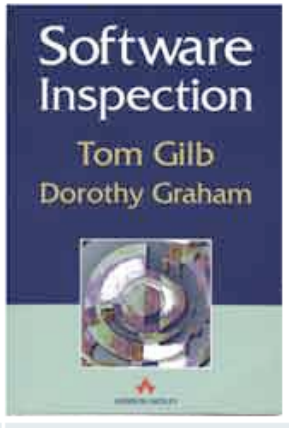


Keynote 2:

45-60 MINUTES 21 MAY 2009, London, Unicom

How to Evaluate if Requirements Specifications are Good Enough for Testing: The Agile Inspection Measurement Process and Numeric Exit

Tom Gilb

(TOM@GILB.COM, WWW.GILB.COM)

Basic Standards for Testable Requirements !

- The Testers Dilemma: Garbage In
  - But we cannot allow it to produce Garbage Tests
- The notion of standards for requirements

The Test Planning Dilemma

- **Delivery to specified requirements must be tested**
- **Interpretation of the requirement must be correct (as intended, and same as developer)**
- **Most people write requirements with 50-250 unclear words per 300 words of text!**
- **There are no standards applied most places**
- **There is no 'enforcement' of standards most places**
- **Result: testers inevitably get extremely bad requirements to develop tests from**
  - **So do developers!**
- **Who is responsible?**
  - **IT Management –**
  - **But, Test Management must make sure they take their responsibility**
  - **Maybe in an alliance with development management: common interest?**

Abstract

Ten principles for testable requirements

- Some basic requirements specification standards to make requirements testable
- numeric exit and entry levels as a quality level
- two levels of quality: clear and relevant
- defined ‘rules’ to teach and measure requirements
- well defined concepts – like ‘requirement’
- templates to guide requirements specs

Ten Principles for Testable Requirements:

- **Based on use of 'Planguage'; a requirements specification language.**

Kai Gilb

http://homepage.mac.com/tomgilb/filechute/%20%20Gilb%20Competitive%20Engineering%20Book%20copy.pdf

http://www.gilb.com/tiki-download_file.php?fileId=27

Principle 1.

- **The requirements must themselves be**
  - **clear,**
  - **complete,**
  - **and consistent**

Basic 'Rules' for Requirements

- 1. **Unambiguous** to intended Readership
- 2. **Clear** enough to ***test***.
- 3. **No** unintentional **Design**
- **4. Consistent, with itself and all related documentation.**

These are "Clarity" Rules: later, 'relevance' rules need application

Requirement Defect Rates Improvement in 6 months in Financial Business, London, Gilb Client Using Spec Quality Control /Extreme Inspection + Planguage Requirements

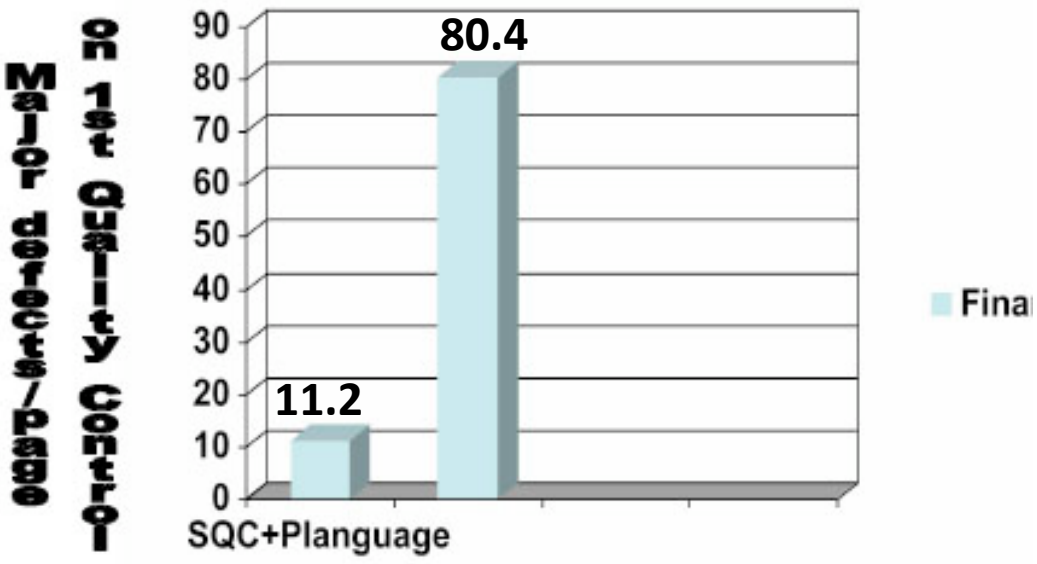
Across 18 DV (DeVelopment) Projects using the new requirements method, the average major defect rate on first inspection is 11.2 per logical page.

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:

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

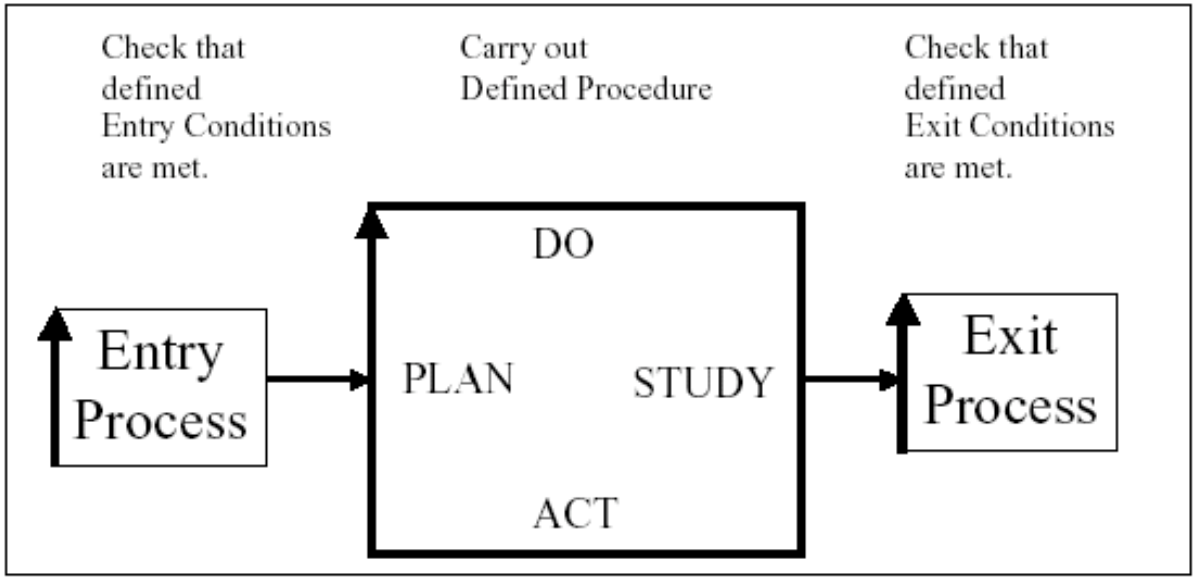
The average major defect rate in this sample was 80.4 per logical page.



Principle 2.

- **The requirements must follow our current standards, *measurably***

You should have NUMERIC exit and entry quality levels from both test processes and related development processes



- **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)**
- **No 'Garbage In' to Test Planning!**

The downstream alternative cost of quality at a UK Defence Electronics Factory. 9 to 1 more

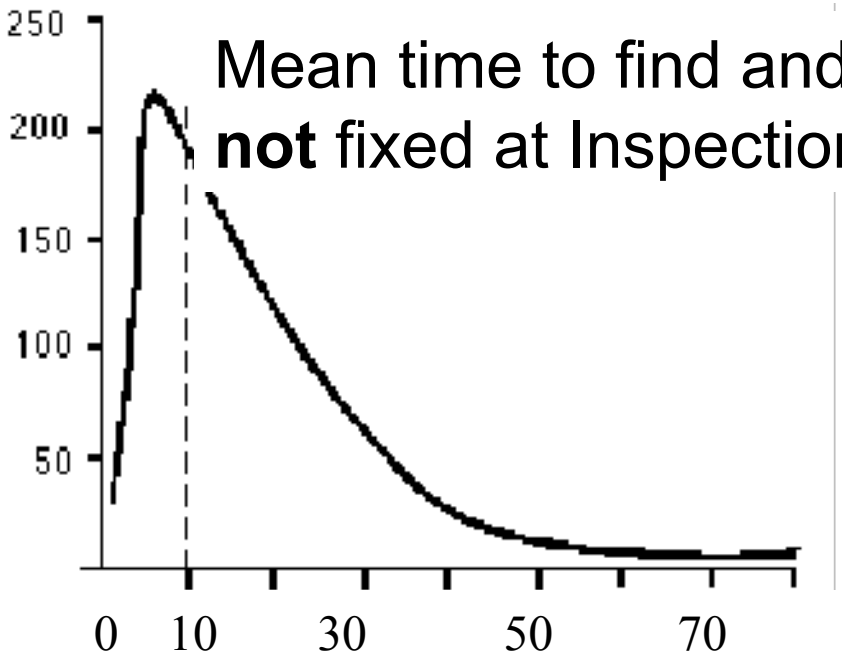
(all types of documents for electronics).

Number of defects of the 1,000 sampled Majors ------>

That we

manually estimated

downstream costs to fix

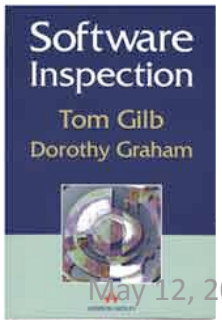


Mean time to find and correct a Major if **not** fixed at Inspection was **9.3** Hours.

It cost about **1** hour to find and fix a Major **at** time of Inspection

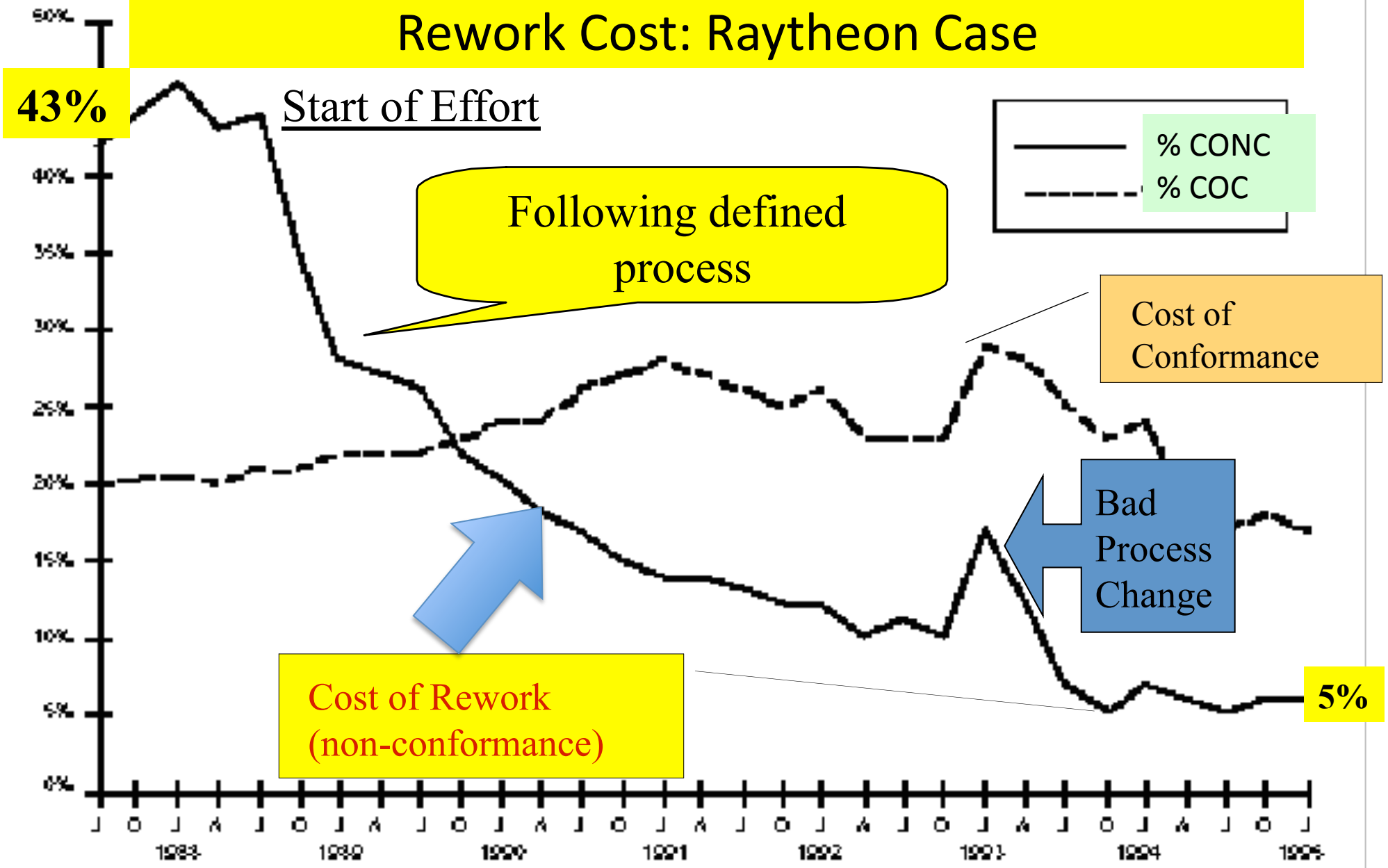
Estimated hours to find and correct later in test, or in field

