




Let's make Quality  
measurable and maintainable

## Quick context for this preso

---

- ▶ 1990's – TQM, CMM for profitable, as agreed delivery
- ▶ Today's scene: quality confusion, time, cost slippage
- ▶ Purpose today:
  - Re-ignite some quality thinking
  - Interaction please: robust debate   

# Introducing quality thinking

---

- ▶ “It’s bad, the number of faults are very high, and immediately after release there were four patches”
  - CTO small product development company
- ▶ “They’re always late, they over-engineer and the quality is poor”
  - Business manager, commenting on their IT department
- ▶ “Our first job is to assemble as many requirements as we can, but we never have the time to test them all”
  - Common issue for most Testers we encounter

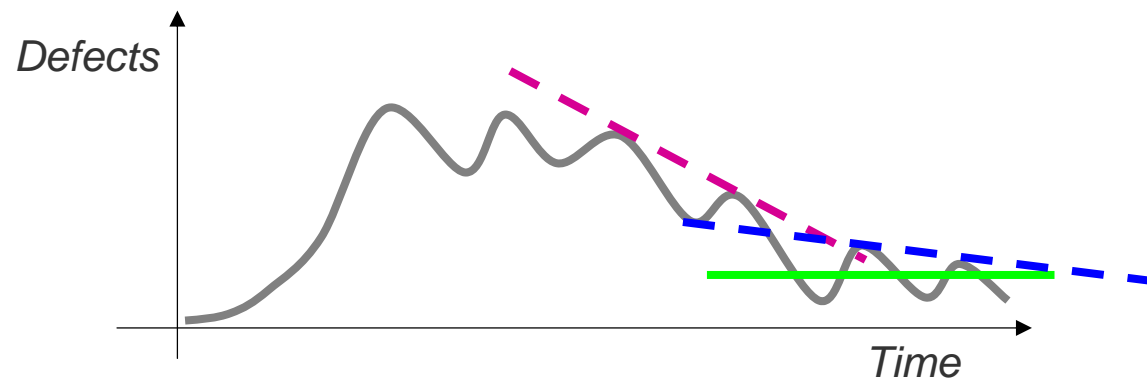
## And what to think of this ...

---

- ▶ “It’s not a bug, it’s working” (yes but not as required)
  - Desperate manager, IT vendor
- ▶ “When we enhance / maintain we have no real idea what to retest”
  - Quality manager, on core business system
- ▶ “Stop writing more requirements, we know the requirements, now let’s get on with it”
  - Same CTO, product development company

## How is this for quality thinking ...

- ▶ “It seems projects veer off their targets, I would measurably like to know if we’re meeting our business requirements”
  - CIO of a large SOE, 2008
- ▶ “We accept the system when it has met this threshold of requirements, and the defect curve flattens out”
  - CMMI / quality auditor, Dutch KPN, 1998



# Outline

---

- ▶ What is quality
- ▶ How to make quality measurable
- ▶ Traceability, the key to change & maintainability
- ▶ Tools can help, but just make a start
- ▶ Full circle – let's test our Quality system
  
- ▶ Measurably deliver more, on-time, to quality

# 1

## What is quality

---

- ▶ BMW or Toyota Yaris?
- ▶ Lazy boy or Straight backed chair
- ▶ Pencil or \$200 Golden ballpoint pen?

# Quality - What do the standards say:

## ▶ PMBOK:

- Features & defects levels (😊)
- Satisfy stated & implied (😞) needs
- “Benchmarking”, “Flowcharting”, “Normal process variation”

**But:** 🚧

- unique one-off; no production-line (a house, **not a car!**)
- Implied = mis-communication = scope creep to the max

## ▶ Quality management systems

- “project quality = meeting 100% of the requirements”

**But:**

- How about bugs?!
- What if I specify a **concrete life-jacket?!**

# Quality = Perception ?

- ▶ “Quality” for many is equivalent to:
    - “Perfection”, “smooth”, “soft”, “beautiful”
    - “Low defects”
- Personal need (scope)*
- “Build” tolerance (scope + defects)*
- Doesn’t leak; fall off hinges (bugs)*
- ▶ Quality = scope and communication
    - Features (requirements)
    - Defects levels (requirements)
    - Sometimes very hard to specify
      - know it when you see it ?
      - “Soft & Beautiful” ?

Use “equivalents”, or time / cost “budgets” (requirements)

# Requirements revisited

---

- ▶ A “requirement” is a condition or capability to which the product or service must conform.
  - the list of things the software must provide / do ...
- ▶ Good requirements:
  - Have a unique identifier! BR\_001; SR\_142; OR\_015 ...
  - Have attributes:
    - Priority: Must, Should, Would, Nice –to have
    - Stability: Certain, Flux, Volatile
    - State: Raised, approved etc
  - Have a description which is:
    - Verifiable / Measurable, **Atomic**, Unambiguous,
    - Concise, Complete, Consistent

# Quality can be specified

## Features

- SR\_3 The system shall provide a Forum
- SR\_25 The Forum allows the user to create sticky notes
- SR\_31 The Forum can be configured to do ...
- SR\_29 The Forum blocks posting of unwanted vocabulary

