

The Importance of the Requirements Phase (in the Software Industry)

Shuly Cooper

Agenda

- Definition – Product Quality
- A Tool – Quality Function Deployment
- Cost of Quality – or lack of it
- The Requirements Phase – Definition, Attributes & Deliverables
- Additional Tools – Inspections and formal languages

Product Quality - Definition

1945-1980s

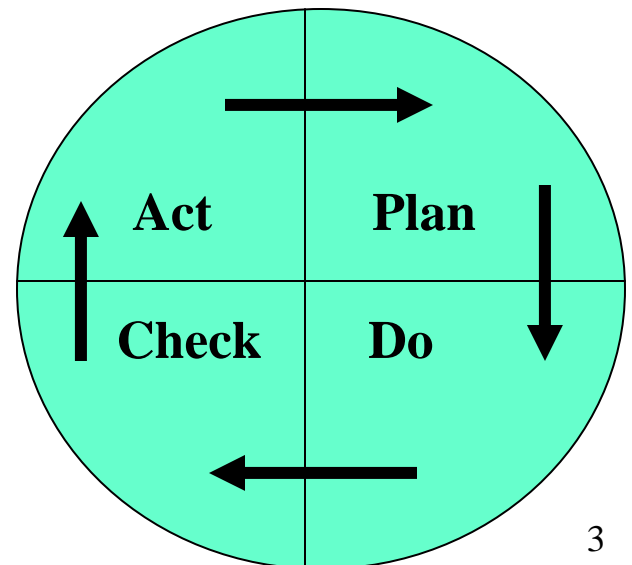
“Higher Accuracy & Reliability”

The Era of Statistical Process Control (SPC)

Dr. W. Edward Deming

Dr. Peter F. Drucker

The Deming Cycle:



Product Quality - Definition

1980s →

“Exceeding Customer expectations”

The tool: Quality Function Deployment (QFD)

The innovator: Prof. Yoji Akao -- U. of Tokyo

QFD deals with the total customer experience.

QFD Maps “What” to “How”

L=1
M=3
H=7

		I M P O R T A N C E	H O W						T O T A L	C O M P E T I T O R
			A	B	C	D	E	F		
M R K T. W H A T	1	H		H	L		M			
	2	M	L			H				
	3	L			M		H			
	4	H		L	H					
	5	M	H					M		
	6	L			M		L	H		
TOTAL										

L=1, "Bells & Whistles"
M=3
H=7, "A Show Stopper"

QFD for this Lecture

← HOW →

MRKT.	DEV.	I M P O R T A N C E	Research	Examples	Provide Data	Colors	Sections	Font size	T O T A L	C O M P E T I T O R
Promote "QA"		H	H	M	M				91	
Correct (no errors)		H	H	M	M				91	
Clear/Easy		M	L	H	H	H	H	H	108	←
Low cost		M							0	←
Long shelf life		M	M	L	L		M		24	
Fun/Aesthetic		M				H	L	L	39	
TOTAL			110	66	66	42	39	30	353	

↑ ↑ ↑

↑
W
H
A
T
↓

Market Requirements in the SW Industry

- Feature set – **Scope**
- Reliability (MTBF)
- **Cost**
- Safety
- Security
- Recoverability (zero loss)
- Scalability (tunable, modular)
- Performance (higher speed)
- Usability (easy to set-up, use, maintain)
- Friendliness (easy to learn)
- Supportability (service, 24x365)
- Release date/Availability – **Time**
- Aesthetics

Companies that are using QFD

IBM

AT&T

HP

Sony

Mitsubishi

Ford

Toyota

GE

Sybase – Clustering

Micro Focus – Year 2000 Solution

QFD - Benefits (1 of 2)