Trevor Reeve



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

Rework Cost: Raytheon Case



http://www.sei.cmu.edu/pub/documents/95.reports/pdf/tr017.95.pdf

End 1988

Raytheon

End 1994

Figure 8: Cost of Quality Versus Time

‘Avoidable’ 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 Philip Crosby’s Model according to Raytheon95 Fig. 7

Source: Raytheon Report 1995
http://www.sei.cmu.edu/pub/documents/95.reports/pdf/tr017.95.pdf

Principle 3.

- **Performance, including quality requirements must be specified quantitatively.**
- **A ‘Scale’ and a future point on the scale must be defined.**

A 'Bad' Requirement
"Rock solid robustness"

- *While **robustness** is an **essential** HORROR requirement in all its uses, it is especially critical in MINING applications where the much longer job durations afford software defects (e.g. memory leaks) a greatly expanded opportunity to surface.*
- In this regard,
- HORROR will provide the following features or attributes:
  - ***Minimal down-time***
- A critical HORROR objective is to have **minimal downtime** *due to* **software failures**.
- This objective includes:
  - **Mean time between forced restarts > 14 days**
- HORROR's goal for mean time between forced restarts **is greater than 14 days**.
- *Comment: This figure does not include restarts caused by hardware problems, e.g. poorly seated cards or communication hardware that locks up the system. MTBF for these items falls under the domain of the hardware groups.*
  - **Restore system state < 10 minutes**
- Log scripts and test scripts, subsystem tests
  - ***Built-in testability***
- HORROR will provide the following features and attributes to facilitate testing.
  - **Tool simulators**

- *GILB COMMENT:*
  - *For once a reasonable attempt was made to quantify the meaning of the requirement!*
  - *But is could be done much better*
  -
  - *As usual the **set of designs** to **meet the requirement** do not belong here.*
  - *And none of the designs make any **assertion** about how well (to what degree) they will meet the defined numeric requirements.*
  - *And as usual another guarantee of eternal costs in pursuit of a poorly defined requirement is most of the content.*

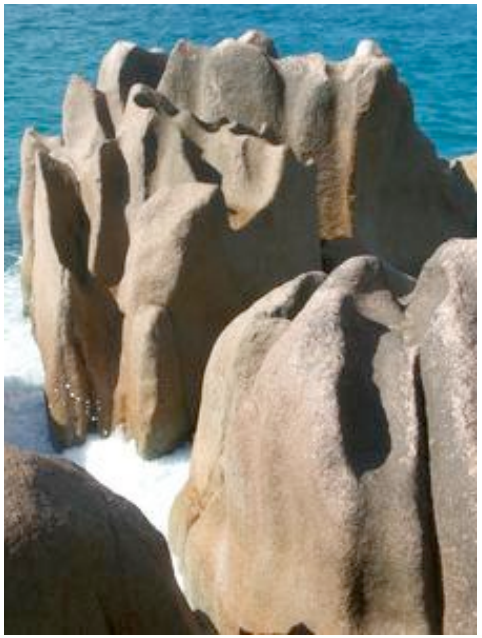
Real case of requirement for project costing over $100,000,000 without delivering testable results

Better Testable Definition of the Requirement:

<u>Rock Solid Robustness</u>:

Type: *Complex* Product Quality Requirement.

Includes: { Software Downtime, Restore Speed, Testability, Fault Prevention Capability, Fault Isolation Capability, Fault Analysis Capability, Hardware Debugging Capability}.



Defining One Component Clearly:

Software Downtime:
Type: Software Quality Requirement.
Ambition: *to have minimal downtime due to software failures <- HFA 6.1*
***Issue**: does this not imply that there is a system wide downtime requirement?*

Scale: <mean time between forced restarts for defined [Activity], for a defined [Intensity].>

Fail [Any Release or Evo Step, Activity = Recompute, Intensity = Peak Level] **14 days** <- HFA 6.1.1

Goal [By 2008?, Activity = Data Acquisition, Intensity = Lowest level] **: 300 days** ??
Stretch: **600 days**

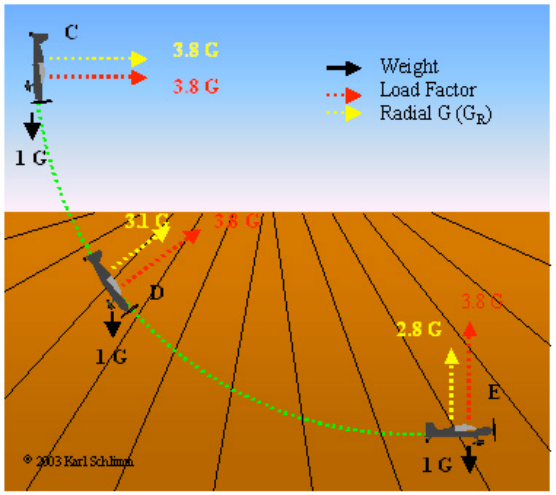
Defining a Second Component Clearly:

Restore Speed:

Type: Software Quality Requirement.

Ambition: Should an error occur (or the user otherwise desire to do so), Horizon shall be able to restore the system to a previously saved state in less than 10 minutes. <-6.1.2 HFA.

Scale: Duration from Initiation of Restore to Complete and verified state of a defined [Previous: Default = Immediately Previous]] saved state.



Initiation: defined as {Operator Initiation, System Initiation, ?}. Default = Any.

Goal [Initial and all subsequent released and Evo steps] 1 minute?

Fail [Initial and all subsequent released and Evo steps] 10 minutes. <- 6.1.2 HFA

Catastrophe: 100 minutes.

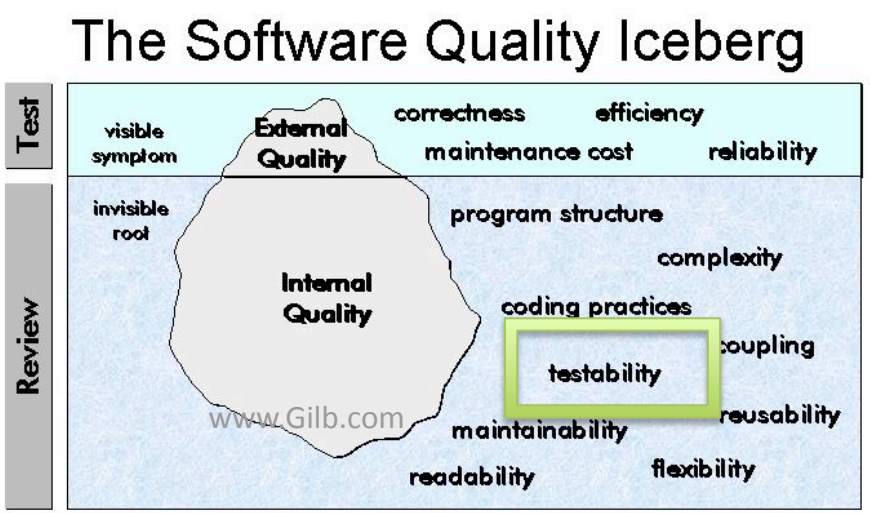
Defining a Third Component Clearly:

Testability:
Type: Software Quality Requireme
Version: 20 Oct 2006-10-20
Status: Demo draft,
Stakeholder: {Operator, Tester}.
Ambition: Rapid-duration automatic testing of <critical complex tests>, with extreme operator setup and initiation.

Scale: the duration of a defined [Volume] of testing, or a defined [Type], by a defined [Skill Level] of system operator, under defined [Operating Conditions].

Goal [All Customer Use, Volume = 1,000,000 data items, Type = WireXXXX Vs DXX, Skill = First Time Novice, Operating Conditions = Field, {Sea Or Desert}. **<10 mins.**

Design Hypothesis: *Tool Simulators, Reverse Cracking Tool, Generation of simulated telemetry frames entirely in software, Application specific sophistication, for drilling – recorded mode simulation by playing back the dump file, Application test harness console <-6.2.1 HFA*



Principle 4.

- **Requirements must be correctly "Typed"**
  - **(Function, Quality, Constraint etc.)**

Requirement Types

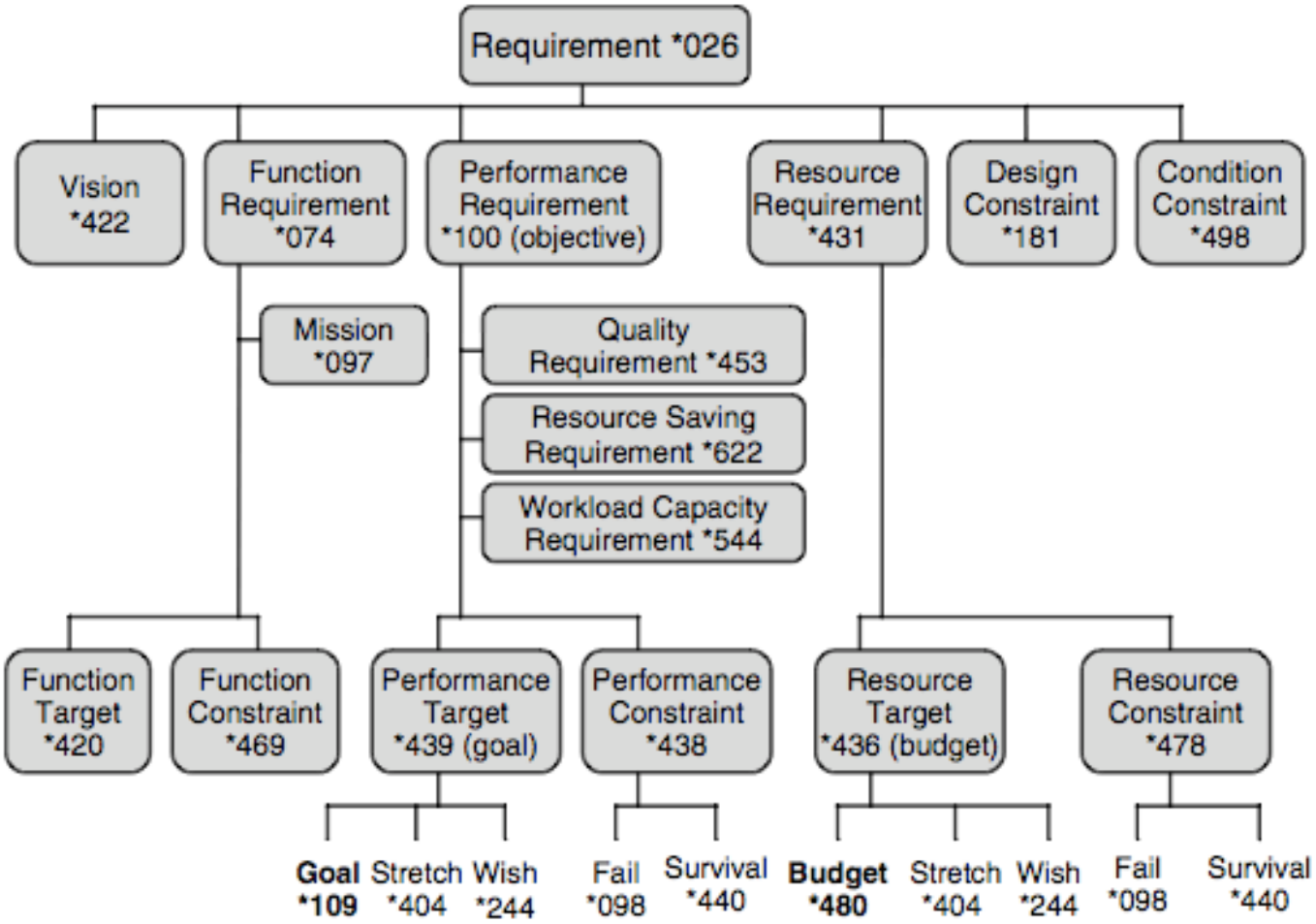
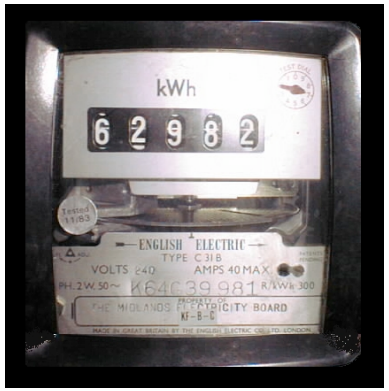


