

**Software  
Inspection**

Tom Gilb  
Dorothy Graham



# What are good test requirements

**and what to do if you don't get them?**

**Presenter Tom Gilb**

♦ — ♦  
**MASTER 2016**



# 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**



# 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**

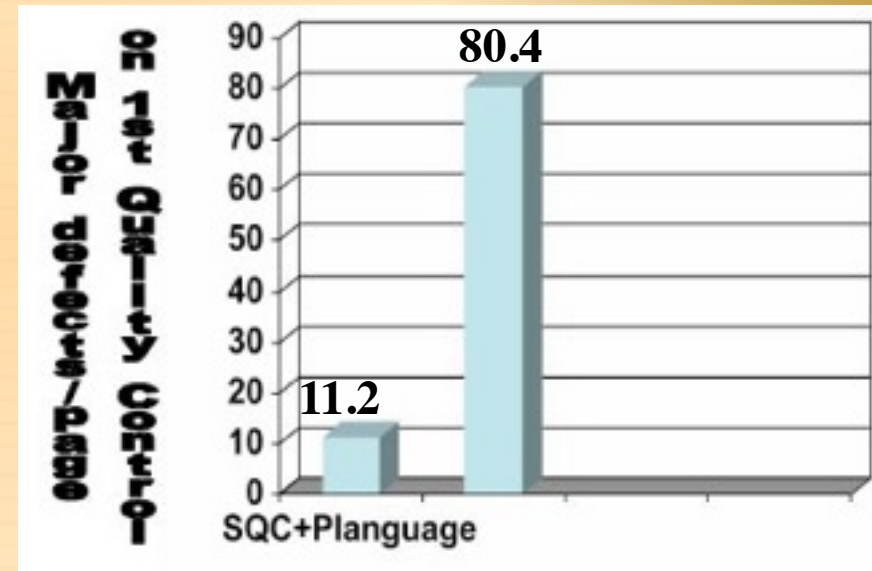
**Across 18 DV (DeVelopment) Projects using the new requirements method, the average major defect rate on first inspection is 11.2 per logical page.**

**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.**



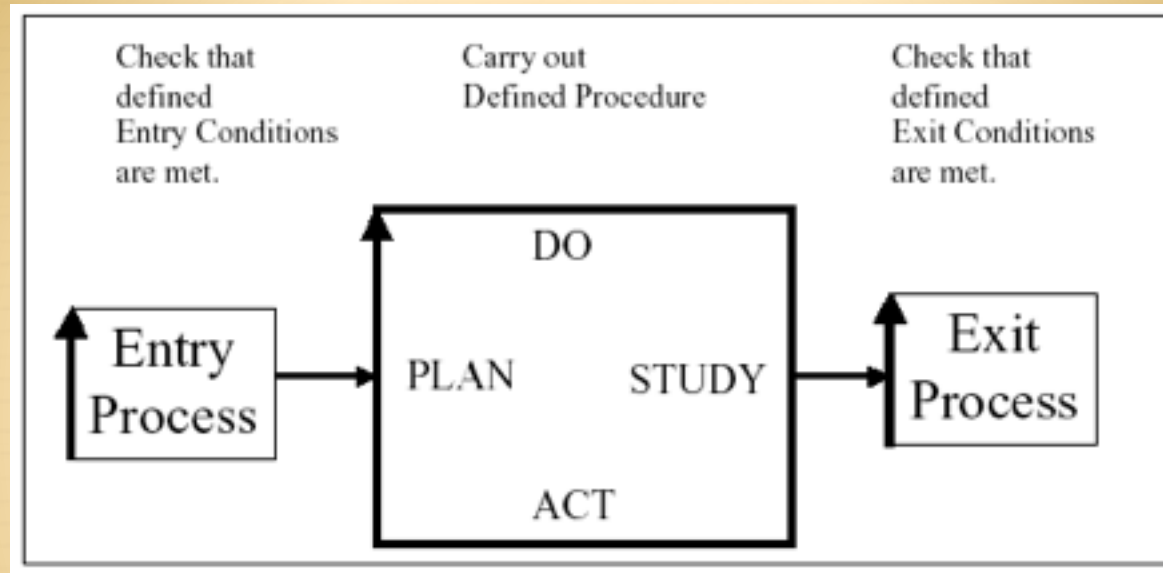
# Principle 2.