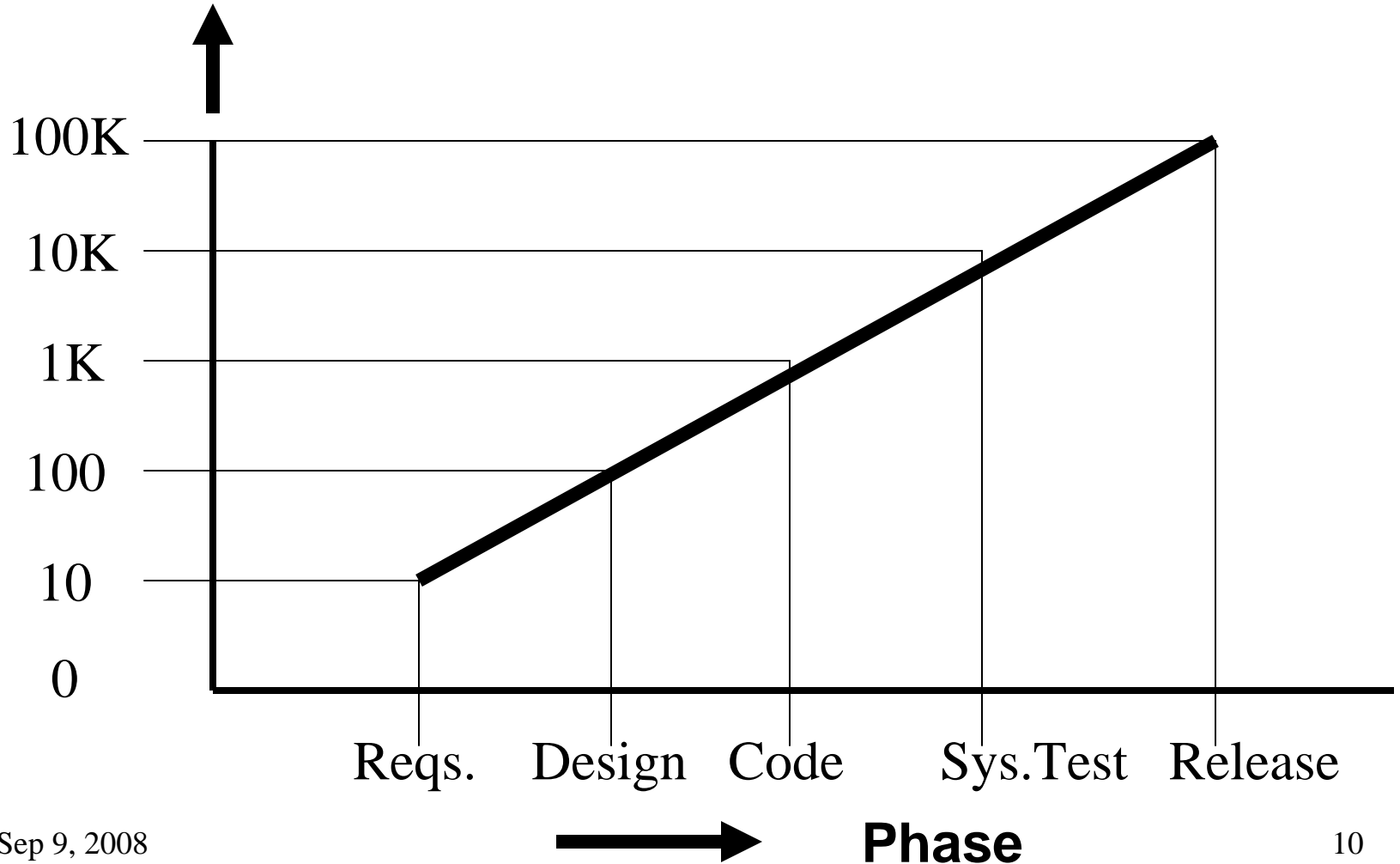
QFD is a cross functional, team process

- A communication tool
- Drives consensus across organizational boundaries
- Prevents “politics”, “lobbing” and “power struggle”
- **Stabilizes requirements**

Log (Cost to Find&Fix per Defect) (\$)

Cost of Quality

IBM -- late 70's



Cost to Fix a Defect

– During Requirements	\$195
– During Design	\$489
– During Coding	\$997
– During Testing	\$7,136
– During Deployment	Even more

