www.stsc.hill.af.mil/crosstalk/2004/10/0410Jones.htmlOct/\2004/\Issue, **Software Project Management Practices: Failure Versus Success, Capers Jones, Software Productivity Research LLC**

- *Unclear*
  - *'very user friendly'*
- Clear: (but inappropriate)
  - 'can be learned by user in 8 weeks'
- **Appropriate/Suitable – The Right Stuff** (*and* clear)
  - 'can be learned by the user in less than 5 minutes'

- An ideal organization
  - Sets clear measurable critical management objectives
  - works towards those long term goals in an evolutionary way
    - early results, continuous results, testing each selected strategy for effect and costs
  - judges strategies (fro reaching goals) entirely on estimated, then REAL tested effect on their goals
  - Motivates managers based on real results
    - not ‘inspections in place’
    - But ‘ 1,000 reduction in field bugs’

M8. Suggested Policy as foundation for the ideal organization

P1: The foundation of our management work is a set of *agreed, official, quantified long term critical* objectives

- P2: potential strategies will be selected based on real prior experience
  - –somewhere that they are capable of meeting our goals on time

P3. Strategies will be implemented evolutionarily

- – (early, measured, continuously, learning and changing, tuned to work, or discarded if not)

M9. Suggested quantified objectives for the organizational improvement.

- Development Product Quality:
  - Ambition: all critical defined qualities Goal levels are reached or exceeded on time
- Development Productivity:
  - Ambition: the available staff can meet current Requirements on time.
- Timeliness:
  - Ambition: the critical deliveries are made early or on time, never late.
- Value Delivery:
  - Ambition: the highest priority planned value items are delivered first before effort is expended on lower value items.

Development Product Quality:

Ambition: all critical defined qualities Goal levels are reached or exceeded on time

Scale:

the % of defined [Critical Quality Goals] that are provably met or exceeded on initially planned time schedule.

Meter: <project accounting tracking of quality objectives reached, dates and specified Goal levels>

Past [2006, Critical Quality Goals = Reliability] 50±50%??

Goal [2007, Critical Quality Goals = Reliability] ~ 100%

Development Productivity:

Ambition: the available staff can meet current Requirements on time.

Scale:

- % of defined [Priority] Requirements that are in fact testably delivered as originally Committed.

Meter: <project tracking database data on requirements, deadlines and actual tested confirmation>.

Past [2006] 50%±45% ?? <- placeholder guess.

Goal [2007, Priority = Highest] 90% ? <- suggestion

Stretch [2007, Priority = Highest] <99% ?

Priority: defined as: official company priority category when requirement is stated and approved. {Highest, High, Medium, Low}.

Committed: defined as: officially approved by Requirements Board, and officially agreed by development.

Requirements: defined as: in Planguage, all classes of requirements, function, performance, constraints etc.

Requirement Timeliness:

Ambition: the critical requirement deliveries are made early or on time, never late.

Scale:

- % of defined [Criticality] Requirements that are not late in being delivered, and testably available at Goal or complete level, for relevant and defined stakeholders.

Meter: <project tracking data analysis?>

Past [2006, Criticality = All] 50%±49% ??

Goal [2007, Criticality = {High, Highest}] 98% ? <- tg

Criticality: defined as: The level of criticality indicated by the Requirement specification. {Highest, High, Medium, Low, All}.

Value Delivery:

Ambition: the highest-priority planned value items are delivered first, before effort is expended on lower-value items.

Scale:
% of defined [Priority] items delivered within their Deadline.

Meter: <project statistics, or sampling of project data?>

Past [2006] 20%?? <- wild intuitive guess, no data yet.

Goal [2007, Priority {Highest, High}] 90%±10%? <- initial try at level.

Priority: defined as: official company priority category when requirement is stated and approved. {Highest, High, Medium, Low}.

M10. Suggested main strategies to achieve these objectives

- The management Process
  - Adopt the Policy
  - Decide on your quantified objectives
  - then find appropriate strategies
  - Test best value strategy first
  - Then 'evolve' towards meeting your goals.
- Probable technical strategies
  - Improve requirements specification standards
  - Use Spec QC to measure the improvement, and also to *motivate* the improvement
  - Use Spec QC to measure all specs/code
  - Focus efforts upstream first (not on code first)

M11. The notion of evolutionary measurable change to meet objectives.