✦ **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!**



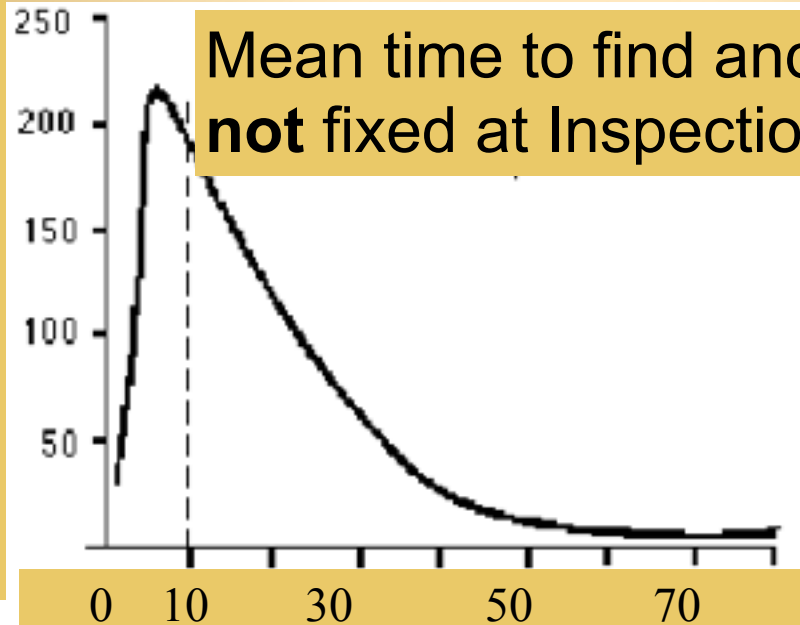
# The downstream alternative cost of quality at a UK Defence Electronics Factory.

**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



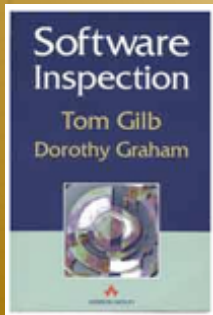
Mean time to find and correct a Major if  
**not** fixed at Inspection was **9.3** Hours.