Figure G20
Requirement Concepts.

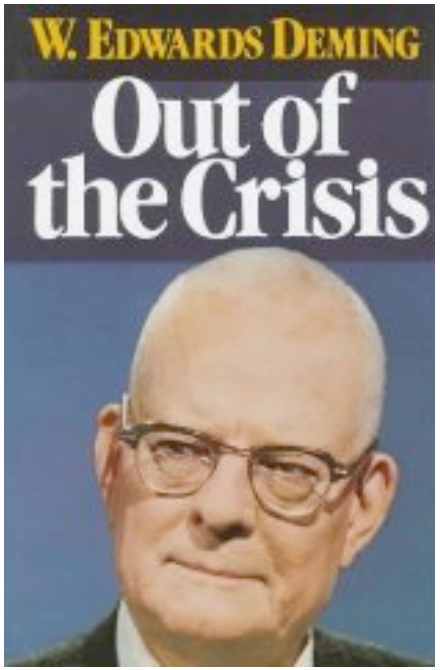
Type determines Test

- **Type: Function Requirement.**
  - **Test for presence**
- **Type: Quality or Performance Requirement**
  - **Test along the scale using a defined 'Meter' (test process)**
- **Test: Constraint**
  - **Test to see if the defined constraint (example Legal) is violated or not**

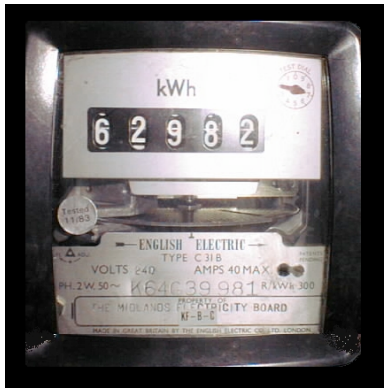
Meter Concept *093



- **A 'Meter' parameter is used to**
  - **identify, or specify,**
  - **the definition of a practical measuring device, process, or test**
  - **that has been selected for use in measuring a numeric value (level) on a defined Scale.**
- *"there is nothing more important for the transaction of business than use of operational definitions." (W. Edwards Deming, Out of the Crisis (Deming 1986))*



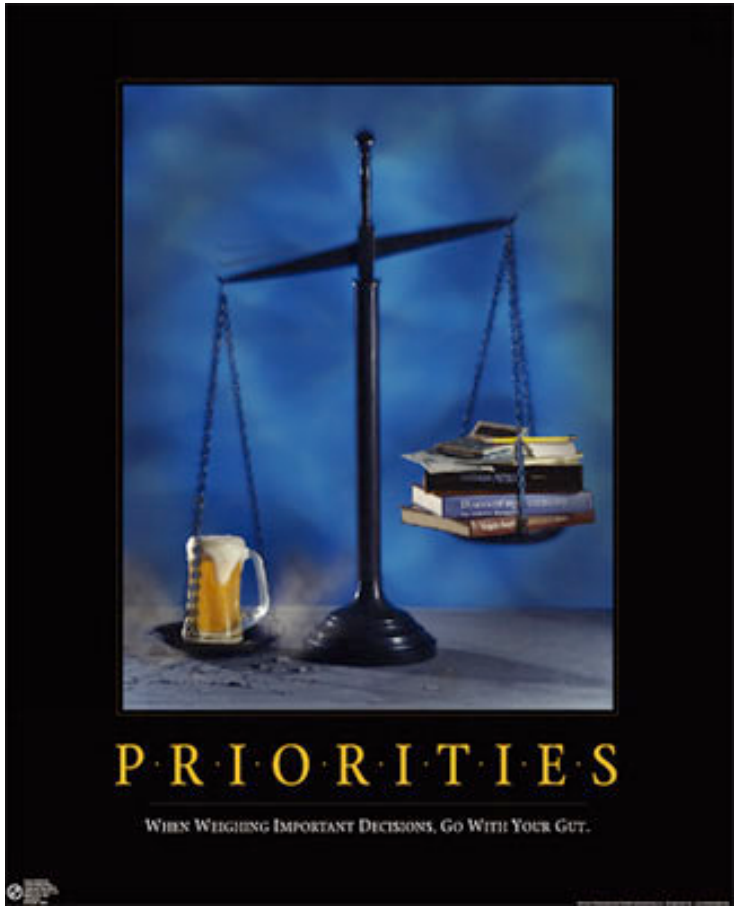
Example of ‘Meter’ Use



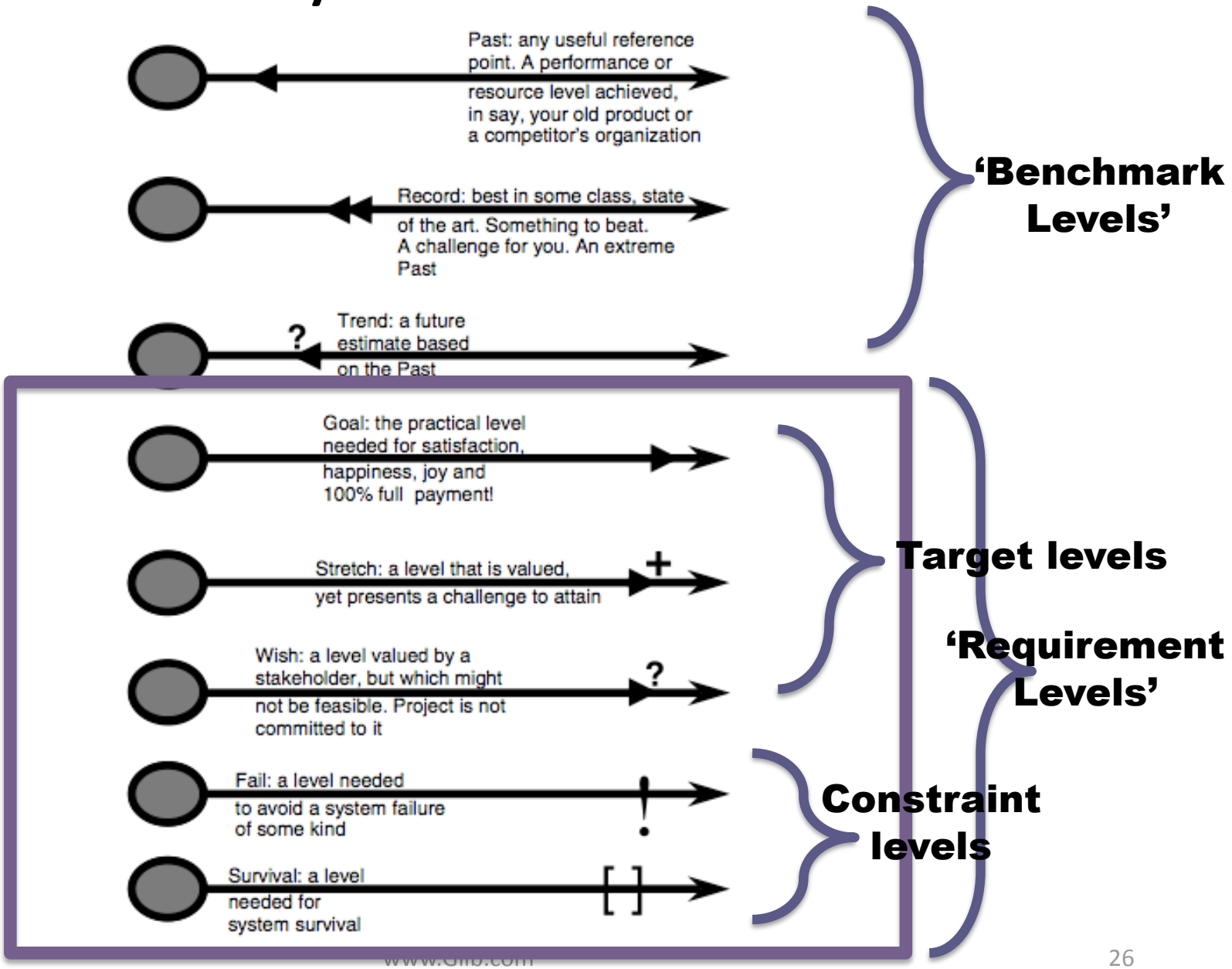
- **Satisfaction:**
- **Scale: Percentage of <satisfied> Customers.**
  - **Meter [New Product, After Launch]: On-site survey after 30 days use for all Customers.**
- **Past [This Year, USA]: 30%.**
  - **Meter [Past]: Sample of 306 out of 1,000+ Customers.**
- **Record [Last Year, Europe]: 44%.**
  - **Meter [Record]: 100% of Customers. Goal [After Launch]: 99% <- Marketing Director**

Principle 5.

- **The requirement level *priorities* must be specified**
  - **(Survival, Fail, Goal, Stretch)**



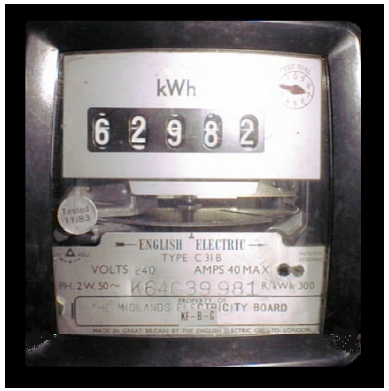
Levels of System Performance



Priority

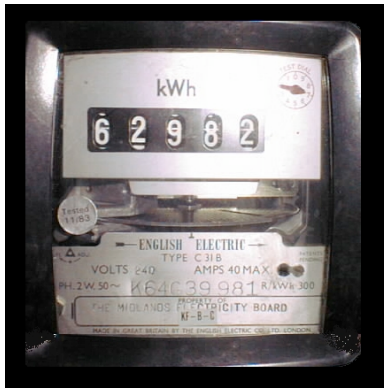
1. **Survival**
2. **Fail/ Tolerable**
3. **Goal**
4. **Stretch**
5. **Wish**
6. **Ideal**

Principle 6.