Source: SEMATECH Technology Transfer #92111389A-TRG, March 5, 1993 ←

Source: Chuck Hoffman - Virtual Consulting Inc. - Spin 2001

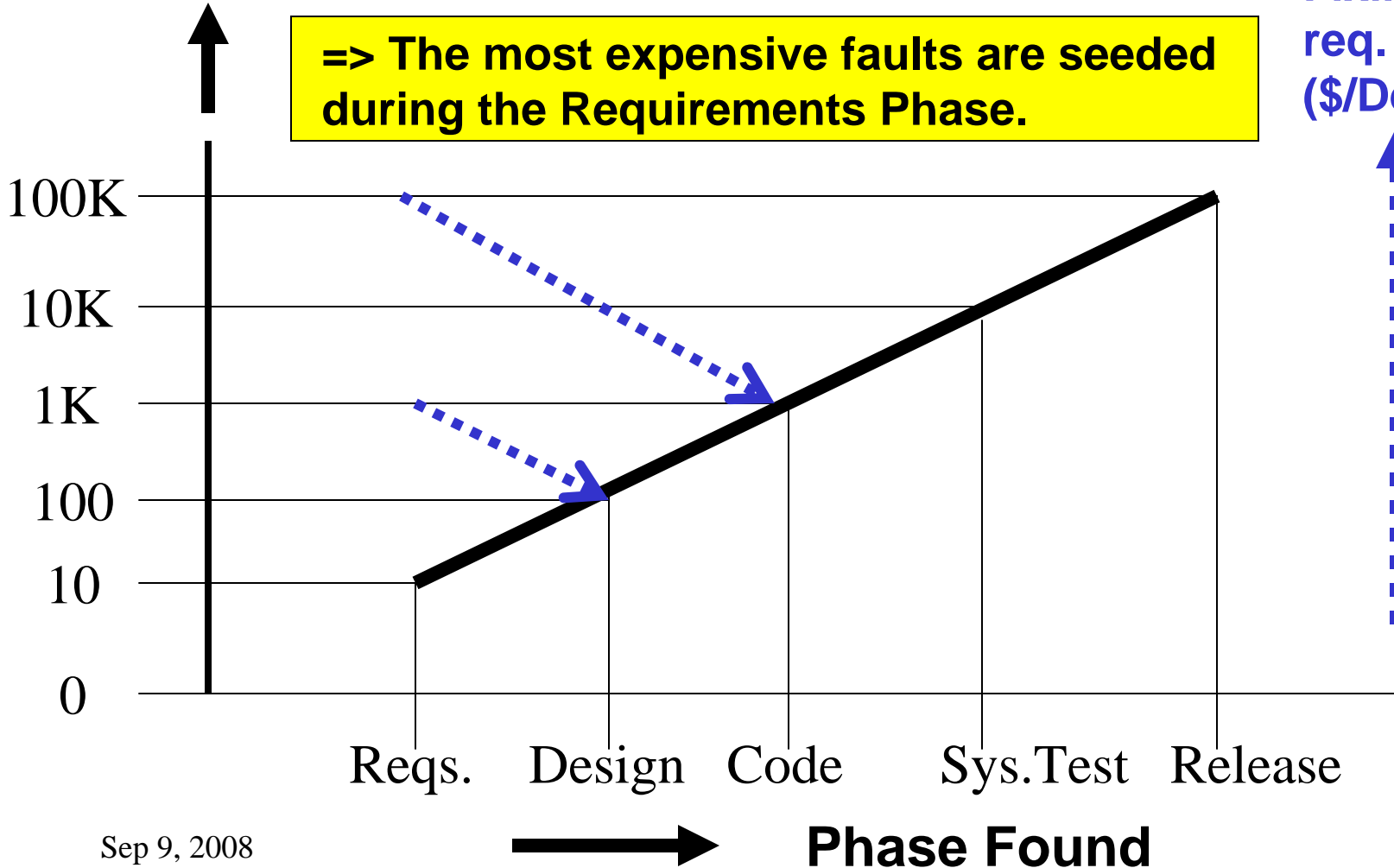
Cost of Quality

Log [Cost to Find&Fix per Defect] (\$)

Log [Cost of NOT Finding & Fixing a req. defect (\$/Defect)]

When faults migrate from phase to phase, they multiply 10-12 times

=> The most expensive faults are seeded during the Requirements Phase.