- The following key ideas apply to this method:
  - quantified objectives as primary control
  - Early and continuous measurement of progress (weekly, monthly)
  - One strategy at a time
  - One objective at a time
  - One project at a time
  - highest value strategies first

Potential Additional Topics if there were time or priority:

-
- Requirements
- Design
- Prioritization methods
- Risk control
- Evolutionary project management
- Agile methods

M12: analysis of your current Inspection, testing, and operational defect data. What does it tell us?

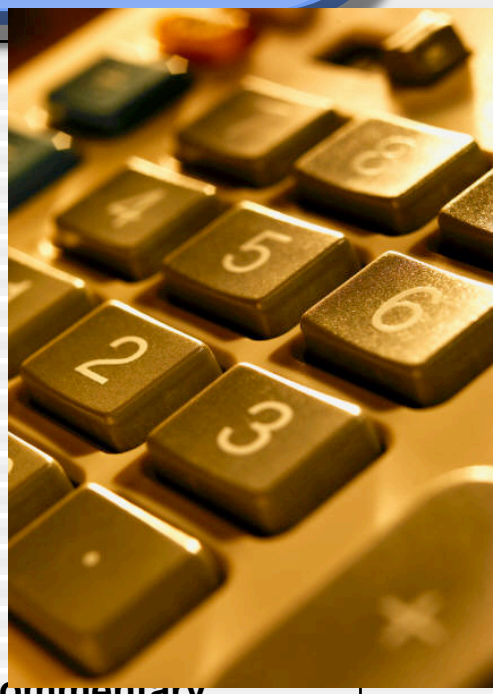
- we need access to more of this this data.
- We need to discuss it with you
- This could be shared Monday during the day

M15: What would an ideal data collection and analysis programme look like?

- Here is the Agile SQC suggestion
  - (next slides)

Agile Inspection/ Extreme SQC Form

Inspection/SQC Blank Form: Version May 29, 2003
Date Started:
Leader:
Author:
Other Checkers:
Specification Reference: **Total Physical pages:**
Spec Sample Reference:
Rules Checked:
Sample Size: **(Non commentary words)**
Checking Time Planned: **Actual:**
Checking Rate Planned: **Actual:**
Defects Identified:
Majors:
Minors:
Estimated Unique Majors Found by Team: **±**
Estimated Average Majors/Logical Page: **(Logical Page = 300 Non Commentary Words)**
Estimated Total Majors in Specification:
Majors in Relation to Exit level:
Recommendation:
Causes (of defect level):
Actions suggested to mitigate Causes:
Responsible for Action:
Completion Date/Time:



Form with Hints

Date Started: <date and time you started writing this plan>
Leader: <planners name , YOURS!>
Author: <the person who wrote or updated the spec, and is on the QC team>
Other Checkers: <others joining the QC team>
Specification Reference: <identify the spec, include version date, scope> Total Physical pages: <of the spec>
Spec Sample Reference: <scope the sample selected, usually a representative page or 2>
Rules Checked: <refer to names of all specification rules you intend to check against now>
Sample Size: <approximate number of non commentary words in sample> (Non commentary words)
Checking Time Planned: <minutes for the checking> Actual:
Checking Rate Planned: <Logical Pages/hour, usually 2 to 1> Actual:
Defects Identified:
Majors: <the set found by checkers on the team eg 2, 3, 6>
Minors: <the set found by checkers on the team eg 12, 8, 16>