- One or more high level ‘Meter’ specifications should have been agreed
- and integrated into the specification initially.

‘Meter’ Specifications



- **<u>Task Productivity</u>:**
- **Type: System Level Critical Quality Requirement**
- **Scale: Average Time to Correctly Complete defined [Task] using defined [Employee] under defined [Circumstances]**
- **Meter:**
  - **Over 100 random representative instances of Tasks will be tested**
  - **for all defined Employee Types**
  - **under all defined Circumstances.**

Principle 7.

- **Pointers**
  - **–to corresponding Test Plans, Test Scripts, Test Results, and Test Responsible people**
  - **–should be integrated into the requirement specification itself**
  - **–by the test planners.**

Adding Information to the 'Meter'

- Task Productivity:
- Type: System Level Critical Quality Requirement
- Version: 10 September 2008: 15:14 <- TsG
- Scale: Average Time to Correctly Complete defined [Task] using defined [Employee] under defined [Circumstances]
- Meter: Over 100 random representative instances of Tasks will be tested for all Employee Types under all defined Circumstances.
  - Approved: User Committee, 10 Sept. 2008
  - **Test Plan: <not made yet, Task Productivity.Test Plan>.**
  - **Scripts: <Task Productivity.Scripts>**
  - **Test Results: <Task Productivity.Outputs>**
  - **Test Responsible: Project Test Planner (TG).**
  - **First Test: <scheduled 1 Oct 2008>.**
  - **Last Test: <none>.**

Principle 8.

- **Version Control**
  - **each *requirement* change must be sent to responsible test planners**
    - **(automatically)**

Making ***Test*** Aware of Requirement Changes in Real Time

Requirement Change Inspection:

Are all related instances (like Test) formally notified of the change?

Change Notification Recorded

Task Productivity:

***Type*: System Level Critical Quality Requirement**

***Version* : 11 September 2008: 15:00 <- TsG**

***Scale* : Average Time to Correctly Complete defined [Task] using defined [Employee] under defined [Circumstances]**

***Meter* : Over 1,000 random representative instances of Tasks will be tested for all Employee Types under all defined Circumstances.**

***Test Responsible* : Project Test Planner (TG).**

***Changes Emailed* : 1 September 2008: 16:00**

Principle 9.

- **The Total set of conditions for a requirement**
- **will be specified in one or more [Qualifier] statements.**

[Conditions] – ? cases to test

Task Productivity:

Type: System Level Critical Quality Requirement

Version: 11 September 2008: 15:00 <- TsG

Scale: Average Time to Correctly Complete defined [Task] using defined [Employee] under defined [Circumstances].

Goal [Task = Initiate, Employee = Any, Circumstances = Stressful] 10 minutes.

Goal [Task = Initiate, Employee = New Hire, Circumstances = Training] 20 minutes.

Stretch Goal [Task = Initiate, Employee = New Hire, Circumstances = Training] 15 minutes.

Task: {Initiate, Process, Complete, Correct}.

Employee: {New Hire, Average, Expert, Manager, Any, All}.

Circumstances: {Training, Stressful, Any, All, Normal}.

Test All combinations

Principle 10.

- The *Priority Information* about a requirement
  - will be suitable
  - to help test planners understand
  - the priority of
    - test quality
    - and timeliness.

Understanding Test Timeliness

- **Task Productivity:**
- **Type: System Level Critical Quality Requirement**
- **Version: 11 September 2008: 15:00 <- TsG**
- **Scale: Average Time to Correctly Complete defined [Task] using defined [Employee] under defined [Circumstances].**
- **Goal [Task = Initiate, Employee = Any, Circumstances = Stressful, Deadline = June 2009] 10 minutes.**
- **Goal [Task = Initiate, Employee = New Hire, Circumstances = Training, Deadline = December 2009] 20 minutes.**

Understanding Test *Quality*

- Task Productivity:
- Type: System Level Critical Quality Requirement
- Version: 11 September 2008: 15:00 <- TsG
- Scale: Average Time to Correctly Complete defined [Task] using defined [Employee] under defined [Circumstances].
- **Meter: Over 1,000 Representative instances of Tasks will be tested for all Employee Types under all defined Circumstances.**
  - **Representative: by Frequency of tasks, and % of Employee, and proportion times of Circumstances**

Testers Bill of Rights: ©Tom Gilb

1. **Testers have the right to sample their process inputs, and reject poor quality work (no entry).**
2. **Testers have the right to unambiguous and clear requirements.**
3. **Testers have the right to test evolutionarily; early as the system increments.**
4. **Testers have the right to integrate their test specifications into the other technical specifications.**
5. **Testers have the right to be a party to setting the quality levels they will test to.**
6. **Testers have the right to adequate resources to do their job professionally.**
7. **Testers have the right to an even workload, and to have a life.**
8. **Testers have the right to specify the consequences of products that they have not been allowed to test properly.**
9. **Testers have the right to review any specifications that might impact their work.**
10. **Testers have the right to focus on testing of agreed quality products, and to send poor work back to the source.**

Why should testers have any rights?

- **Main Argument:**
  - **Because it will reduce total costs, time and increase quality at the same time.**
- **Real Reason (hidden agenda):**
  - **To make other project members do their own work properly in the first place.**
- **Altruistic Reason:**
  - **to make their workday more meaningful,**
  - **and to show them some due respect.**

What are testers going to *do* with their 'rights'?

- **Use them to negotiate service agreements with the rest of the project**
- **Use them to train testers in rational expectations**
- **Use them to set their own test process entry and exit standards**
- **Use them to enhance their own defined processes**
- **Use them as a starting argument;**
  - **when they are 'mistreated' by other project members**

Part 2:

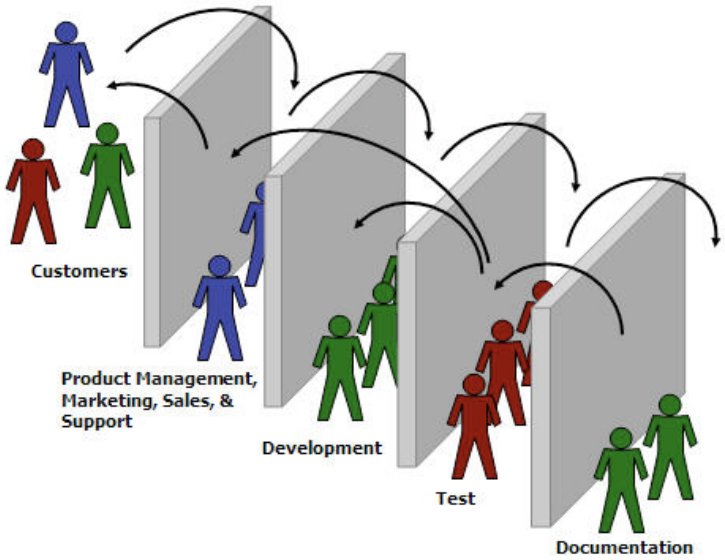
- **Quality Control:**
  - **How should we organise quality control inspections**
  - **of requirements and test planning**
  - **so as to drive people to better practices,**
  - **and to avoid wasting time on bad inputs?**

The direct answer?

- we should organise quality control inspections
- of requirements and test planning
- according to the following management policy.

Test QC Policy for Test Inputs and Outputs

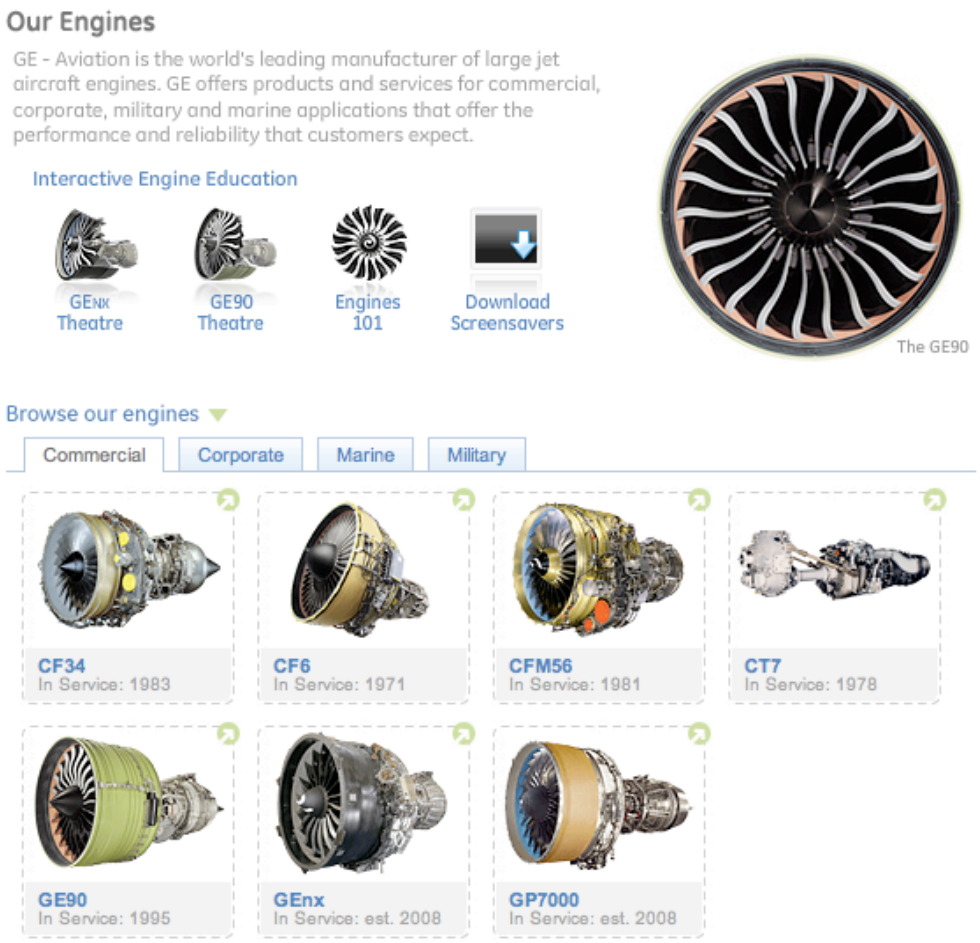
- All test process inputs (like requirements), and all test process human outputs (like test scripts, test plans, and test strategies) will be subject to strict, numeric Entry and Exit control.
- The primary numeric Entry/Exit Condition will be based on the specifications' degree-of-conformance to agreed standards.
  - As measured by Inspections, and consequent Major Defect Density Remaining Calculations
  - The Generic Acceptable level of major defects is maximum estimated 1.0 Majors/Logical Page (300 words)
- We will demand such a standard from our input suppliers, and we will demand that standard from ourselves.



The 30 minute agile review for requirements: how to measure defect density of requirements !

- How to measure requirement quality levels
  - Cheaply (15-60 minutes, 2 to 4 reviewers)
  - Reliably (accurately enough to sort out unacceptable work)
  - An 'Agile form of Inspections
- The requirement specification QC (Quality Control)
  - In simple terms, SQC measures the density of serious malpractice (Major defects) in relation to official written rules for good practice

Here is a real case example of such a **static test**, of a requirement specification



Case:
Real Agile Spec QC

- of System Requirements Specification (SRS)

of 82 pages for a major US corporation.

Framework

- Demonstration of power of Agile Inspection
  - 8 Managers
  - 2 hours
  - 4 real requirements specifications offered ,
    - One 82 page 'System Requirements Specification' actually used

We Introduced best-practice Rules for Requirements

- 1. **Unambiguous** to intended Readership
- 2. **Clear** enough to test.
- 3. **No** unintentional **Design**

We explained the definition of Spec Defect (1)

- A '**Specification Defect'** is a ***violation of a Specification Rule***
  - *–(violation of a 'standard')*
  - *– Note: If there are 10 ambiguous terms in a single requirement*
    - *then there are 10 defects!*

We further explained the definition of Spec Defect (2)

- **A 'Specification Defect' is a *also a 'Potential Defect'***

  - ***In the next level of specification***
    - ***Like in Design, test plans, code or test scripts.***
  - ***With about 1/3 chance (potential) of becoming a downstream real defect.***
  - ***In 'code' we call this a 'bug' (a potential malfunction).***
  - ***If we discover the bug in test or operation we call it a 'malfunction'***

The definition of Major defect

- **Major:**
  - **a Defect that *potentially***

costs more

– to find and fix

– later in the development process

– than it would cost *now*.