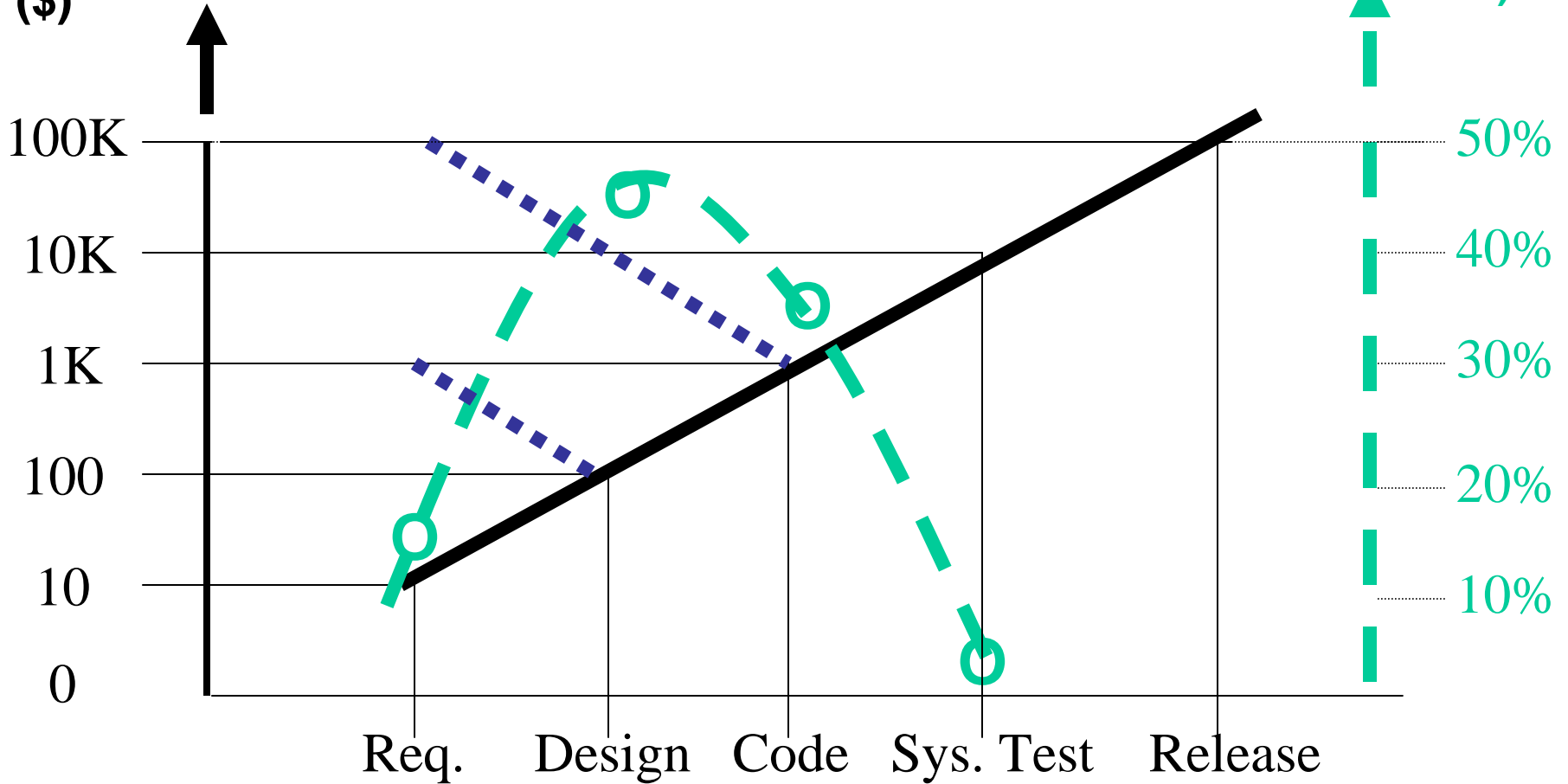


Cost of Quality

Root-Cause Analysis

Log Cost to Find & Fix per Defect (\$)

% of Seeded Defects (@ Bell Labs, late 1980s)