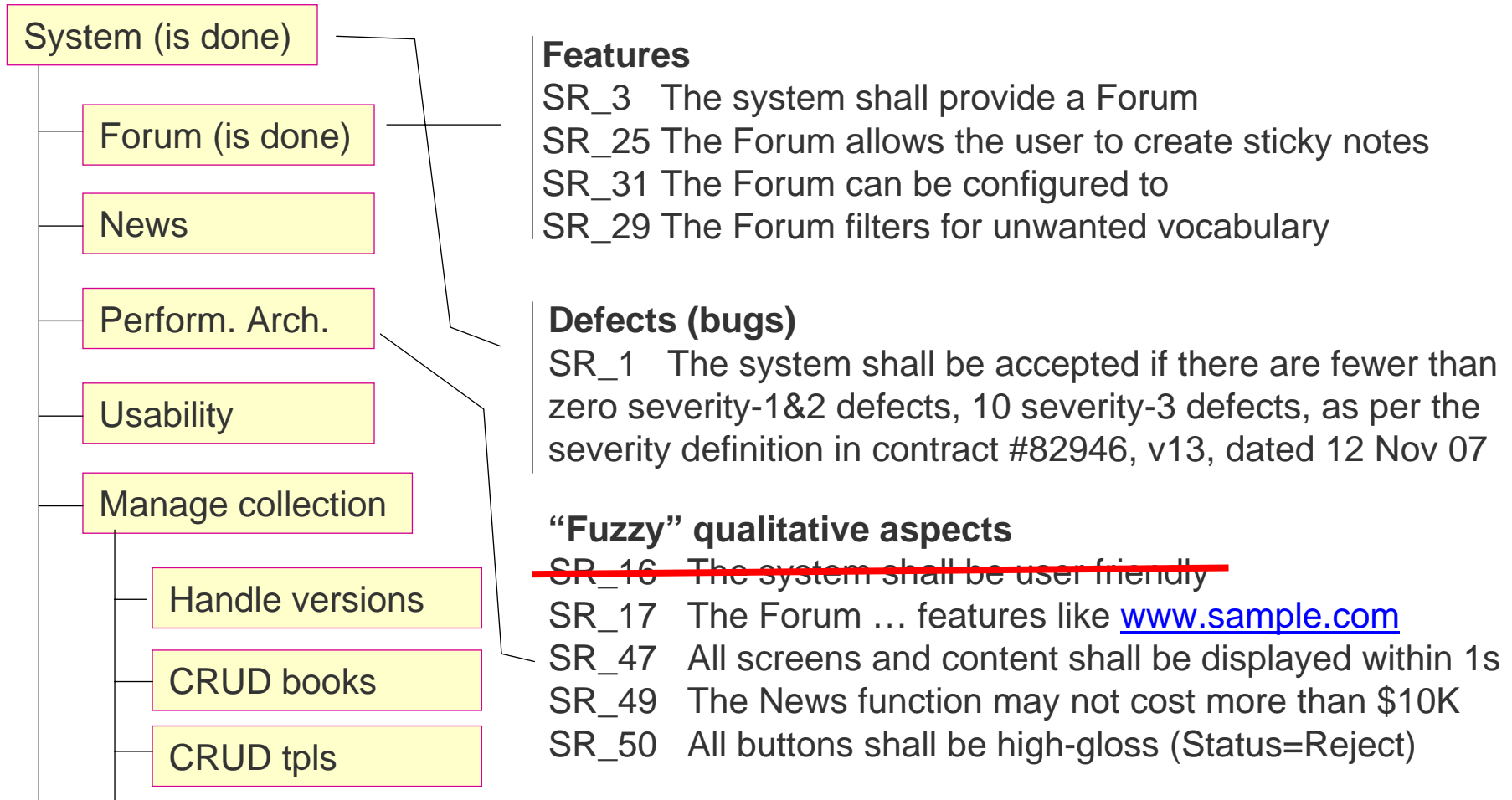
## Defects (bugs)

- SR\_1 The system shall be accepted if there are fewer than zero severity-1&2 defects, 10 severity-3 defects, as per the severity definition in contract #82946, v13, dated 12 Nov 07

## “Fuzzy” qualitative aspects

- ~~SR\_16 The system shall be user friendly~~
- SR\_17 The Forum ... features like [www.sample.com](http://www.sample.com)
- SR\_47 All screens and content shall be displayed within 1s
- SR\_49 The News function may not cost more than \$10K
- SR\_50 All buttons shall be high-gloss (Status=Reject)

# Quality is a statement of scope



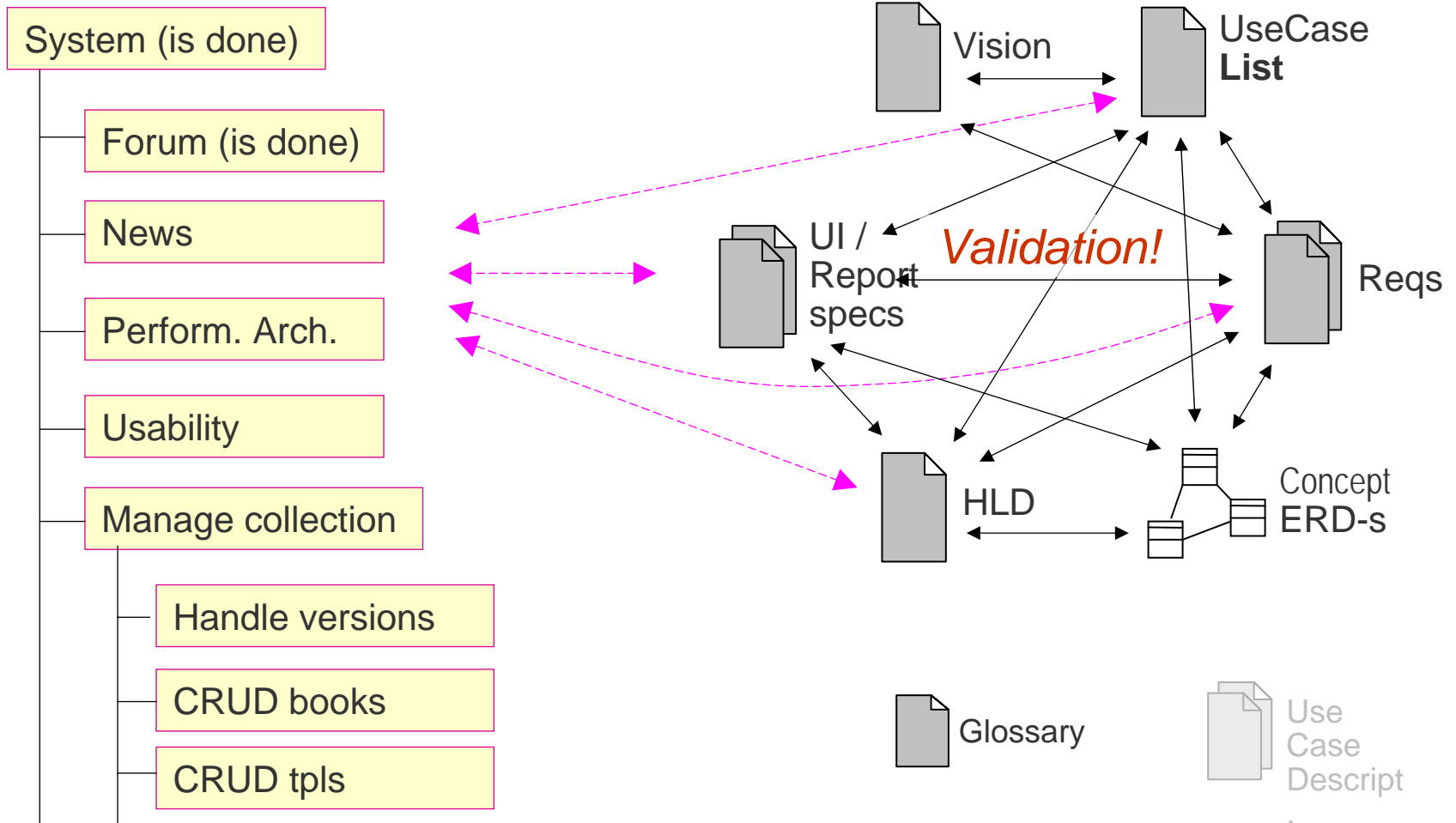
WBS = A deliverable oriented breakdown of work, anything on the WBS is in scope, anything not on it is out of scope. A WBS is **NOT** a task list