– We need to get rid of it NOW!

Agree with Management on Exit level

- **Exit Conditions:** (when Requirements can go forward to Design, Test, etc. with little risk)
  - **Maximum 1 Major Defect/ (Logical) Page, estimated remaining**
  - **Logical Page = 300** Non-commentary words.

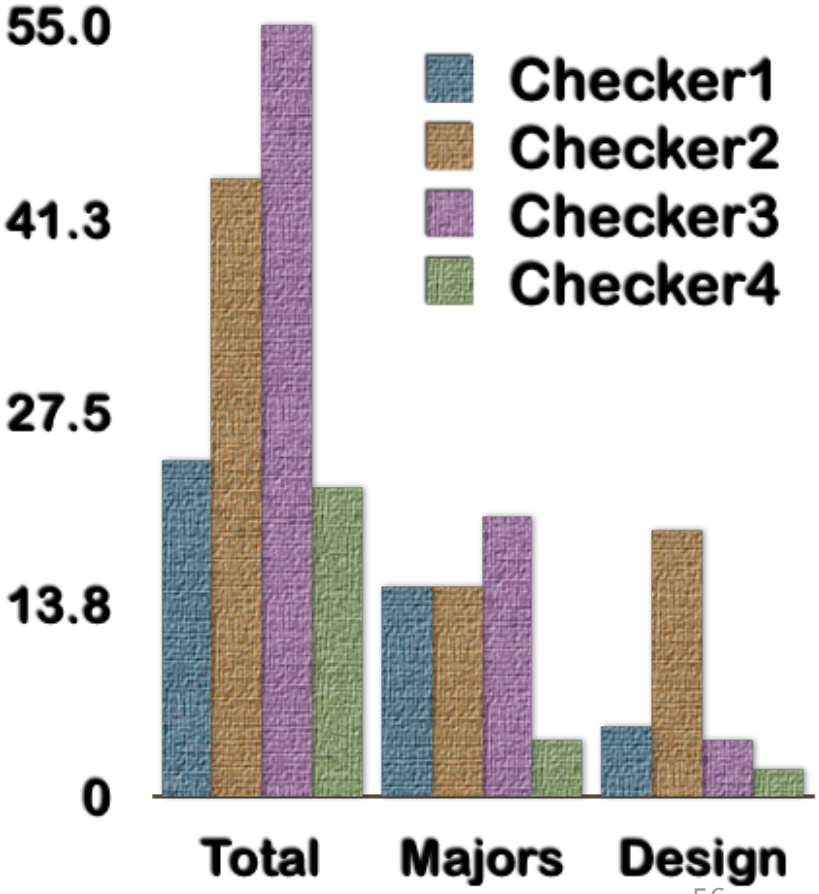
The notion of numeric exit:
When are requirements 'good enough' to build tests on !

- **A major defect in requirements has 33% chance of causing a bug or worse.**
- **Most IT shops are 'uncontrolled': have no standards enforced, no QC of requirements**
- **They have about 100 ±50 major defects per page**
  - **33 ± 10 potential bugs per page**
- **This is deemed completely unacceptable by managements, and in relation to cost and quality options**
- **This (max. 1 major/page) should be your requirements process exit standard, and also your test planning entry standard**

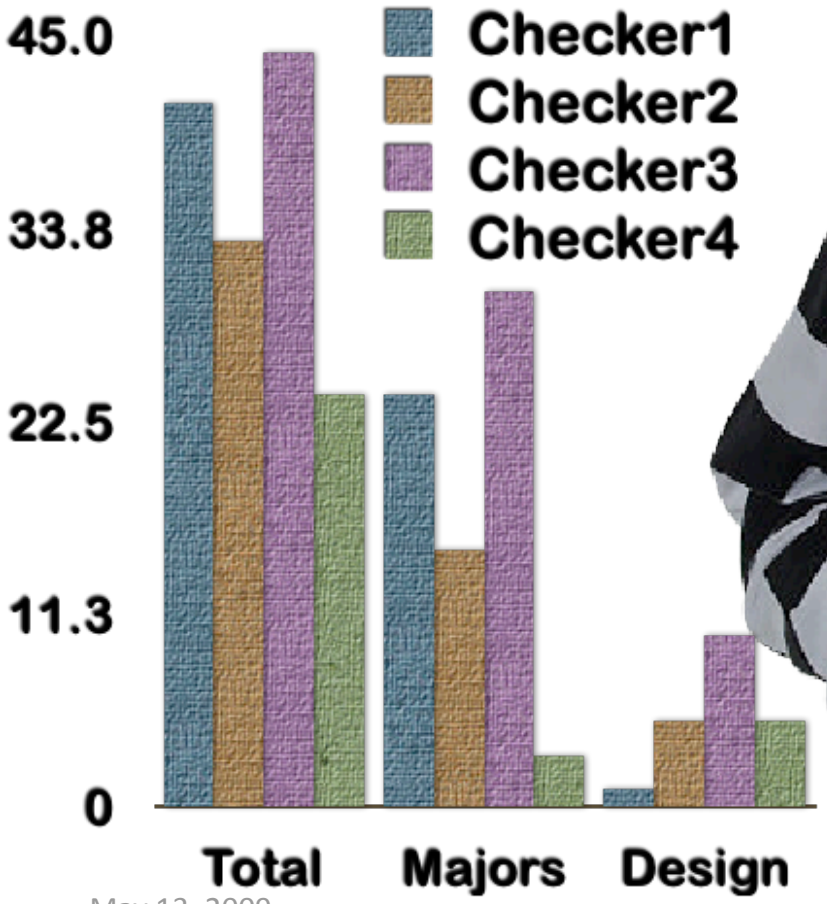
The assigned checking process

- **You have up to 30 minutes**
  - **check 1 *sample* requirements page (from an 82 page document)**
- **Count all *potential* Rule Violations**
  - **= Defects**
- ***Classify* Defects as Major or minor**

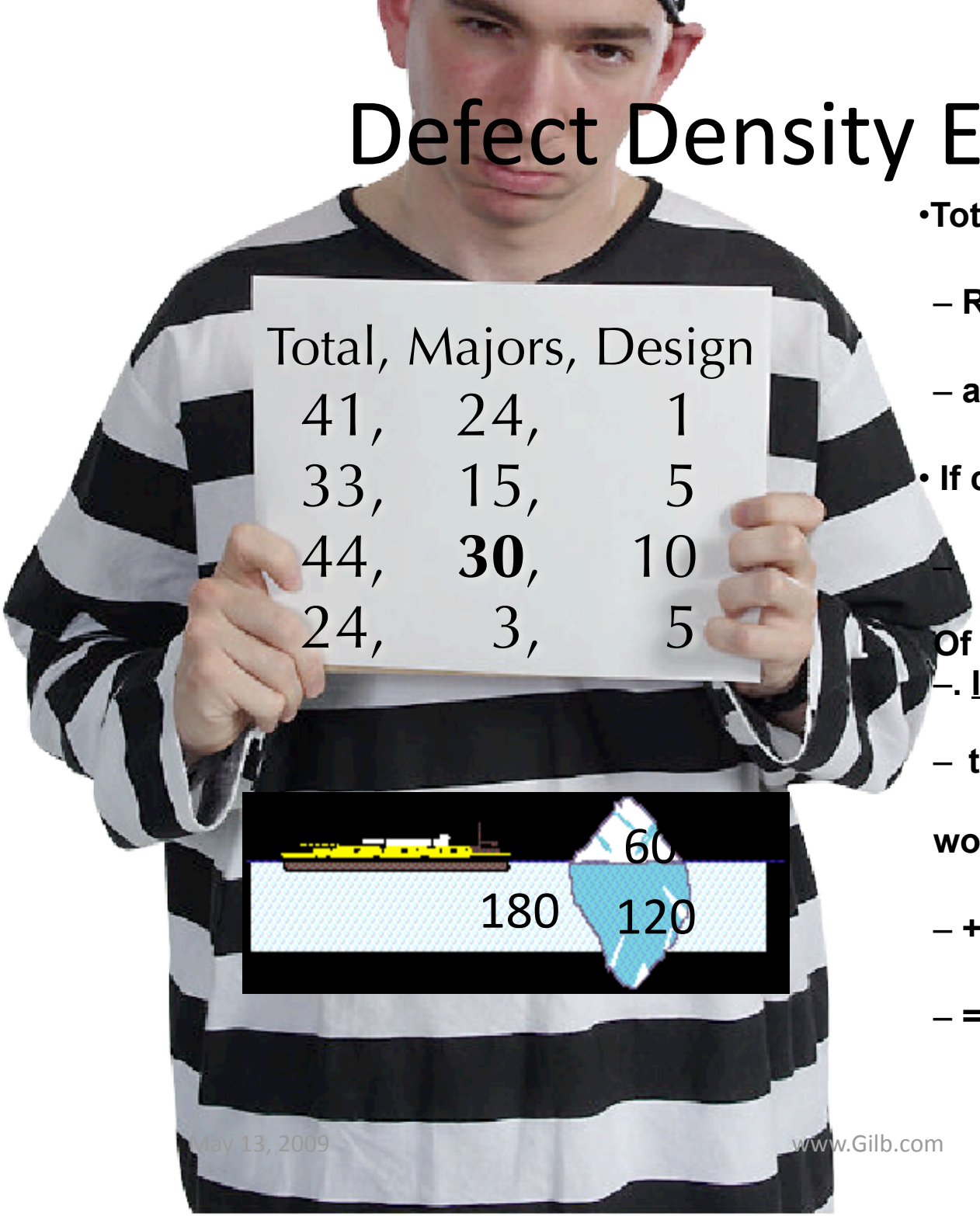
Report
Page 81



Report
Page 82



Defect Density Estimation



- **Total for group (page 82)**
  - **Rough Est. 30 x 2 = 60 Majors**
  - **assume 60 ±10 are unique.**
- **If checking is 33.33% effective,**
  - **total in page = 3 x 60 = about 180±30**

Of which 2/3 (or 120) were <u>not yet</u> found.

–. <u>If</u> we fix all we found (60),

– then the estimated remainder of Majors would be 120 (not found)

– +10 "not fixed correctly"

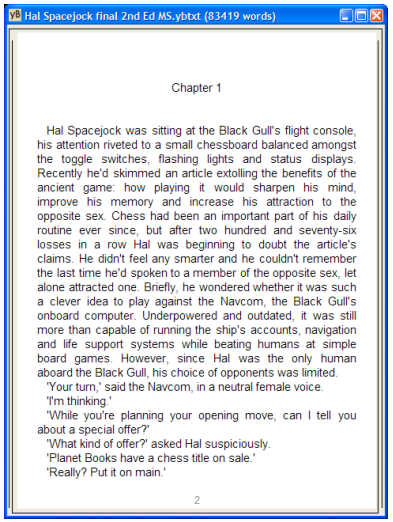
– = 130 Majors remaining.