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

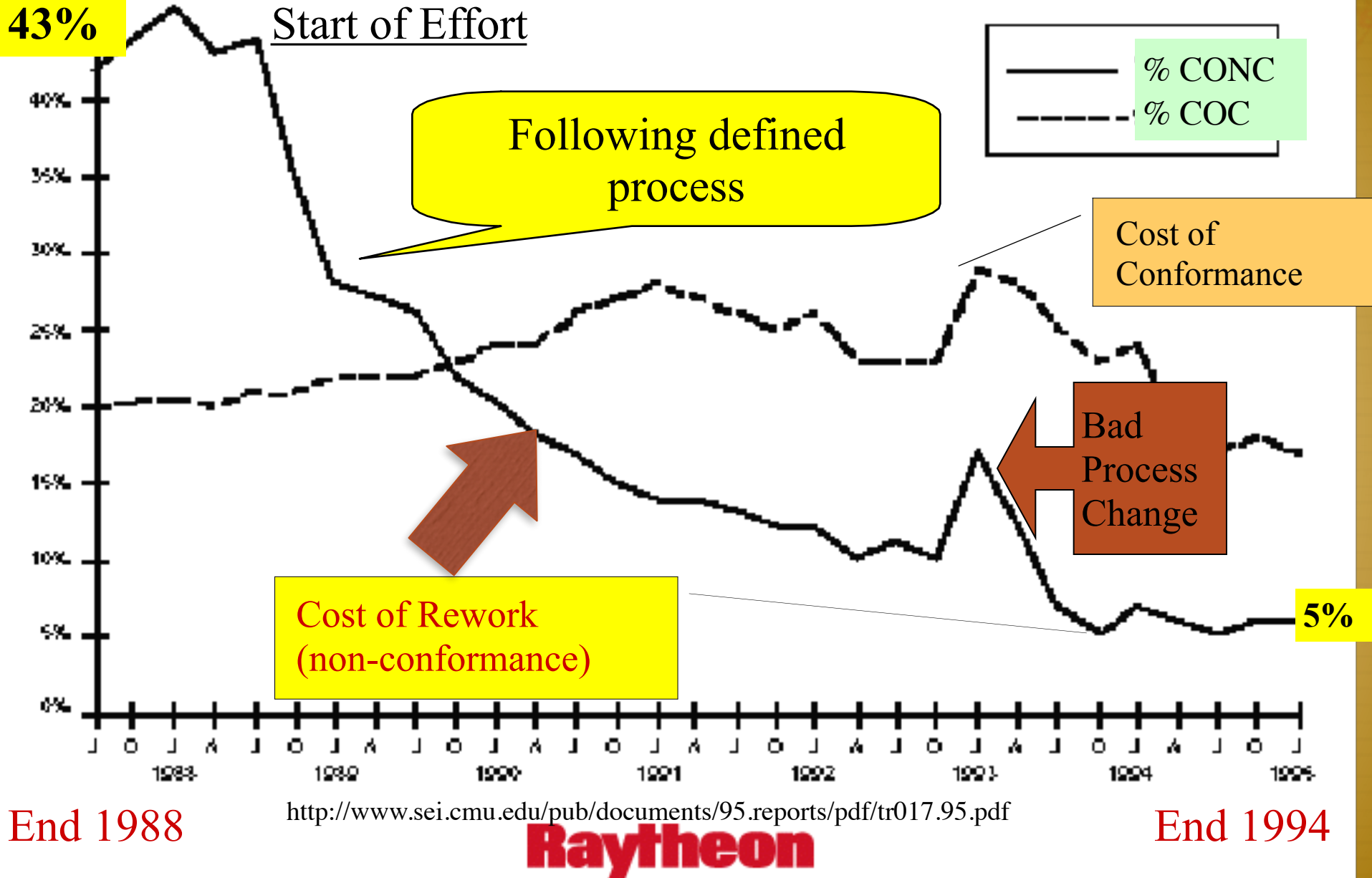


Figure 8: Cost of Quality Versus Time

# **‘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 it 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:



## Rock Solid Robustness:

**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.

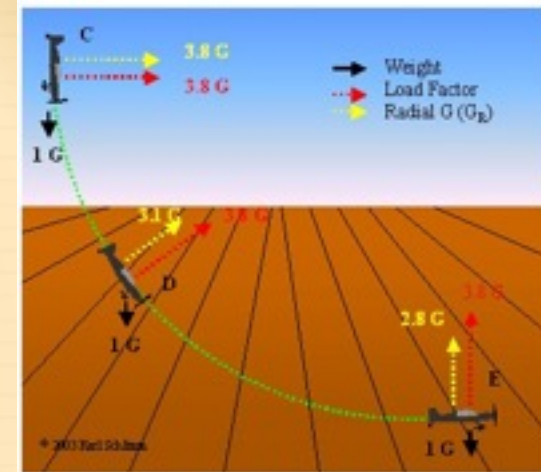
**Scale:** Duration from Initiation of Restore to Complete and verified state of a defined [Previous: Default = Immediately Previous]] saved state.

**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.





## Testability:

**Type:** Software Quality Requirement.

**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.

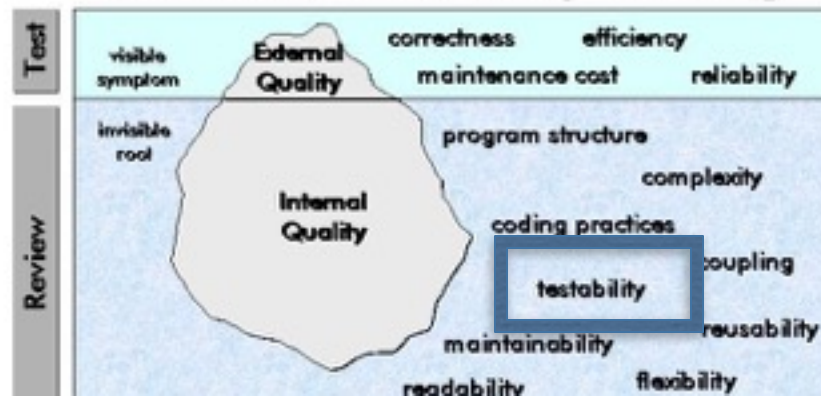
**Scale: the duration of a defined [Volume] of testing, or a defined [Type], by a defined [Skill Level] of system operator, under defined [Operating Conditions].**