# Quality needs to be communicated

---

- ▶ Communicate the list of things it must comply with to see if the things are the right things
  - With whom 🧨
  - Validated to see the specified product is right
- ▶ To be prioritized and agreed

# Validate your Quality concrete life-jacket



WBS = A deliverable oriented breakdown of work, anything on the WBS is in scope, anything not on it is out of scope. A WBS is **NOT** a task list

# Key to success – avoid over specifying

## ▶ Fundamental

If it is important to test have a requirement

## ▶ Avoid over specifying

- is it really important to specify all the buttons?
- Or all the data?
- Or having requirements for all aspects of the UI standard?
- Or creating precise screen layout mock-ups
- Or endless words of how the **Case of Use** has to unfold?
- Or allocating all enterprise architecture requirements even if
- Or ...



# Quality = meeting the agreed requirements

## ► Essential “tricks”:

- If it is important to test have a requirement
- SR\_1 [M] The system shall be accepted if there are zero severity-1&2 defects, no more than 10 severity-3 defects, as per the severity definition in ...
  - 1 – system crashes
  - 2 – system works but can't access functions / data
  - 3 – does not meet “must have” or “should have” requirements
  - 4 & 5
- SR\_2 [M] Code can be extracted from ConfigMgt, compiles and yields the identical binaries as the test version
- SR\_3 [M] 5% of code has been reviewed & passed std xyz
- SR\_4 [M] prior to commencing construction, or testing the conceptual (logical) ERD has passed review.

Why [M] 

# Quality = meeting the agreed requirements

## ▶ More essential “tricks”:

- SR\_# [M] shall deliver the data and functions shown in fig#
- SR\_# [M] layout & colours shall comply with standard xyz
- Performance requirements always as measured on the scr.
  - Don’t be fooled into ...
- Have “layered” requirements (next)

## ▶ And communicate to:

- Validate 🍷
- Test if it is needed (how 🍷 )
- Prioritise (Must, Would, Should, Nice)