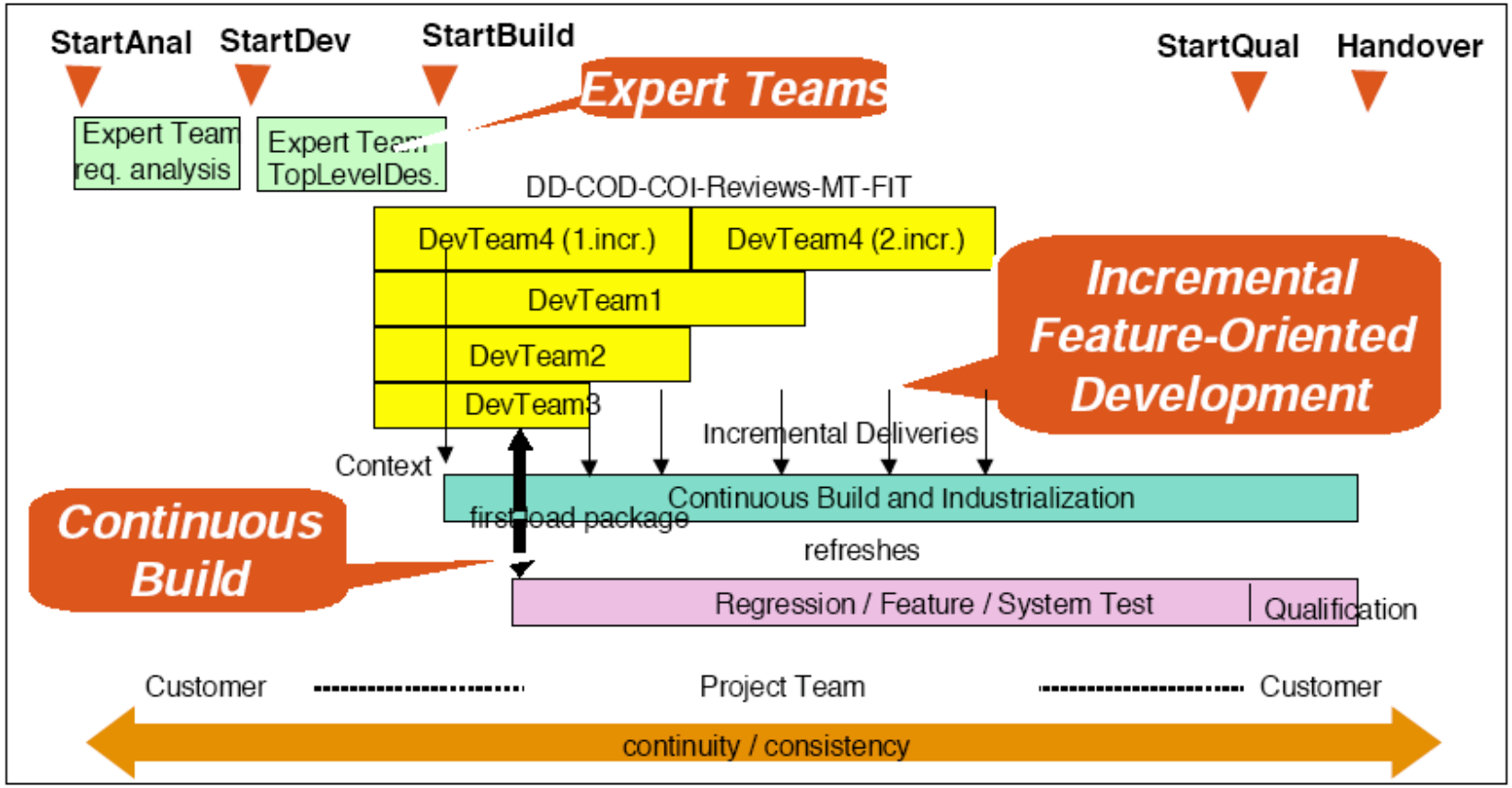
Estimated Unique Majors Found by Team: <usually 2x most Majors found by best checker> ±
Estimated Average Majors/Logical Page: <usually 3x Majors found by team above> (Logical Page = 300 NC Words)
Estimated Total Majors in Specification: <Average/L Page x physical pages>
Majors in Relation to Exit level: Density/L Page in relation to Exit level, assume 1 major max>
Recommendation: < exit, rework etc.>
Causes (of defect level): <why were there so many Majors, underlying cause, forcing author to commit them>
Actions suggested to mitigate Causes: <individual level and organizational level>
Responsible for Action: <who is going to improve the organizational process?>
Completion Date/Time: <date and time this was completed>

Form Filled Out Example

Date Started: May 29 2003
Leader: Tom
Author: Tino
Other Checkers: Artur
Specification Reference: Test Plan V 2.0 Total Physical pages: 10
Spec Sample Reference: page 3
Rules Checked: Generic Rules, Test Plan Rules
Sample Size: ~300 (Non commentary words)
Checking Time Planned: 30 minutes Actual: 25 minutes
Checking Rate Planned: 2 pages/hour Actual:
Defects Identified:
Majors: 6, 8, 3
Minors: 10, 15, 30
Estimated Unique Majors Found by Team: about 16
Estimated Average Majors/Logical Page: ~16 x 3 = 48 (Logical Page = 300 Non Commentary Words)
Estimated Total Majors in Specification: 48 x 10 = 480
Majors in Relation to Exit level: 48/1 (47 too many)
Recommendation: no exit, redo and resubmit
Causes (of defect level): author not familiar with rules
Actions suggested to mitigate Causes: author studies rules, all authors given training in rules
Responsible for Action: project manager
Completion Date/Time: May 29 2003 18:08