**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

# Defining a Third Component Clearly:

## The Software Quality Iceberg



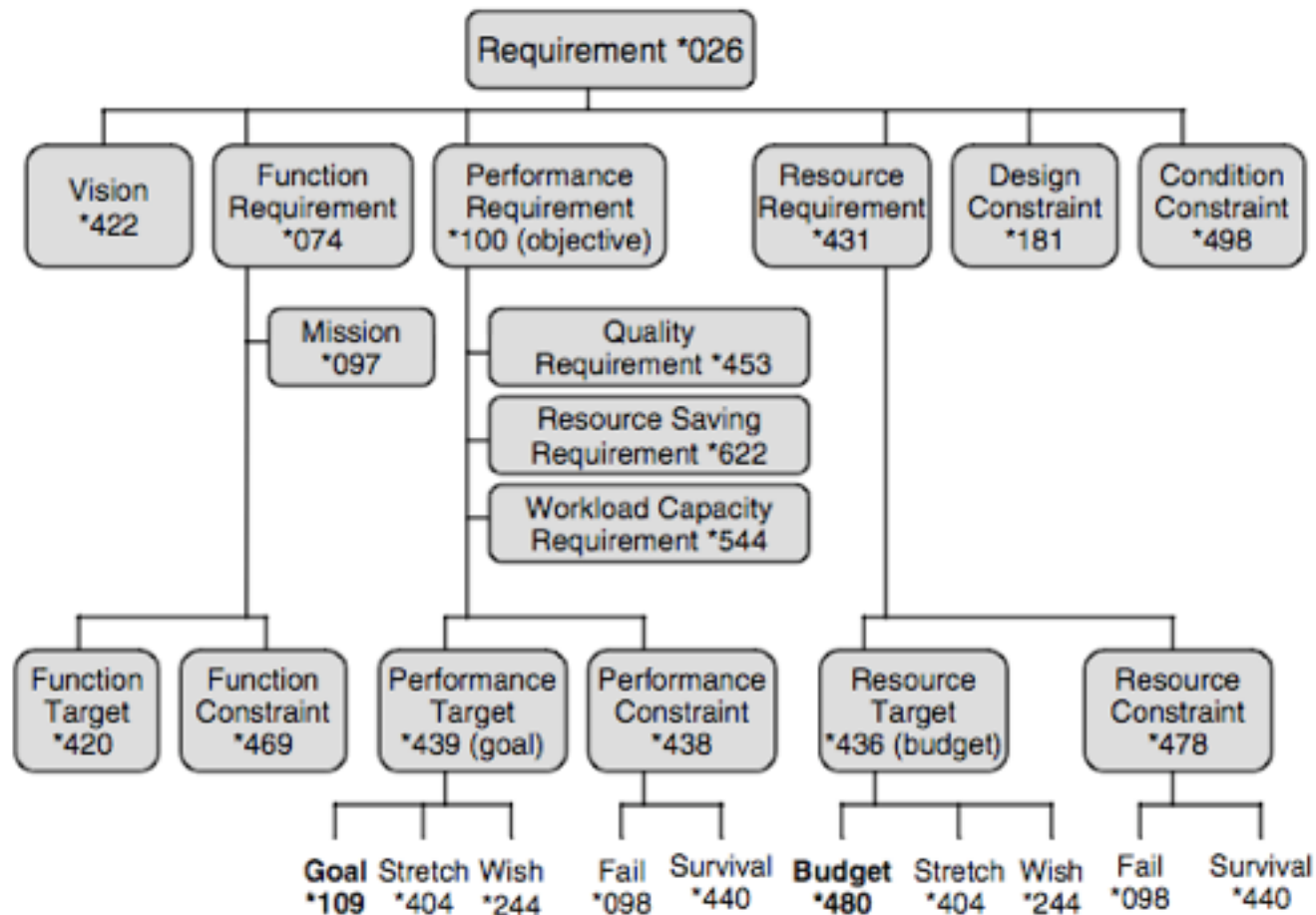
# **Principle 4.**



- ✦ **Requirements must be correctly “Typed”**
- ✦ **(Function, Quality, Constraint etc.)**