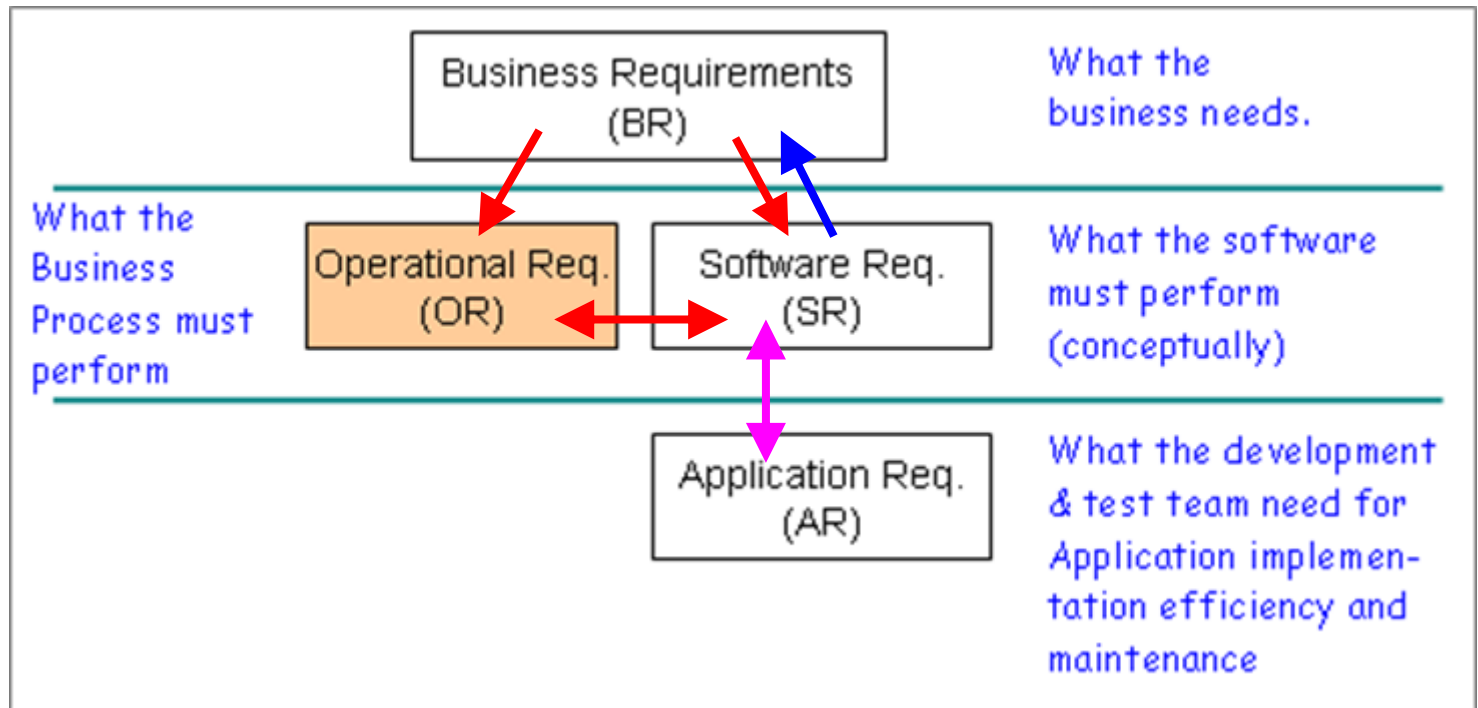
# 2

## How to make quality measurable

---

- ▶ Introducing “layers”
- ▶ Reporting quality & scope in measurable terms

# Layers - essential to time, cost & progress



What "layer" is missing





# Report scope & quality to exec mgmt

## Business requirements

Showstoppers:

Must have daily accurate info No

Credit notes measurable Yes

Must have requirements, met: 87%

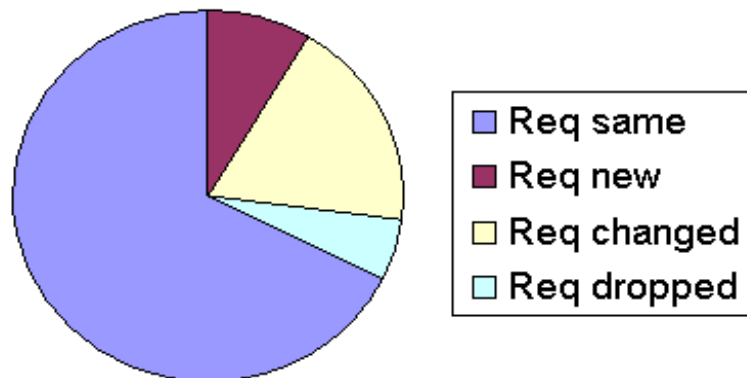
Total requirements met 82%

## Must have requirements, **not met**

BR\_23 customer satisfaction information

BR\_31 verified delivery of goods

## Change rate since business case acceptance



# Report scope & quality software detail

Enter date: 

Test vCur	Test vPrev
<input type="text"/>	

Requirements Verified / met	v	0	0
Requirements Not met	x	0	0
Don't know if met, as we can't test	q	0	0
Under action, not yet complete	a	0	0
To be discussed			

Pending Change request  
On hold - to be done after

Rejected requirement  
Waived requirement

Total  
Check count of ItemDb

Total tested  
Total open  
Percentage not met

		Smartmatix		
		v0.4q	v0.4p	v0.4n
Requirements Met	v	122	112	108
Requirements Not Met	x	2	7	10
Met ? (don't know, not able to test)	q	0	1	2
In Progress / needing action	a	2	7	8
To be discussed	d	0	0	1
Pending CR's	c	2	4	3
On hold	h	6	8	8
Not in Scope (reject)	r	13	12	12
Not testable	n	1	1	1
Waived	w	14	11	10
<b>Total</b>		<b>162</b>	<b>163</b>	<b>163</b>

Total Open	Perc
126	4 3%

# 3

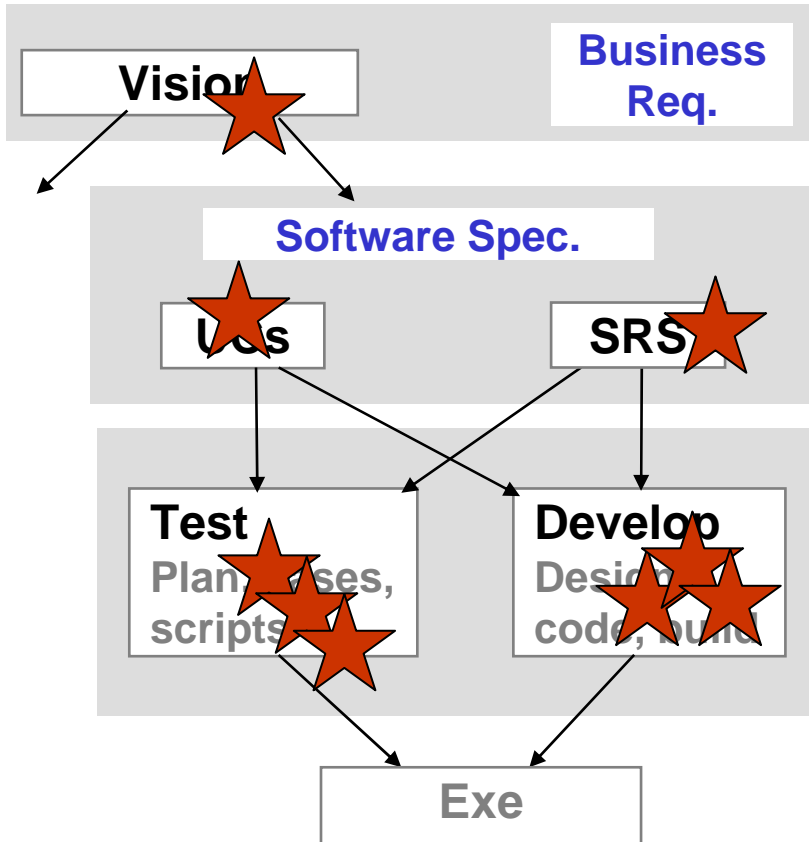
## Maintain quality in the face of change