Defects Classified by Time of Introduction

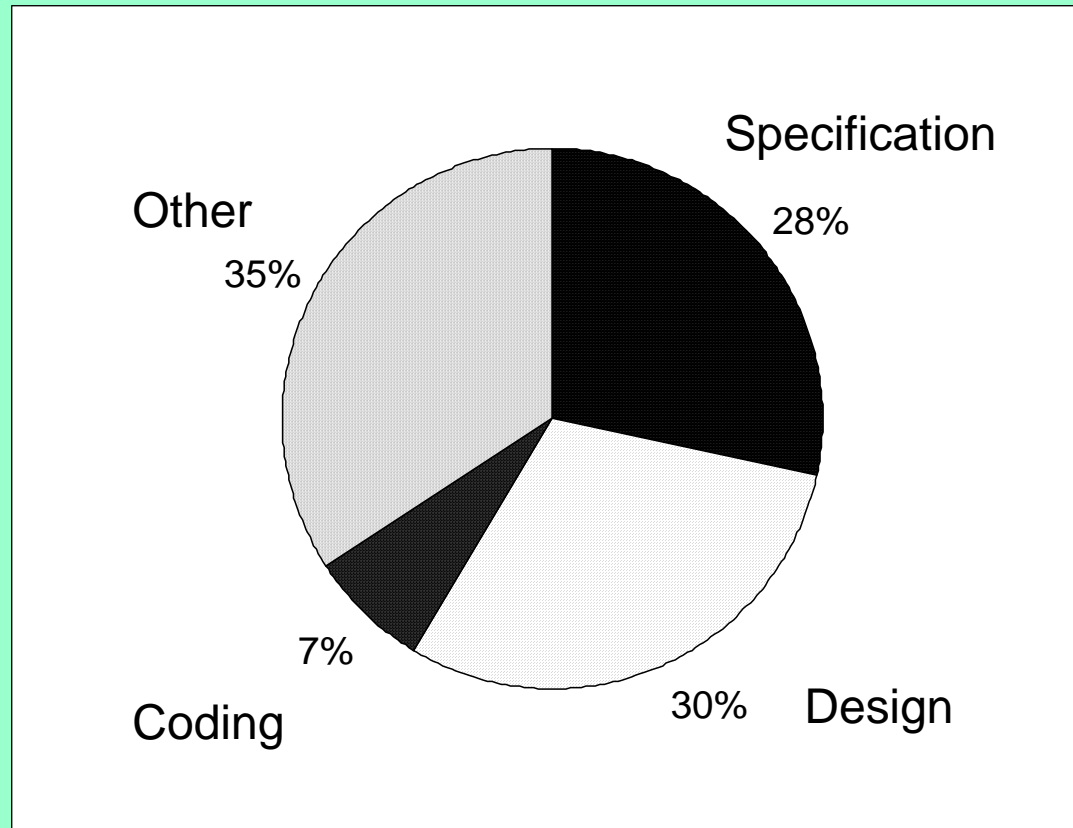
	Jones*	Thayer**	Boehm***	Bell Labs
Requirements		10%	8-10%	15%
Functional Design	15%	55%	15-20%	
Logical Design	20%		25-35%	45% ←
Coding	30%	35%	25%	35%
Documentation, etc.	35%		17-20%	↓

* "Measuring Programming Quality and Productivity," Jones, *IBM Systems Journal*, 1978

** *Software Reliability: A Study of Large Project Reality*, Thayer, Lipow & Nelson, North-Holland, 1978

*** "Developing Small Scale Application Software Projects: Some Experimental Results," Boehm, *Proceedings, IFIP 8th World Computer Congress*, 1980