# Requirement Types

Planguage Concept Glossary 401



**Figure G20**  
Requirement Concepts.

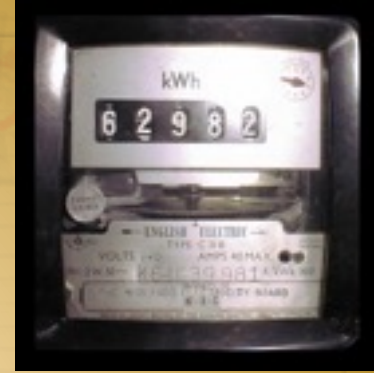
# Type determines Test



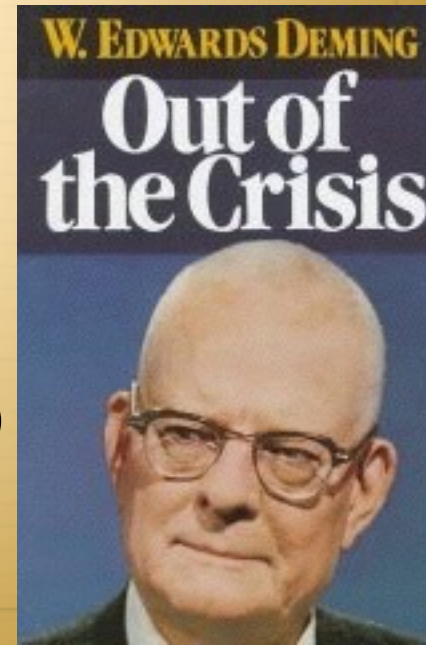
- ✦ **Type: Function Requirement.**
  - ✦ **Test for presence**
- ✦ **Type: Quality or Performance Requirement**
  - ✦ **Test along the scale using a defined 'Meter' (test process)**
- ✦ **Test: Constraint**
  - ✦ **Test to see if the defined constraint (example Legal) is violated or not**



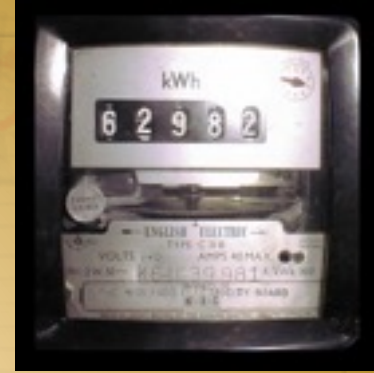
# Meter Concept \*093



- ✦ **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



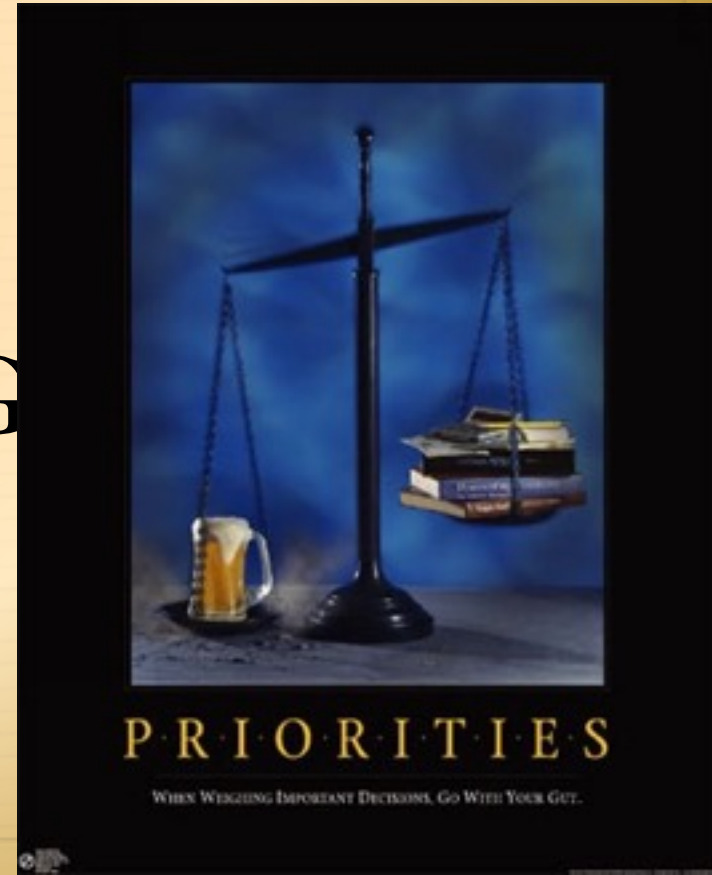
- ✦ **Satisfaction:**
- ✦ **Scale: Percentage of <satisfied> Customers.**
  - ✦ **Meter [New Product, After Launch]: On-site survey after 30 days use for all Customers.**
- ✦ **Past [This Year, USA]: 30%.**
  - ✦ **Meter [Past]: Sample of 306 out of 1,000+ Customers.**
- ✦ **Record [Last Year, Europe]: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