Conclusions

- **Human *defect removal* by Inspections/reviews/SQC is**
  - **a hopeless cause: not worth it.**
- **Spec QC *can* be used, in spite of imperfect effectiveness,**
  - **to '*accurately enough' estimate* 'major-defect-level' density.**
- **EXPERIENCE AND CONSEQUENCES:**
- **This measurement can be used to motivate engineers to**
  - **dramatically (100x! Over about 7 learning cycles)**
    - **reduce their defect insertion (rule violation)**
  - **to a *practical* exit level**
    - **(like *less than* 1.0 Majors/page)**

Extrapolation to Whole Document

- **Average: 150 Majors/page**

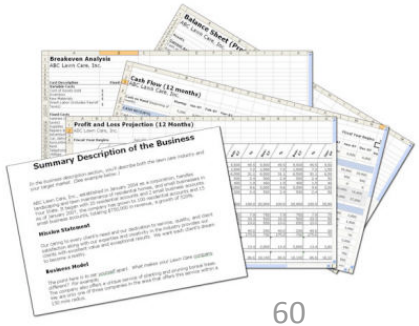


- Page 81: 120 majors/page

- Page 82: 180 Majors/page

- **Total in whole document:**

– 12,300 Majors



- 150 Majors/page x 82 pages.

Estimated Project Loss

- If a Major has
  - 1/3 chance of causing loss
- And each loss caused by a Major is
  - avg. 10 hours
  - then total project Rework cost is
  - about *41,000 hours loss.*
- (This project <u>was</u> over a year late)
  - 1 year = 2,000 hours x 10 people

Agile Spec QC Procedure

P1: Identify Checkers: Two people, maybe more, should be identified to carry out the checking.

P2: Select Rules: The group identifies about three rules to use for checking the specification. (My favorites are clarity ('clear enough to test'), unambiguous ('to the intended readership') and completeness ('compared to sources'). For requirements, I also use 'no optional design'.)

P3: Choose Sample(s): The group then selects sample(s) of about one 'logical' page in length (300 non-commentary words). Choosing such a page at random can add *credibility* – so long as it is *representative* of the content that is subject to quality control. The group should decide whether all the checkers should use the same sample, or whether *different* samples are more appropriate.

P4: Instruct Checkers: The SQC team leader briefly instructs the checkers about the rules, the checking time, and how to document any defects, and then determine if they are *major* defects (majors).

P5: Check Sample: The checkers use between 10 and 30 minutes to check their sample against the selected rules. Each checker should 'mark up' their copy of the document as they check (underlining issues, and classifying them as 'major' or not). At the end of checking, each checker should count the number of 'possible majors' (spec defects, rule violations) they have found in their page.

P6: Report Results: The checkers each report to the group their number of 'possible majors.' Each checker determines their number of majors, and reports it.

P7: Analyze Results: The SQC team leader extrapolates from the findings the number of majors in a single page (about 6 times the most majors found by a single person, or alternatively 3 times the unique majors found by a 2 to 4 person team). This gives the major-defect density estimate. If using more than one sample, you should average the densities found by the group in different pages. The SQC team leader then multiplies the 'average major defects per page density' by the 'total number of pages' to get the 'total number of major defects in the specification' (for dramatic effect!).**

P8: Decide Action: If the number of majors per page found is a large one (ten majors or more), then there is little point in the group doing anything, except determining how they are going to get someone to write the specification 'properly', meaning to acceptable exit level. There is no economic point in looking at the other pages to find 'all the defects', or correcting the majors already found. There are simply too many majors not found.

P9: Suggest Cause: The team then chooses any major defect and thinks for a minute why it happened. Then the team agrees a short sentence, or better still a few words, to capture their verdict.

Agile SQC Process

Entry Conditions

- A group of two, or more, suitable people* to carry out Agile SQC is assembled in a meeting.
- The people have sufficient time to complete an Agile SQC. Total Elapsed Time: 30 to 60 minutes.
- There is a trained SQC team leader at the meeting to manage the process.

- **Exit Conditions**
- Exit if less than 5 majors per page extrapolated total density, or if an action plan to ‘rewrite’ the specification has been agreed.

The formal Agile SQC Process Sources

- Cutter 5 pg Paper
- http://www.gilb.com/tiki-download_file.php?fileId=64
- INCOSE SQC Paper http://www.gilb.com/tiki-download_file.php?fileId=57

- Agile SQC Slides with Standard for Process
- http://www.gilb.com/tiki-download_file.php?fileId=239

Part 3:

- Systematic Reduction of Defect Injection:
  - The Defect Prevention Process:
  - how can we organise ourselves at the test level
  - to analyse recurrent problems
  - and implement practical changes to avoid them?