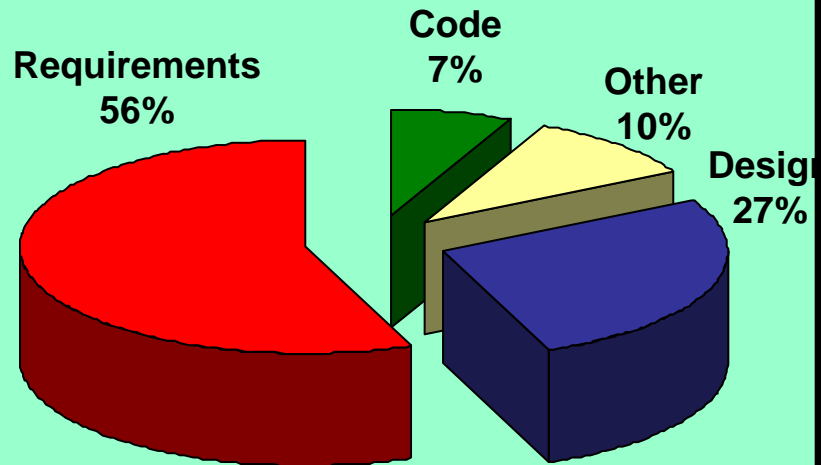
Hewlett Packard's Experience



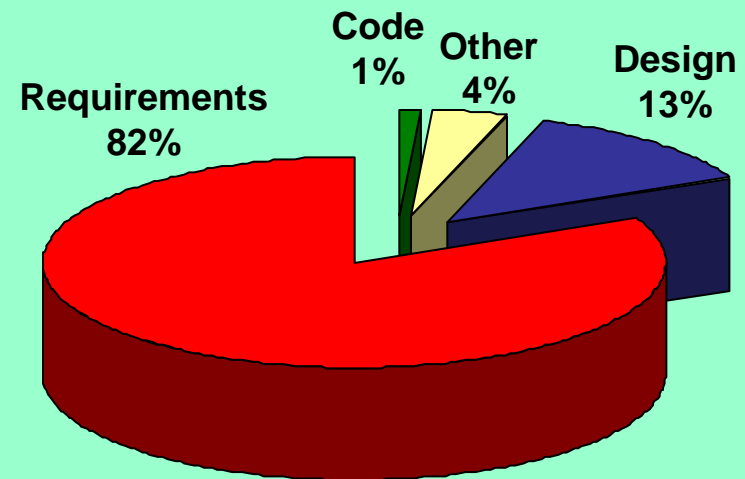
Source: "Implementing and Sustaining a Software Inspection Program in an R&D Environment," MacLeod, *Hewlett-Packard Journal*, June 1993 ←

Why Manage Requirements ?