# Principle 5.

---

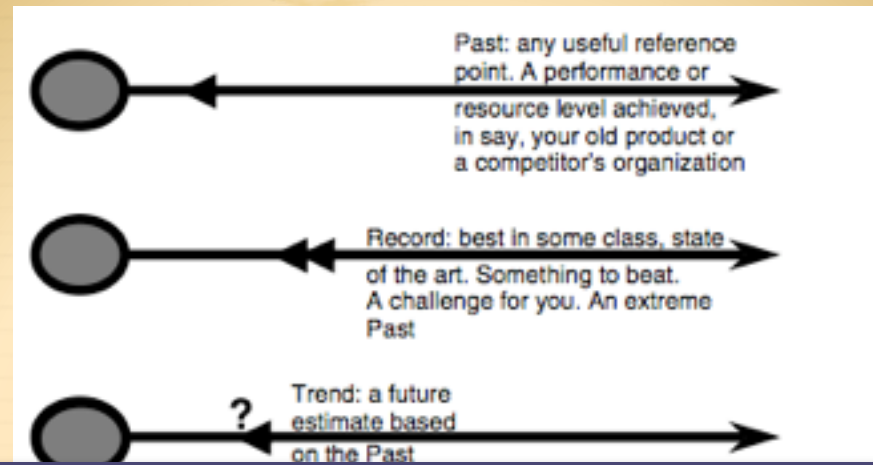
✦ The requirement level *priorities* must be specified

✦ (Survival, Fail, G  
Stretch)