Cost of Quality over Time: Raytheon 95

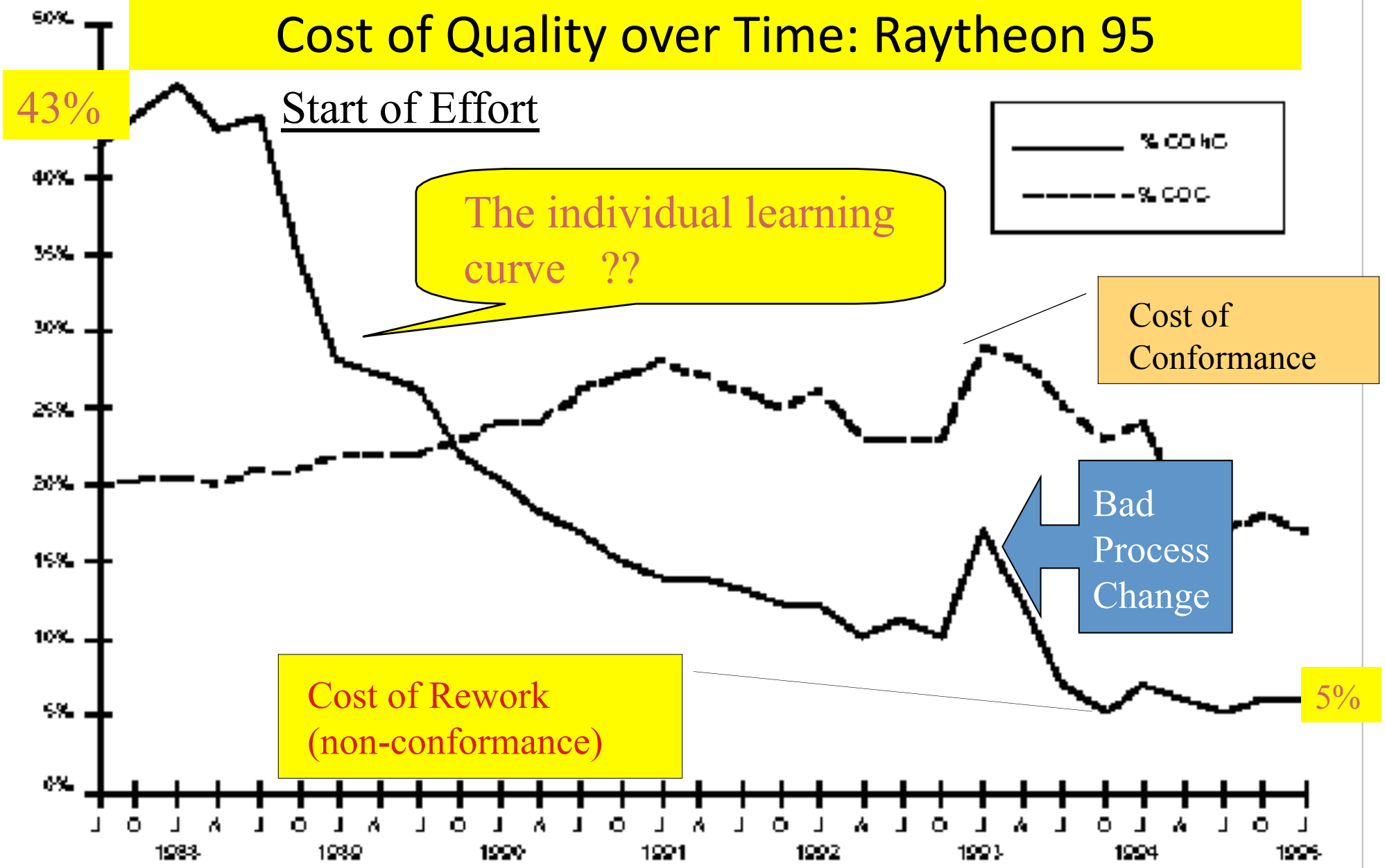


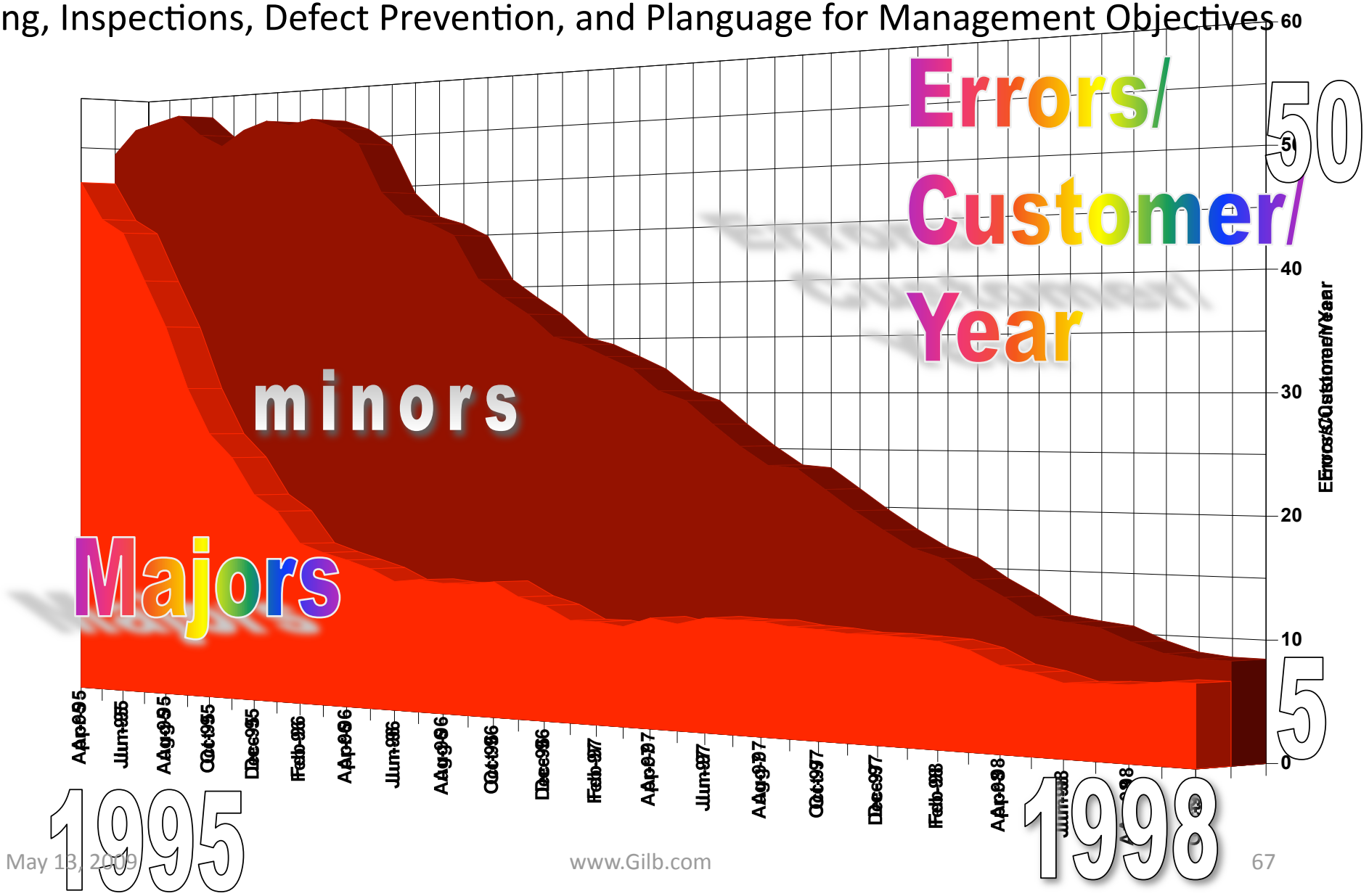
Figure 8: Cost of Quality Versus Time

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



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

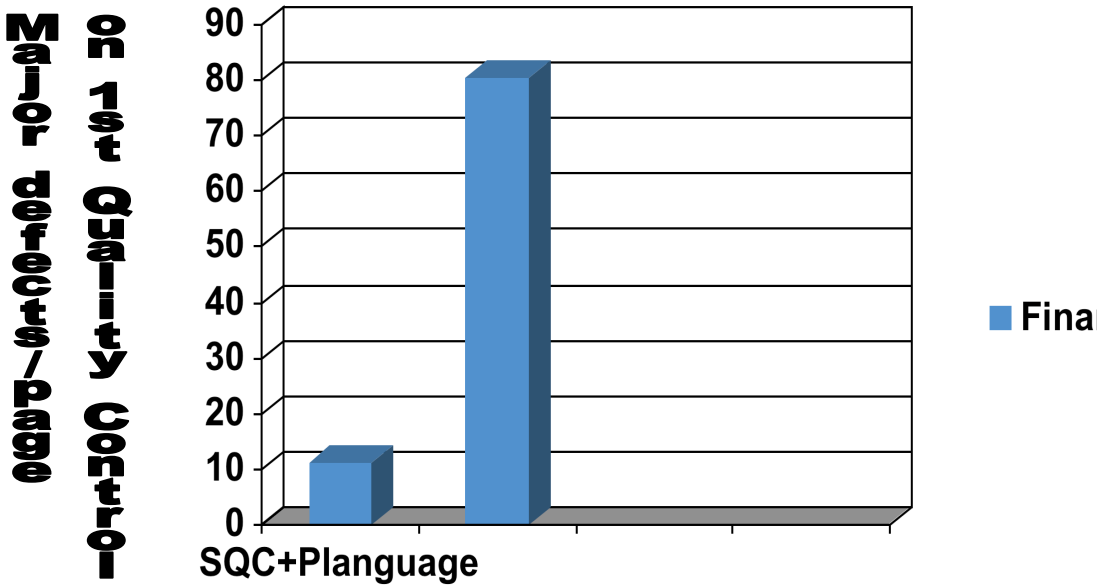
Across 18 DV (DeVelopment) Projects using the new requirements method, the average major defect rate on first inspection is 11.2.

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.

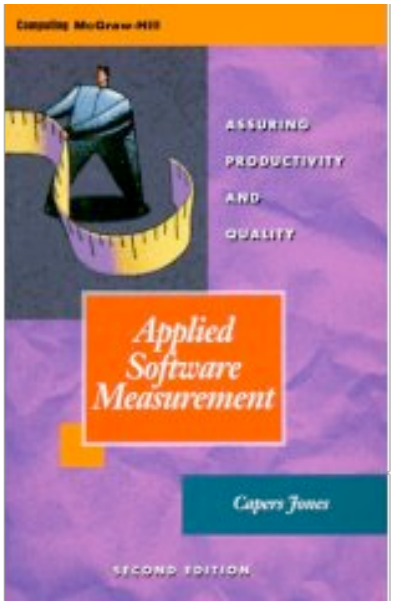


Capers Jones

Defect *Removal* Effectiveness Inspections and Tests

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

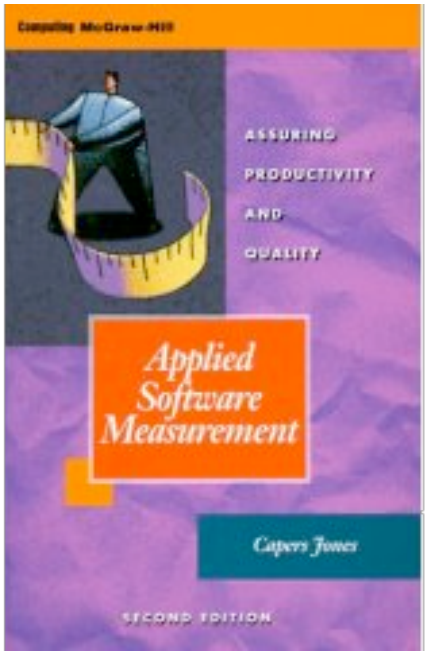
Defect Removal Activity	Ranges of Defect Removal Effectiveness
Informal design reviews	25% to 40%
Formal design inspections	**45% to 65%**
Informal code reviews	20% to 35%
Formal code inspections	**45% to 70%**
Unit test	**15% to 50%**
New function test	**20% to 35%**
Regression test	**15% to 30%**
Integration 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%**



Capers Jones

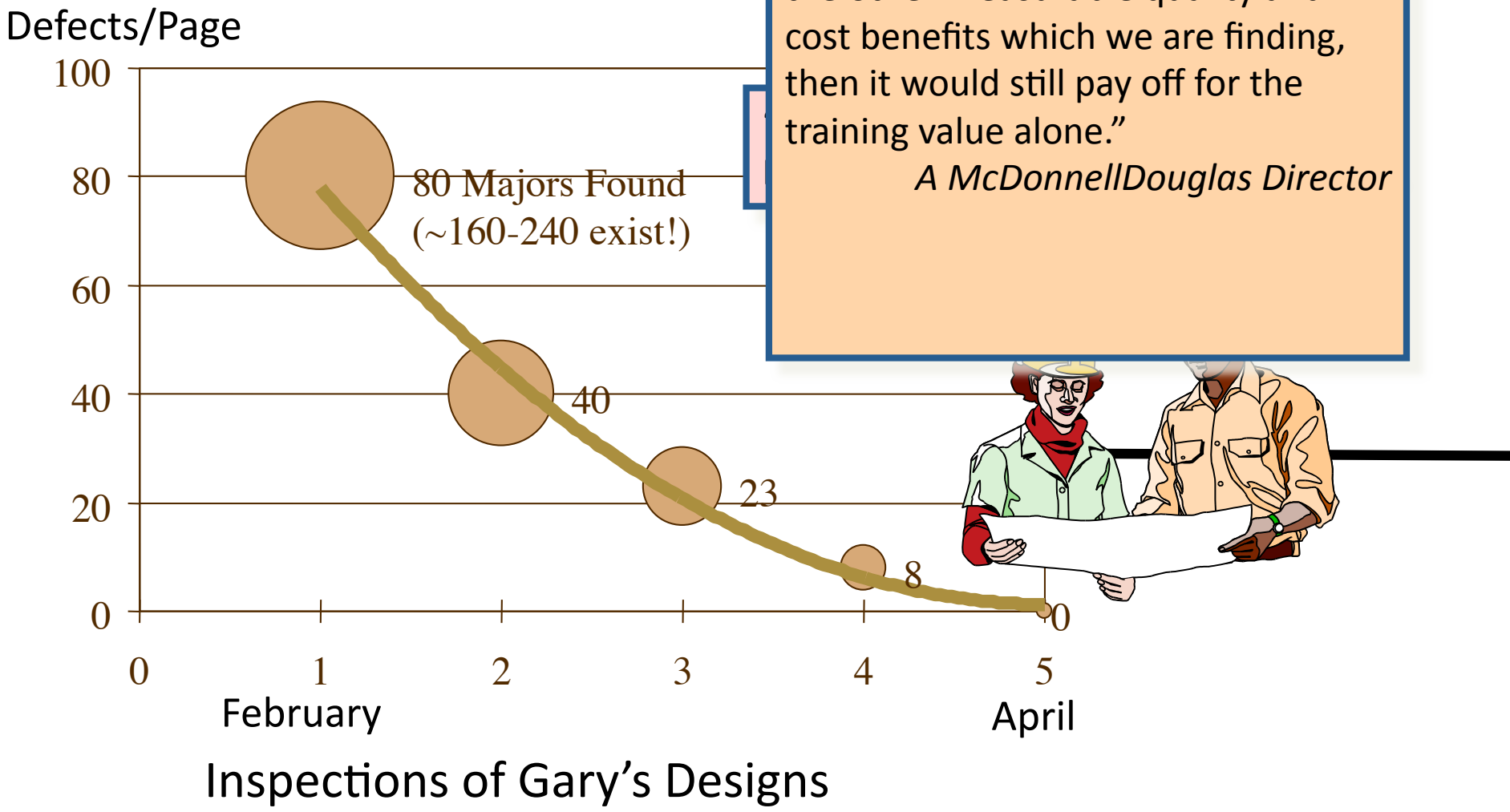
No 'Silver Bullet' Solution Machine Guns Kill Defects

- **"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 (he means 'effectiveness') levels > 95%".**
    - **Jones, Capers; Applied Software Measurement; McGraw Hill, 2nd edition 1996; ISBN 0-07-032826-9; 618 pages.**



Positive Motivation

Personal Improvement



"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."