Distribution of Defects

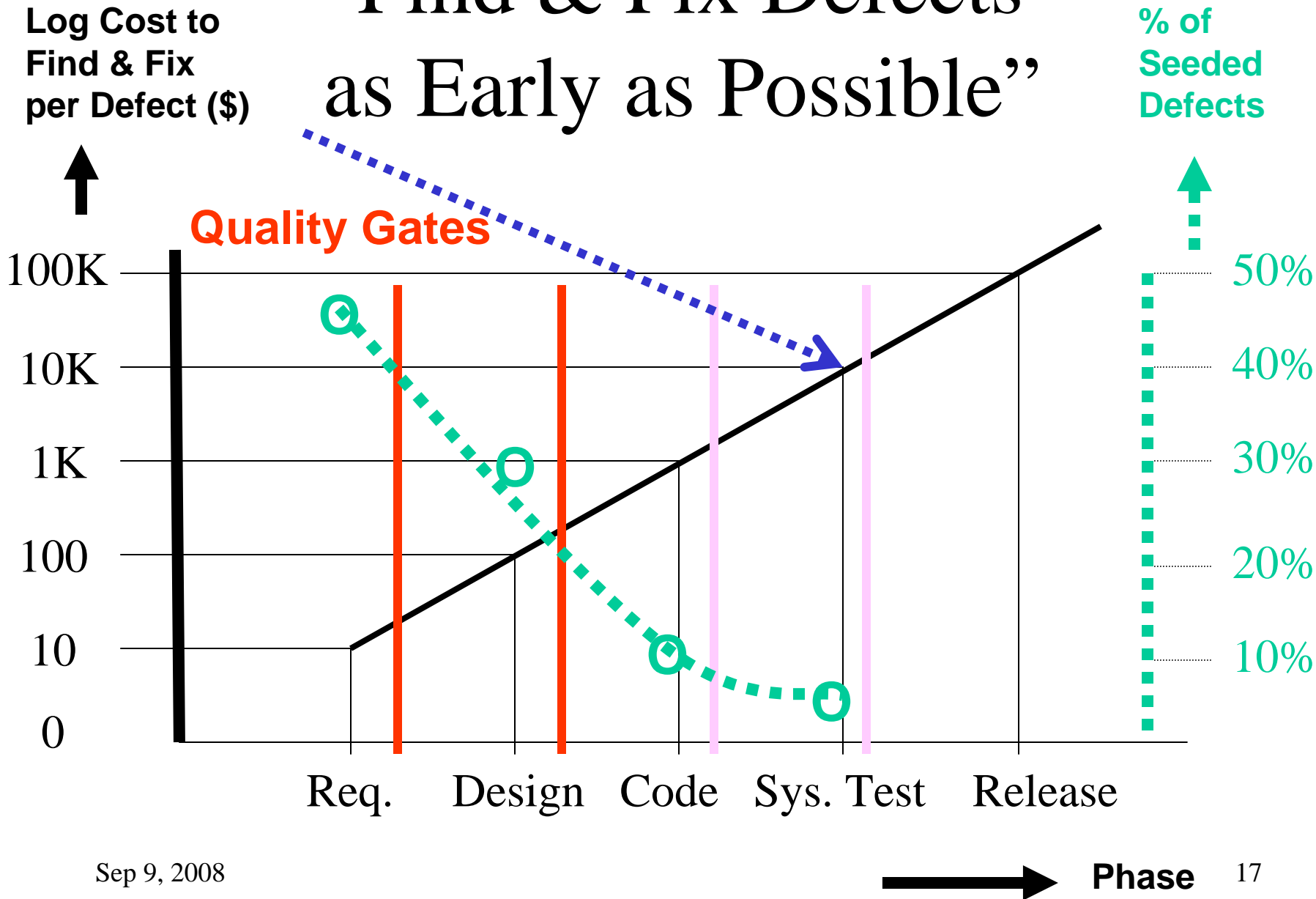


Distribution of Effort to Fix Defects



Source: Larry Boldt - Technology Builders, Inc. - Spin 2001

“Find & Fix Defects as Early as Possible”



Poor Requirements #1 Reason for Schedule Slip

Product Definition Impact

Root Cause for Development Schedule Slip

- **Poor requirements** 27%
- Unanticipated technical difficulties 25%
- Training 20%
- Poor Project Management 17%
- Lack of management support 6%
- Other 5%

Source: John Carter - Product Development Consulting, Inc
- Spin 1997 -

The Current State of the Requirements Phase

Product Definition Impact

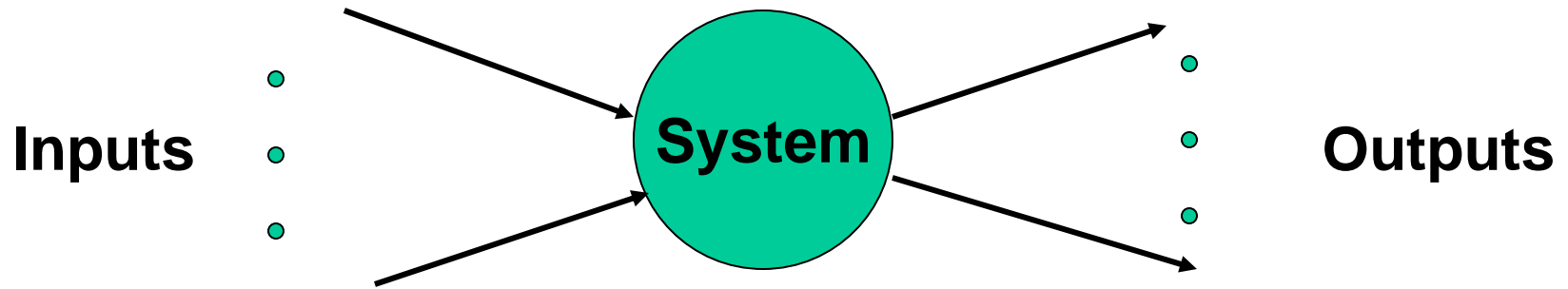
Engineers' Awareness at the Start of a Project

	<u>Others</u>	<u>Best</u>
• Priority Decision Criteria	20%	36%
• Product Positioning	22%	41%
• Strategic Alignment	18%	41 %
• Organization Support	20%	43%
• Risk Assessment	26%	44%
• Competitive Analysis	21%	47%
• Understanding User Needs	21%	48%
• Regulation Compliance	60%	73%



Source: John Carter - Product Development Consulting, Inc
Berkeley Software Forum - 1996

System Requirements – Definition



The system interface = description of the “shell” →
all **inputs & outputs** →

- Their valid domains
- Inter-logical/mathematical relationship
- Dependencies over time
- Technology of delivery

Requirements – Attributes

- Requirements **must** be –
 - **Understandable** by all stakeholders
 - **Unambiguous** – each statement has only one interpretation (!)
 - **Consistent** – none of the statements contradict another statement (or an optimization strategy)
 - **Complete** – including quality parameters, reliability, prioritization and contingencies
 - **Modifiable. Traceable. Usable. Measurable. Verifiable**