Extra Slides

Alcatel Evo : Ebert



Cycle Time Reduction by Early Defect Detection: Alcatel

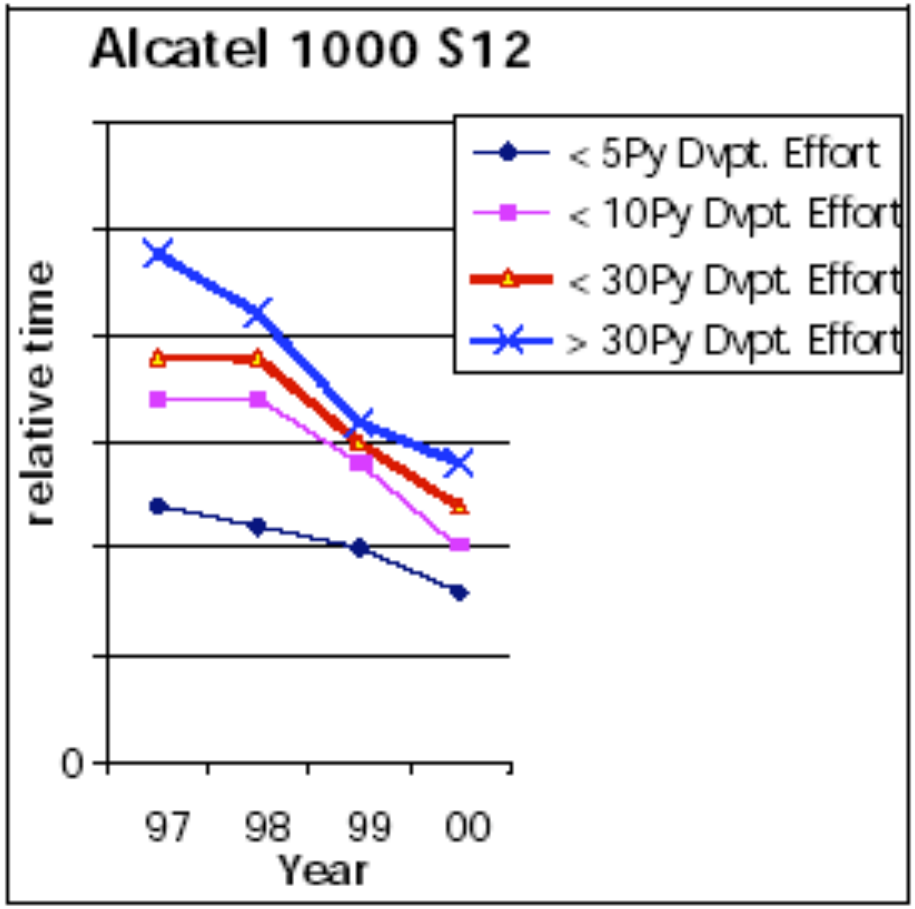


Fig. 8: Cycle Time Reduction is a Desired Side Effect of Earlier Defect Detection and Continuous Build