---

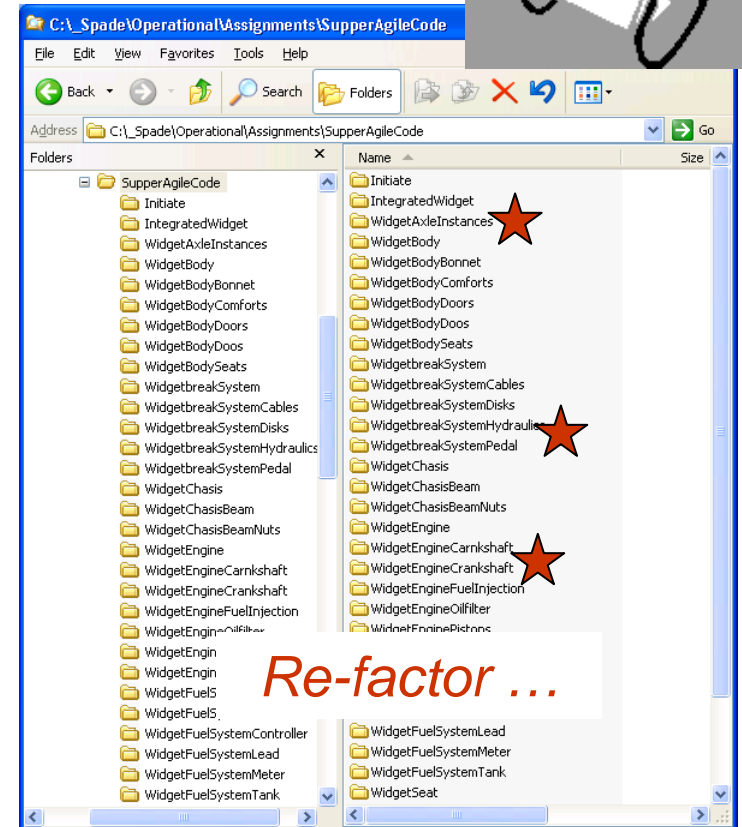
- ▶ What happens when things change
- ▶ bi-directional traceability -  
the key to maintainability
- ▶ Why & who
- ▶ Example using a tool

# What happens when things change?

“Planned”



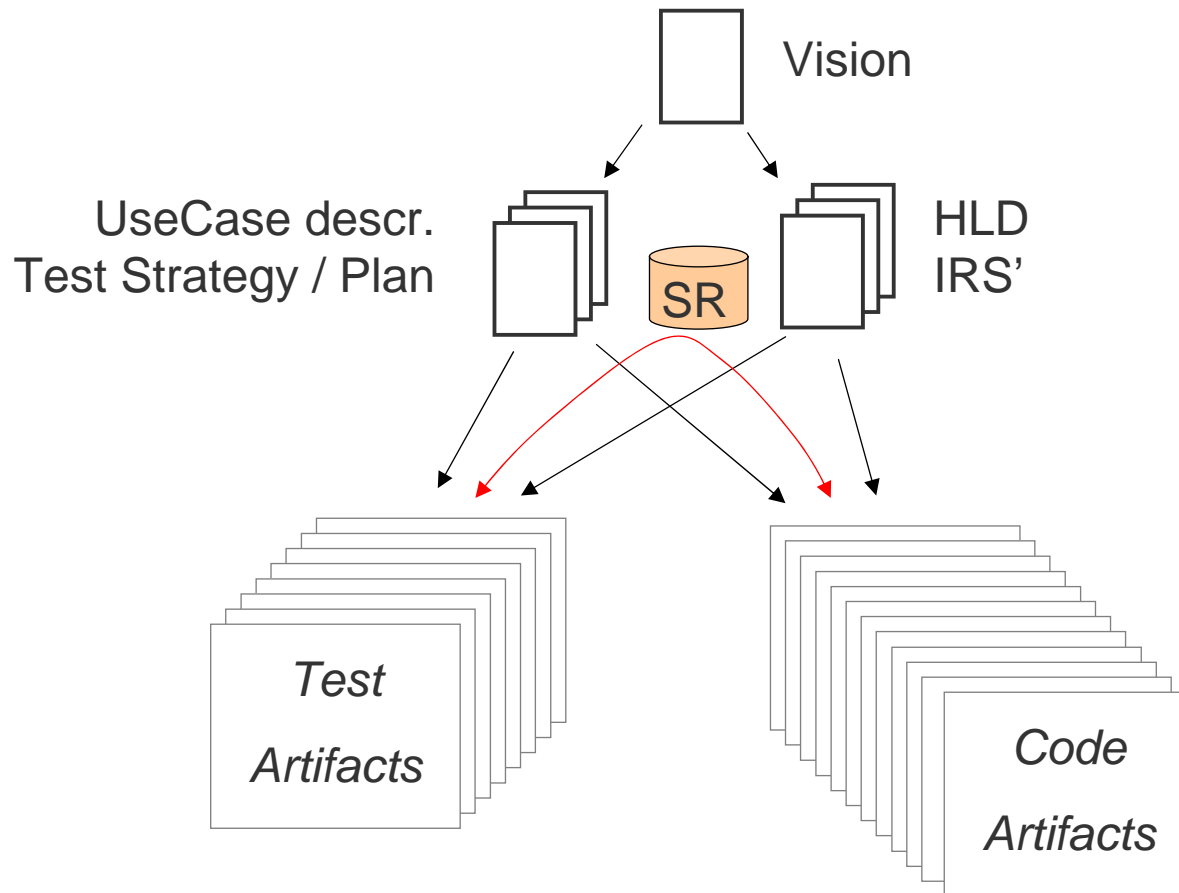
“Agile”