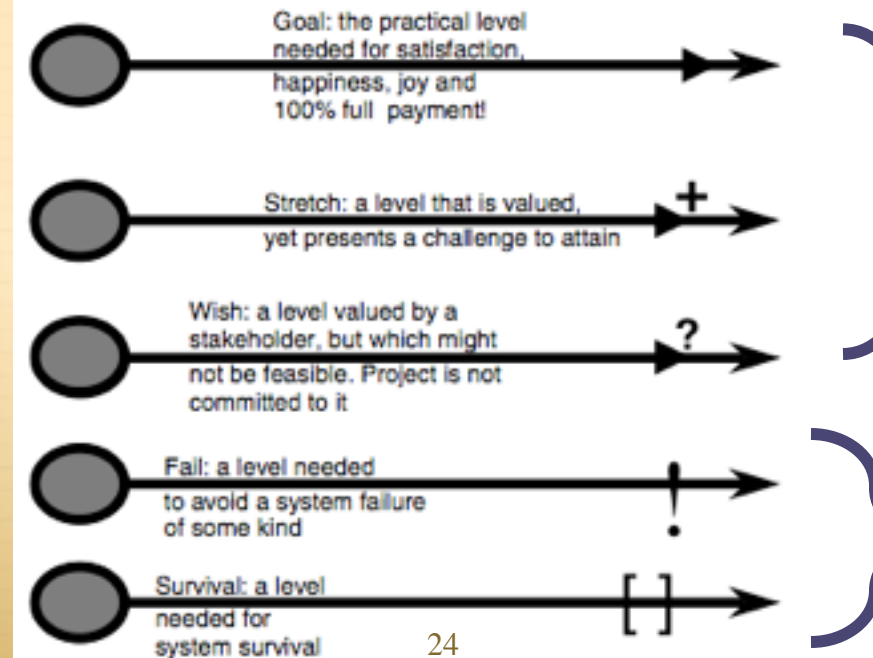




# Levels of System Performance



**‘Benchmark Levels’**



**Target levels**

**‘Requirement Levels’**

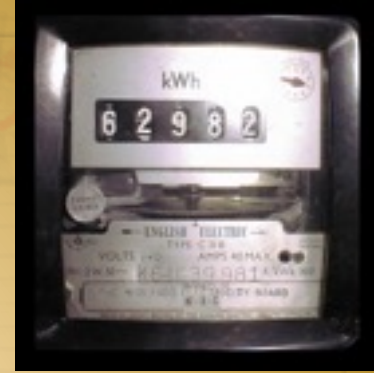
**Constraint levels**

## 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



- ✦ **Task Productivity:**
- ✦ **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.**

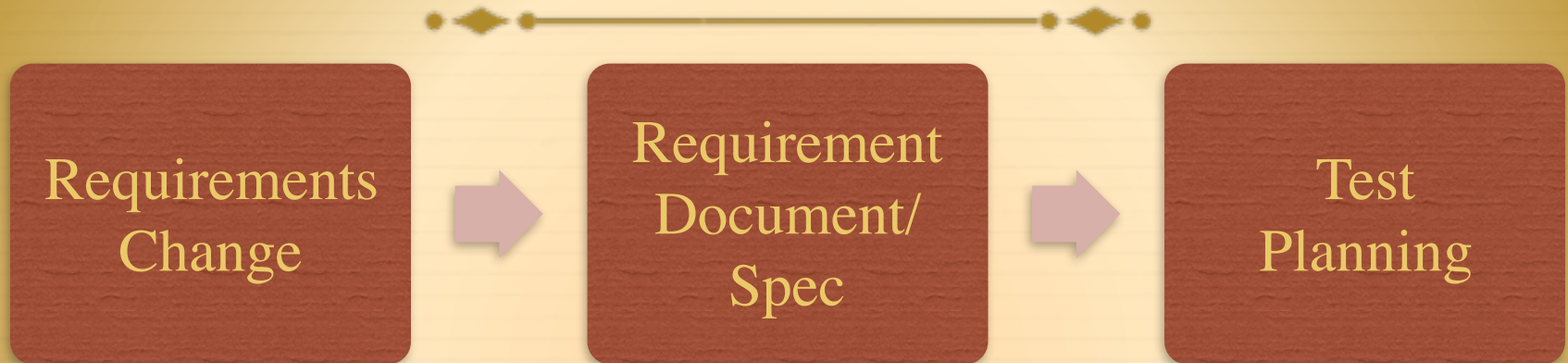


## **✦ Version Control**

**✦ for each requirement change must be fed to responsible test planners**

**✦ (automatically, with note it is done)**

## 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 2012: 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 2012: 16:00**

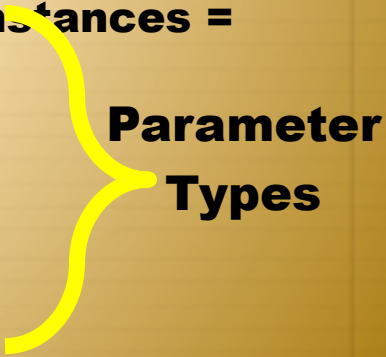
# **Principle 9.**



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



# **[Conditions] – 3 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}.**
- 
- Parameter Types**

# 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:

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**

# That's All Folks !



- ✦ [www.Gilb.com](http://www.Gilb.com)
- ✦ For more information.
- ✦ If you email me at Tom at Gilb .com
- ✦ Subject . 'book'
- ✦ I'll send digitally you the CE Book (60% on requirements) , the Evo book and links to papers on requirements, estimation and other subjects.