Cost of Non-Quality: Alcatel

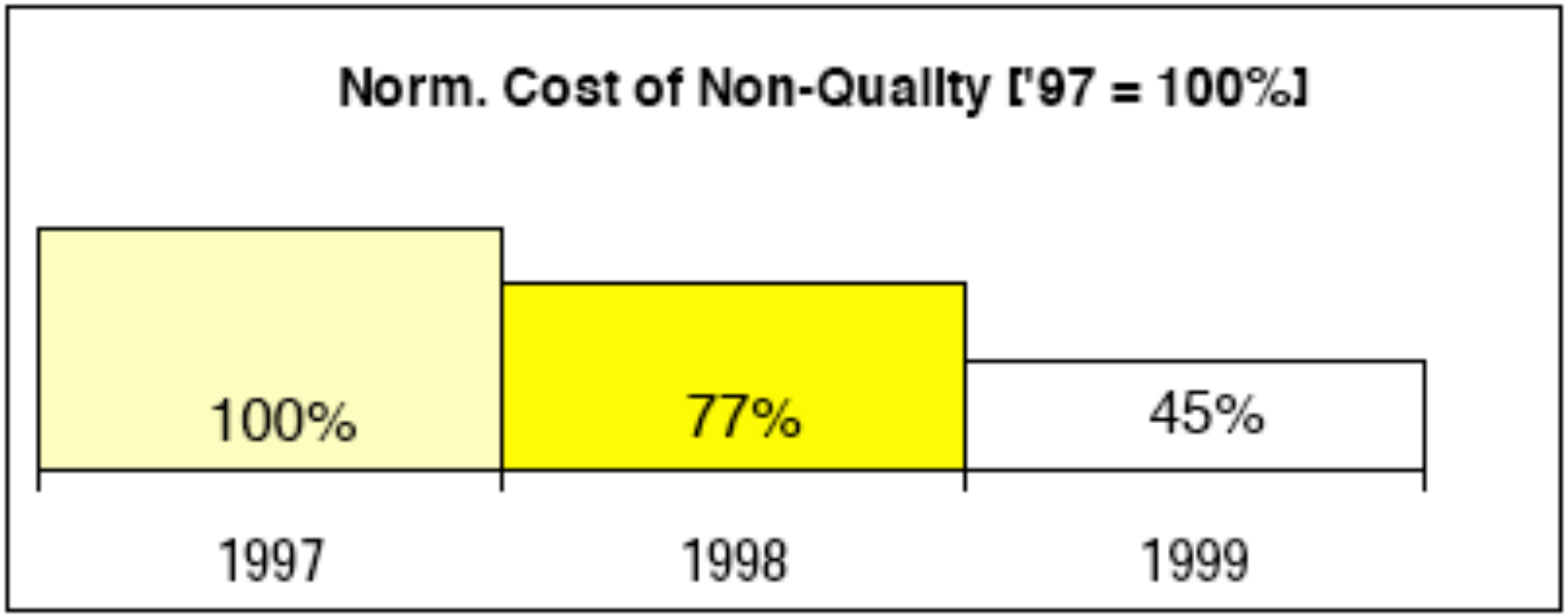
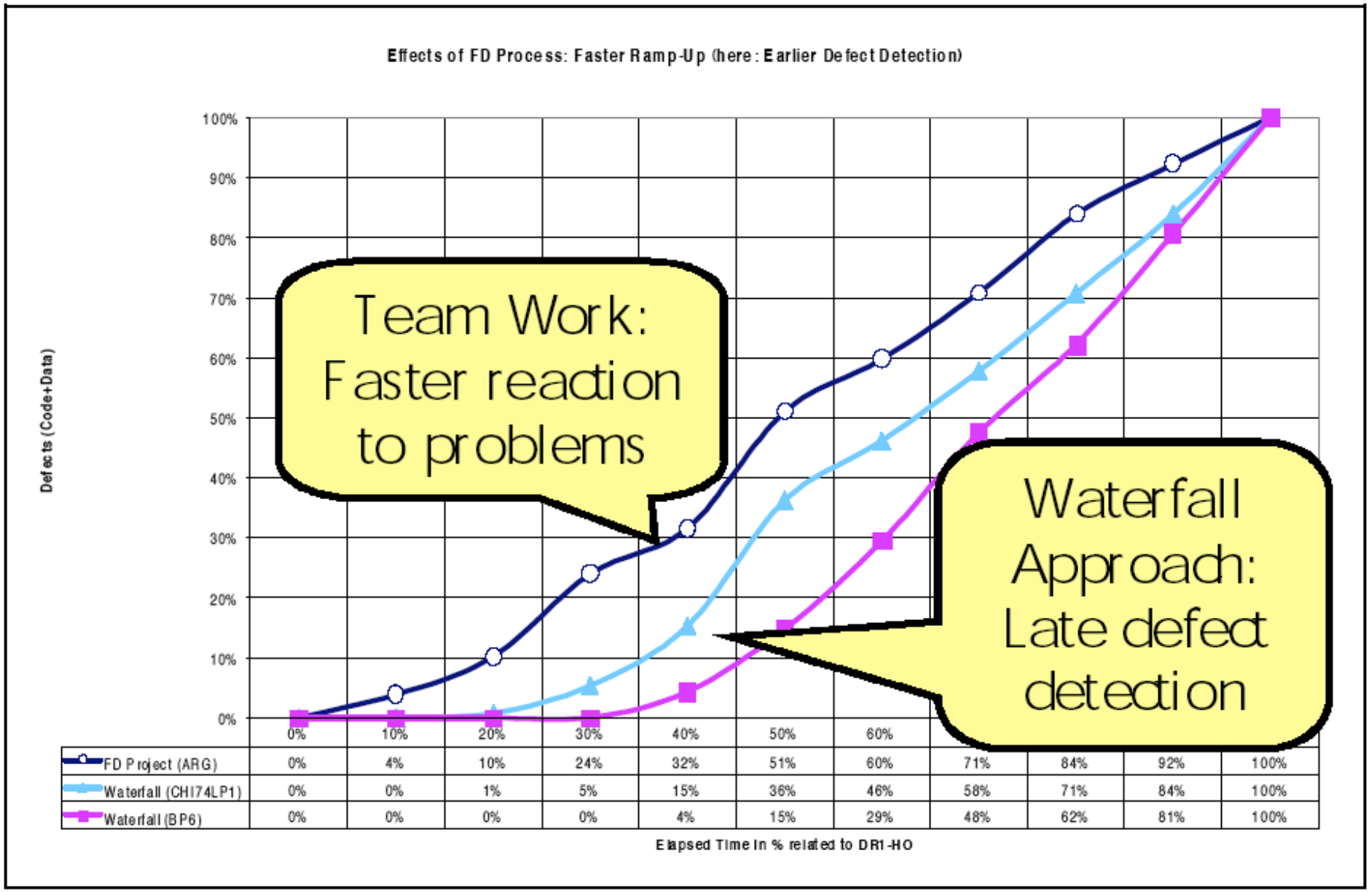