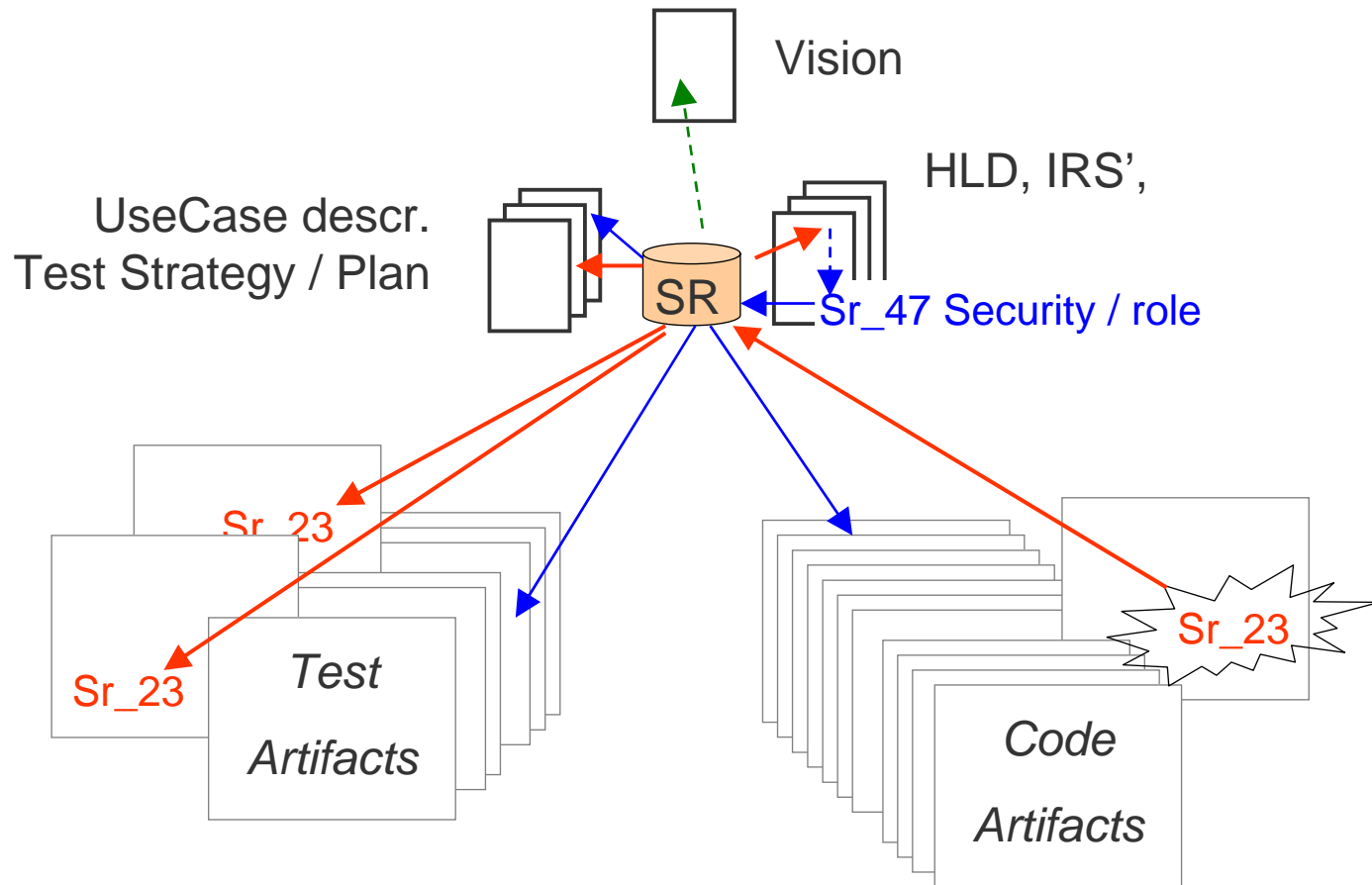
... before long no quality!

# Enable change – have a tree of small artefacts

- ▶ “Decompose” and “explode”, & don’t drown in detail ...