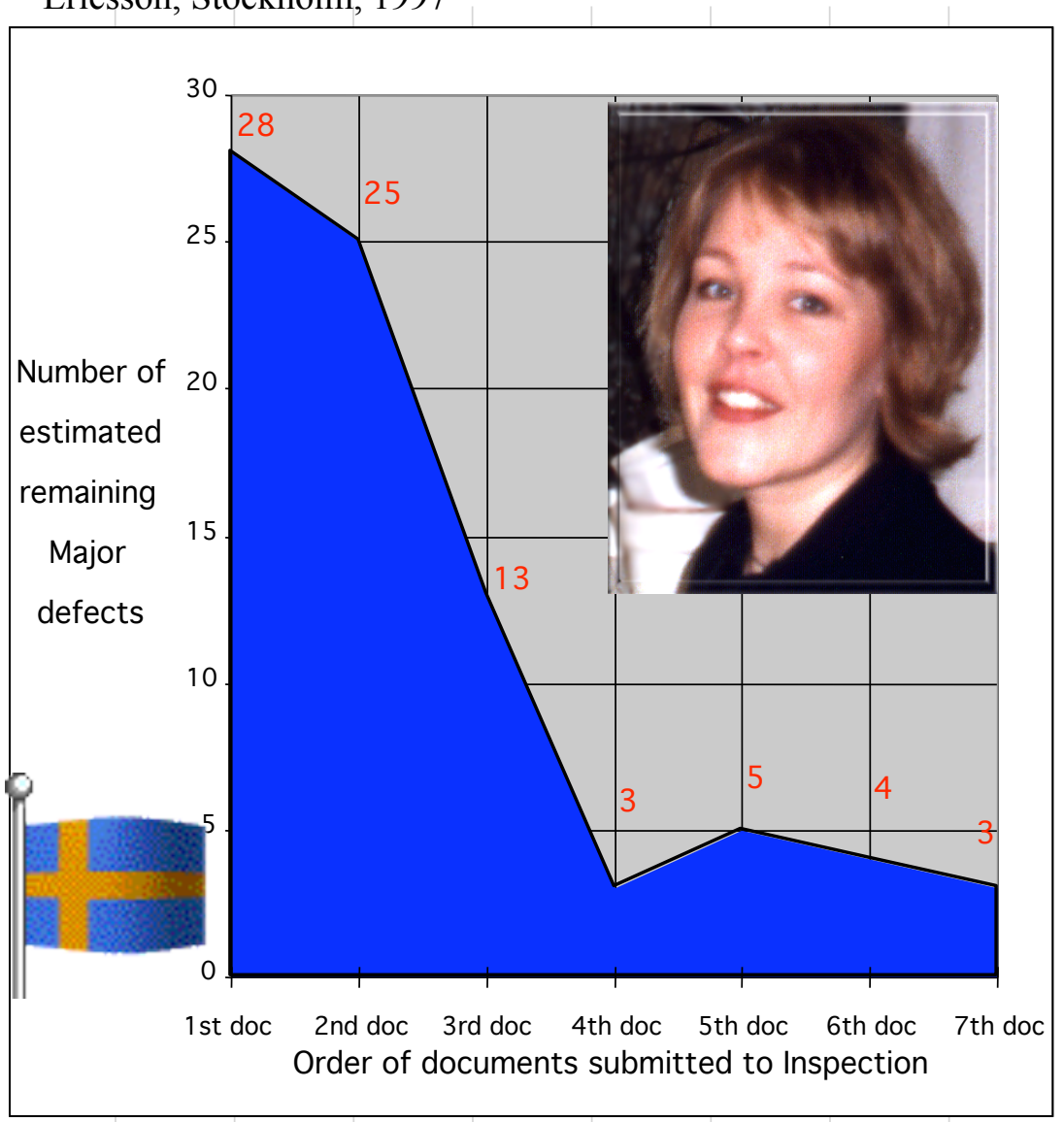
A McDonnellDouglas Director

Individual learning Curve

- 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,

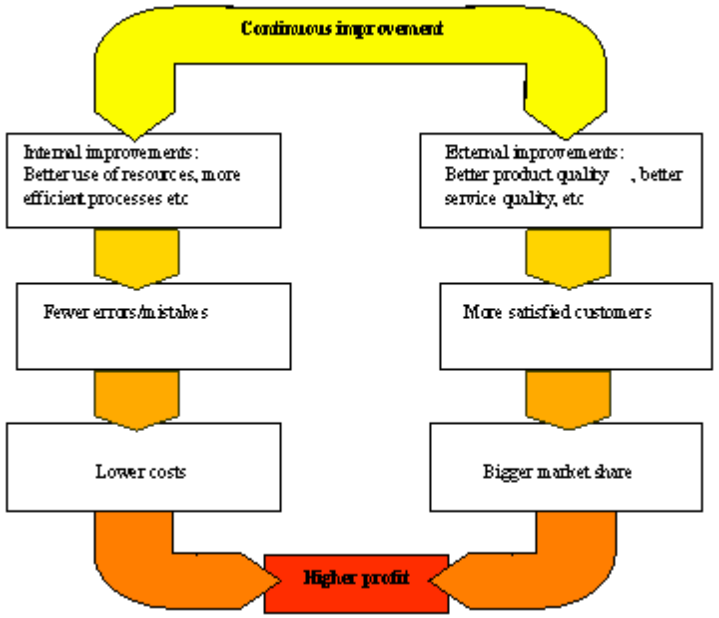
Ericsson, Stockholm, 1997



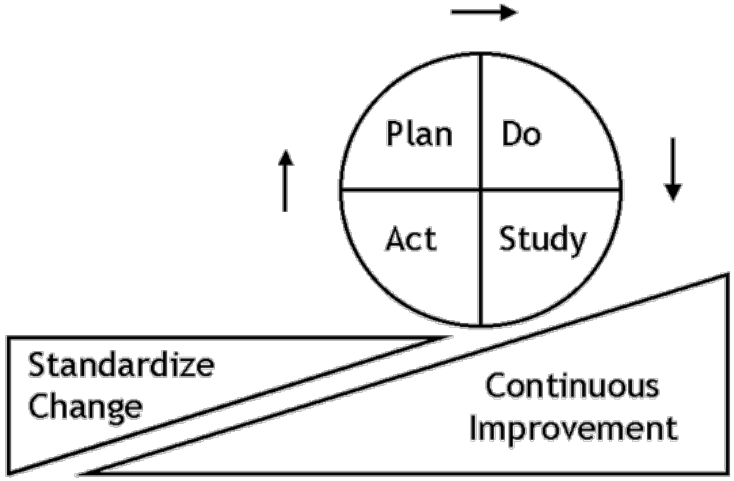
See also the Raytheon Learning Curve

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



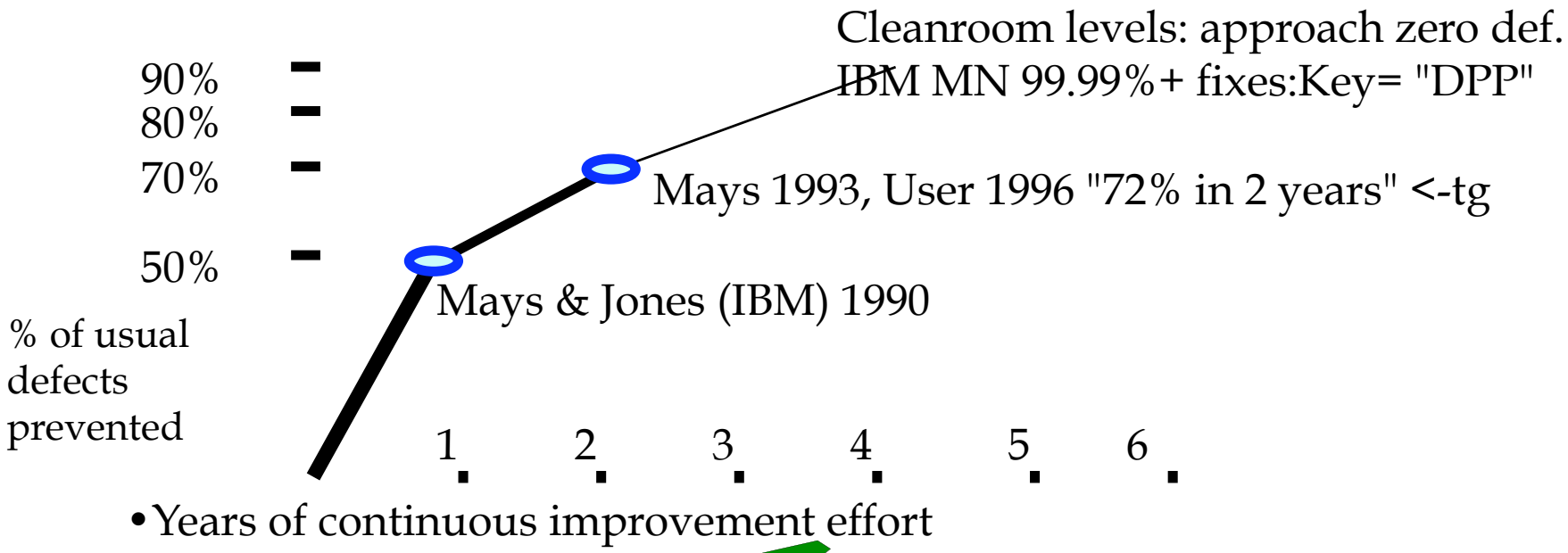
Prevention Costs

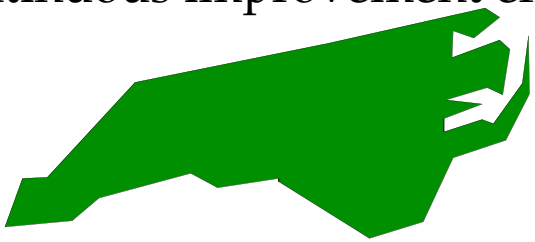


Deming Cycle

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

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

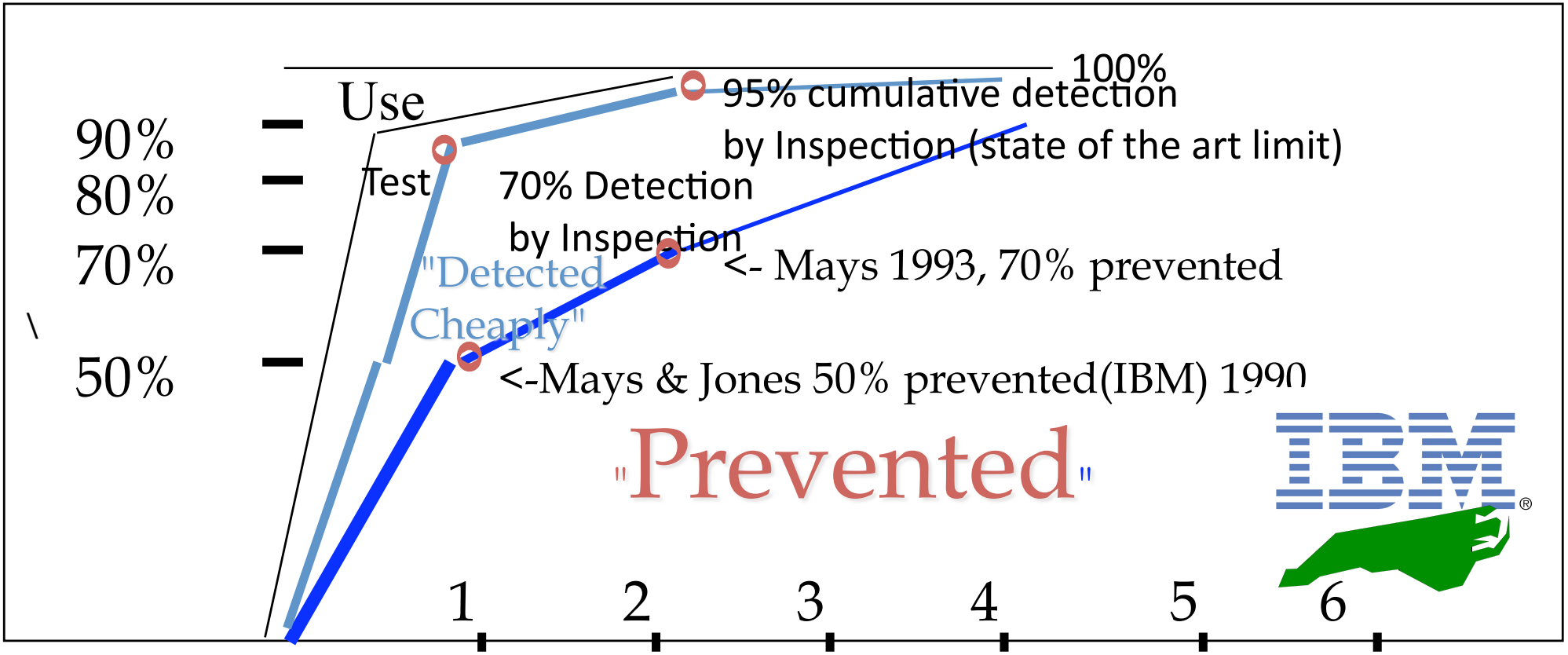




North Carolina

IBM Research Triangle Park Networking Laboratory

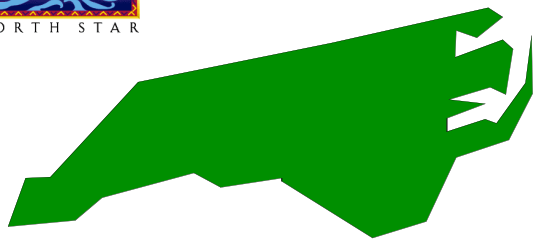
Prevention + Pre-test Detection is the most effective and efficient



- 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

- 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



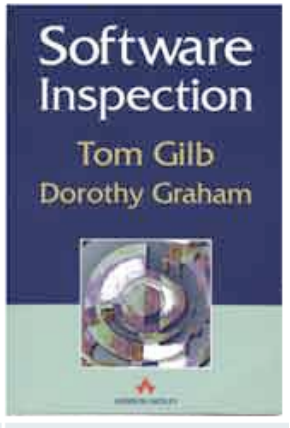
Dealing with your wayward requirements writers: How to get them to write decent requirements

- IT Management must take charge
- Policy:
  - Requirements will be written to our standards
  - If not they are not deemed acceptable for any purpose
  - Requirement quality levels (majors/page) will be measured as basis for process exit and entry.
  - The maximum level will be 1 major/page.
- People who cannot loyally follow this policy should leave the organisation.M

Conclusion

- Set decent standards
- Measure adherence to your standards
- Measure consequences (improvements) of adherence (bugs and delays reduced sharply)
- Propose necessary changes to your IT Management
- Lead by example in Test Specification!
- Measure all input to test process and report back
  - Have your own standards for acceptable inputs to test.

Source of Standards for Good Requirements



End slide

How to Evaluate if Requirements Specifications are Good Enough for Testing: The Agile Inspection Measurement Process and Numeric Exit

Tom Gilb

(TOM@GILB.COM, WWW.GILB.COM)