Fig. 7: Reduction of Normalized Cost of Non-Quality over past 3 Years by Combined Focus on Collocating Peer Reviews, Effective Process Coaching, and Introduction of Teamwork and Continuous Build

Reaction Time by Faster Reaction to problems



	0%	10%	20%	30%	40%	50%	60%				
FD Project (ARG)	0%	4%	10%	24%	32%	51%	60%	71%	84%	92%	100%
Waterfall (CHI74LP1)	0%	0%	1%	5%	15%	36%	46%	58%	71%	84%	100%
Waterfall (BP6)	0%	0%	0%	0%	4%	15%	29%	48%	62%	81%	100%

Reaction Time by Faster Reaction to problems

We evaluated the effects of this reengineered process carefully over the past 2 years. Consequently, we see two effects contributing to the hypothesis. Response time and thus overall cycle time is reduced as defect correction happens in the team (fig. 5). Field defects are reduced due to focus on an optimized test strategy, longer overall test-time and end-to-end responsibility of a development team.

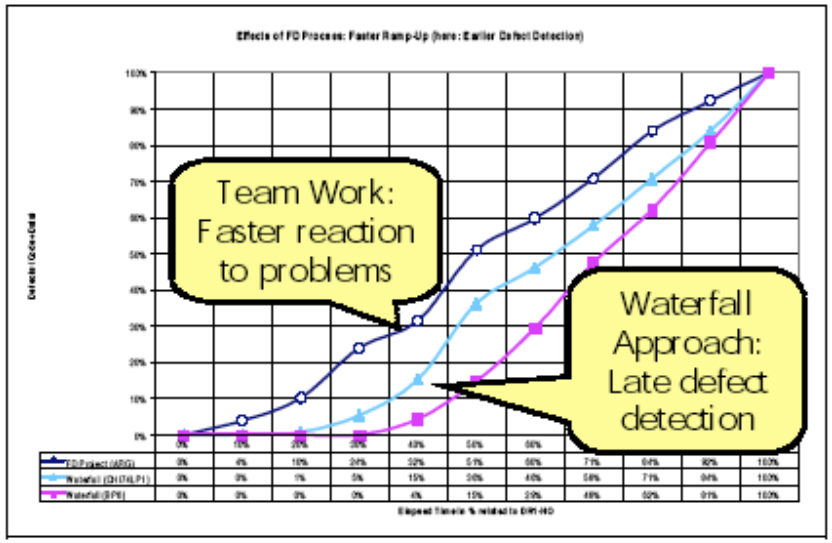


Fig. 5: Effective Team Management Scales Directly up to Faster Reaction Time

Last slide. Comments to tom@gilb.com

- **JUSE Tokyo September 2008**