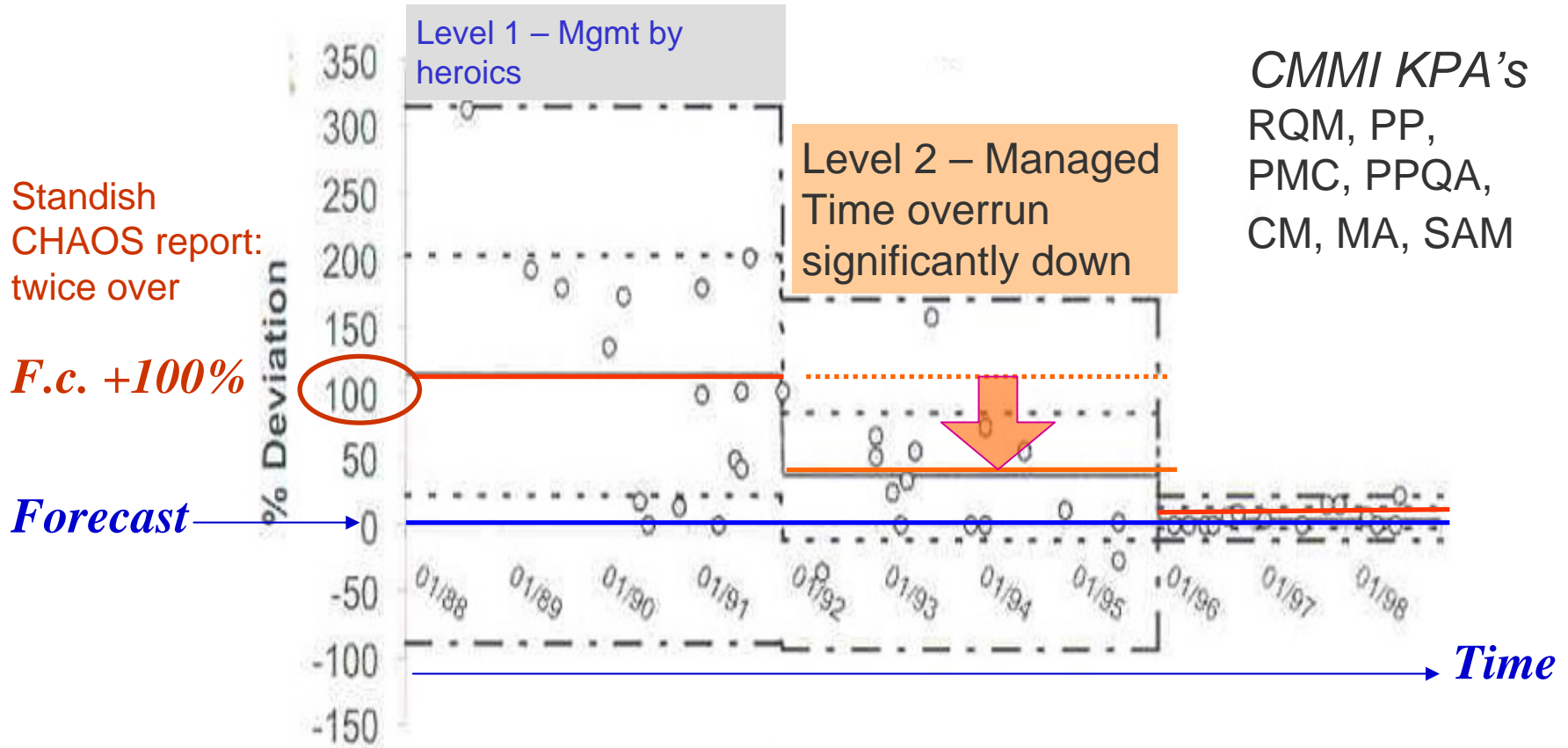


# Know what changes, bi-directional traceability



*From work product to req and back*

# Who says bi-directional traceability is important



## Schedule Deviation

Source: CMM (CMU-SEI)

# CMMI level-2 : bi-directional traceability

## ▶ BA's, Testers, Developers

- 2 RQM SP 1.1 **Obtain an Understanding of Req'ts** with the requirements providers on the meaning of the requirements.
- 2 RQM SP 1.3 **Manage Requirements Changes** as they evolve during the project
- 2 RQM SP 1.4 **Maintain Bi-directional Traceability of Requirements** between requirements, project plans and work products
- 2 RQM SP 1.5 **Identify Inconsistencies** between Project Work & Requirements i.e. between the project plans, work products and the requirements



## ▶ Project manager

- 2 GP 2.4 **Assign Responsibility** and authority for performing the process, developing the work products, and providing the services of the **RQM** process

# 4

## Tools can help, but just make a start

Of course your boss is going to pay  
\$20K tools, \$10K training,  
\$20K consulting  
per user  
for each in the team of 60



# Start recording & “tagging”

## **UC\_01 Case of Use A**

... blah dee blah text  
and so on describing  
an aspect of  
importance {SR\_25}.  
The sentences  
continue.  
Blah more something  
that needs to be  
tested. {SR\_12}  
{SR\_16}

## **HLD**

-----  
-----{SR\_47}. -----  
-----  
---.  
Next heading  
-----  
-----.  
-----  
----- [SR\_12]  
[SR\_16]

## **Req list**

BR\_1 ..reduce..  
BR\_2 ..measure  
BR\_3 ..  
  
SR\_12 ..shall..  
SR\_25 ..shall..  
SR\_16 ..shall..  
SR\_47 ..shall..  
...

*RqDb\_3.0.  
xls*

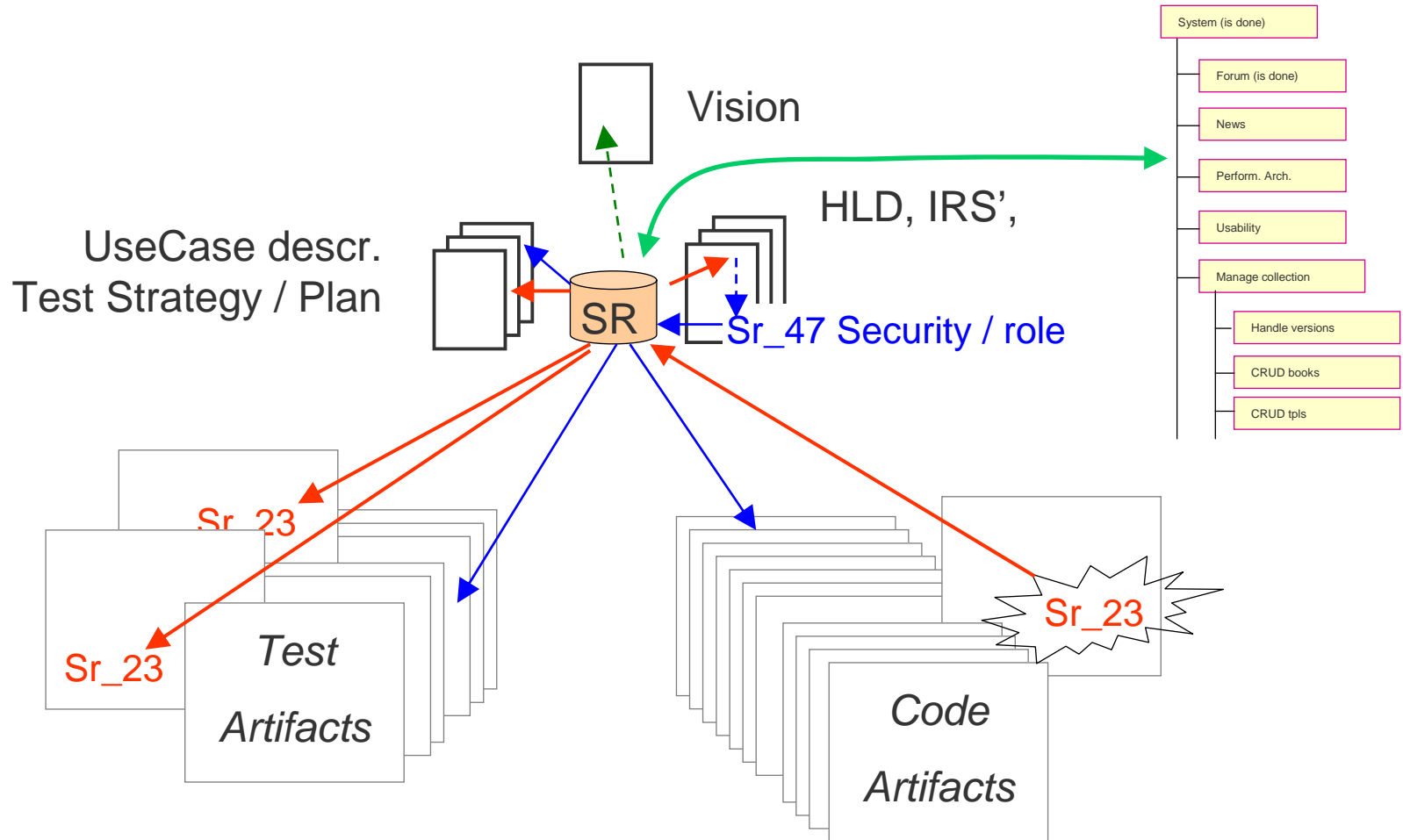
## **Test T1**

Enter abc  
Verify pqr [SR\_12]  
Enter de  
Navigate  
Enter f  
Verify tqm [SR\_13]