More About The Requirements Phase

- It is a **strategic** phase
- If this phase is not performed well → the project has a higher risk of failure
- Most organizations implement the phase poorly or even skip it
- More than 20% change in Product Requirements, the project may “crash*” (Bell Labs – 1980s)
- Requires about 15-20% of project resources

* “*Crash*” = *The project is late & over-budgeted*

The Requirements Phase – Market Demands

- Software systems become **more complex** →
 - distributed architecture
 - multi-threaded, concurrent processes
 - asynchronous/synchronous/proactive interactions
- Demands for real time and **higher speed**
- There is a need for **higher reliability**
 - constant automated, defensive, self testing (for integrity, completeness, collisions, synchronization, deadlocks)
 - in the event of a failure → fast and transparent perfect recovery

The Requirements Phase – The Current Patterns

- Requirements are fluid, poorly articulated and understood
- Late deliveries
- Cost over-runs
- Low reliability
- Slower performance than desired
- High stress level for customers and developers
- 90% of cost occur after the software has been released (!)
- Simply adding more developers is not a solution – “The Mythical Man Month*”

Challenges – Data

Studies by the Standish Group:

- Only 44% of software projects finish on time
 - Projects (*usually?*) complete at 222% of the duration originally planned
 - 189% of the original budgeted cost
 - 70% of projects fall short of their planned scope (technical content delivered)
 - 30% are cancelled before completion
-
- Ref:
<http://www.pqa.net/ProdServices/ccpm/W05002001.html>

Challenges – Data

Capers Jones, Software Productivity Research LLC

- An analysis of 250 large software projects
- Size > 10,000 function points (~ 1,250,000 statements in C)
- 1995 - 2004 data
- Success vs. failure were analyzed, at opposite ends of the spectrum (achieving cost, schedule, quality targets)
 - 25 projects were successful
 - 50 projects had delays or overruns below 35%
 - 175 projects experienced major delays, budget overruns, or were terminated without completion

More Tools – Formal Languages

- Prototyping
- Finite State Machine (FSM) → leading to Specification Definition Language (SDL)
- Unified Modeling Language (UML) → use case diagrams
- Entity Relationship Diagram (ER Diagram)
- Flow Charts (multiple variations)
- Decision Tables

More Tools – Formal Languages

- **1970s** – U.S. Air Force Program for Integrated Computer Aided Manufacturing (ICAM)
- **1981** – IDEF = ICAM Definition
 - IDEF0 = Functional Model
 - IDEF1 = Information Model
 - IDEF2 = Dynamic Model
- **1991** – **The National Institute of Standards and Technology (NIST) defined Federal Information Process Standards (FIPS)**
 - IDEF0 → FIPS 183
 - IDEF1 → FIPS 184

Formal Requirements Languages – Benefits

- Mathematically based → Un-ambiguous = a symbolic system description
- At a high level, it describes **what** the system does, **not how** the system does it
- When reducing the level of abstraction → specifications are transformed into **design (the how)** → to **code** => **CASE*** tools

* *Computer Aided Software Engineering*

More Tools – Inspections

- 1976 – Introduced by **M. E. Fagan - IBM**
- A formal process
- Bell Labs experience:
 - Cost of inspecting all deliverables 5-20% of project cost*
 - ROI of inspecting all deliverables 200-1200%*

* *Bell Labs data*

HP – Inspection Data

Grady, R. B. and Van Slack, T., “*Key Lessons in Achieving Widespread Inspection Use*”, *IEEE Software*, V. 11, N. 4, Month, 1994, pp. 46-57

Phase	Est. Starting Point	Est. Adoption 1993	% Cost Saved	% Rework	Est. \$ Savings Per Year
Spec.	1%	29.5%	28.5%	17%	\$10,175,000
Design	1%	34.8%	33.8%	11%	\$7,808,000
Code	5%	42.3%	37.3%	4%	\$3,133,000
Test Plan	1%	17.1%	16.1%	1%	\$338,000
TOTAL				33%	\$21,454,000

Requirements Phase Deliverables

- Market Requirements Document
- Product Specifications (Specs)
- High Level Project Plan
- Process Documents
- Black Box and Acceptance Test Plans
- An initial contract with the customer(s)
- Management support = Budget and infrastructure (for the next phase)

The Message

A high quality requirements →

“Gets the product right the first time” →

Cost/Time saving and completion of Scope