## **Code M1**

Try ...  
Error  
Else  
End [SR\_12]

# Now you can do Bi-directional traceability



## Most importantly - *make a start*

---

- ▶ Start a requirement management list
  - (if you haven't got one, use [our](#) RqDb spreadsheet, which generates numbers, updates req. versions and logs the user name who made changes)
- ▶ Specify those key requirements!!
- ▶ If you have time, start tagging the most essential relationships
- ▶ When ready, step up to bi-directional traceability for maintainable on-time, quality delivery
  - Consider a tool like [our](#) ScopeTracker, ReqPro or similar

# Benefits

---

- ▶ Deliver on-time, to quality, *as agreed*
- ▶ Manage change impacts & maintain quality over time
- ▶ Acceptance by measure,  
and know upfront when done!
- ▶ Report to management in measurable terms
- ▶ Initially you may not gain much, *but v2, and v3 !!*
- ▶ + You'll get ready for a tool to manage all this for the next step in efficiency! = bottom line!!

# 5

## Full circle – let's test our Quality system

---

- ▶ Can we answer “their” quality thinking?

# Can our system address these?

- ▶ “It’s not a bug, it’s working”
  - Desperate manager, IT vendor



- ▶ “Our first job is to assemble as many requirements as we can, but we never have the time to test them all”
  - Common issue for most Testers

- ▶ BA’s are tagging to help testers; in process help the business & developers understand more clearly

- ▶ “Stop writing more requirements, we know the requirements, now let’s get on with it”
  - CTO, product development company



# Can our system meet these?

- ▶ “When we enhance / maintain we have no real idea what to retest”
  - Quality manager, on core biz system
- ▶ “They’re always late, they over-engineer and the quality is poor”
  - Business manager, commenting on IT dept
- ▶ “It’s bad, the number of faults are very high, and immediately after release there were four patches”
  - CTO small product development company
- ▶ Improved but still limited if you have no tool
- ▶ Our system cannot answer this, **why & what is the A?**



Bell 1: Prioritisation means you can deliver within time cost quality constraints; Thus metrics tell the true status of over-engineering

Bell 2: Defects need to be put in context of size; Requirements do not express a unit of size; the only system that can answer this is Function Points / defect rates

# Is our system on top of the game?

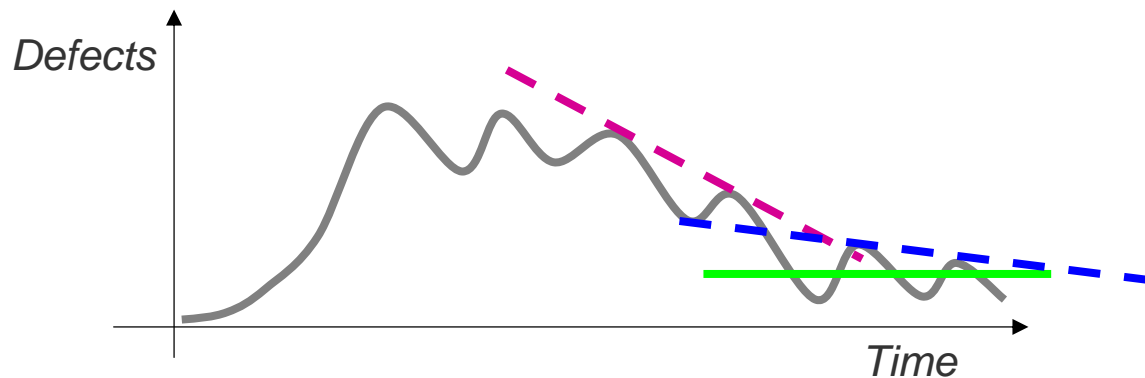
- ▶ “It seems projects veer off their targets, I would measurably like to know if we’re meeting our business requirements”

– CIO of a large SOE, 2008



- ▶ “We accept the system when it has met this threshold of requirements, and the defect curve flattens out”

– CMMI / quality auditor, Dutch KPN, 1998



## Business requirements

### Showstoppers:

Must have daily accurate info	No
Credit notes measurable	Yes
Must have requirements, met:	87%
Total requirements met	82%

### Must have requirements, **not met**

BR\_23 customer satisfaction information  
BR\_31 verified delivery of goods

Change rate since business case acceptance

Bell 1: A report like

Bell 2: Very easy to develop from a list of requirements and test history, and will look like the graph requested

- ▶ Let's make quality measurable
  - All quality aspects can be specified
  - If it is important to test have a requirement
  - Apply the Requirement tricks
  - BA, Test, Dev and PM must tag & understand req
  - A simple list and the start of tagging = on Time & Q
  
- ▶ and maintainable
  - Traceability & tools
  
- ▶ Thus to “behave” at CMMI level-2
  - **smarter** not harder
  - Reducing average time overruns from 200% to 120%
  - Measurably delivering more business value for less cost

# Question & discussion time

---

▶ Anything:

- Would you be able to apply this Quality “system”
- 

▶ Feedback please

▶ Further questions

- [jan@SmartMatix.com](mailto:jan@SmartMatix.com)
- [www.SmartMatix.com](http://www.SmartMatix.com)

SmartMatix - Productivity tools and know-how for IT, projects & programs