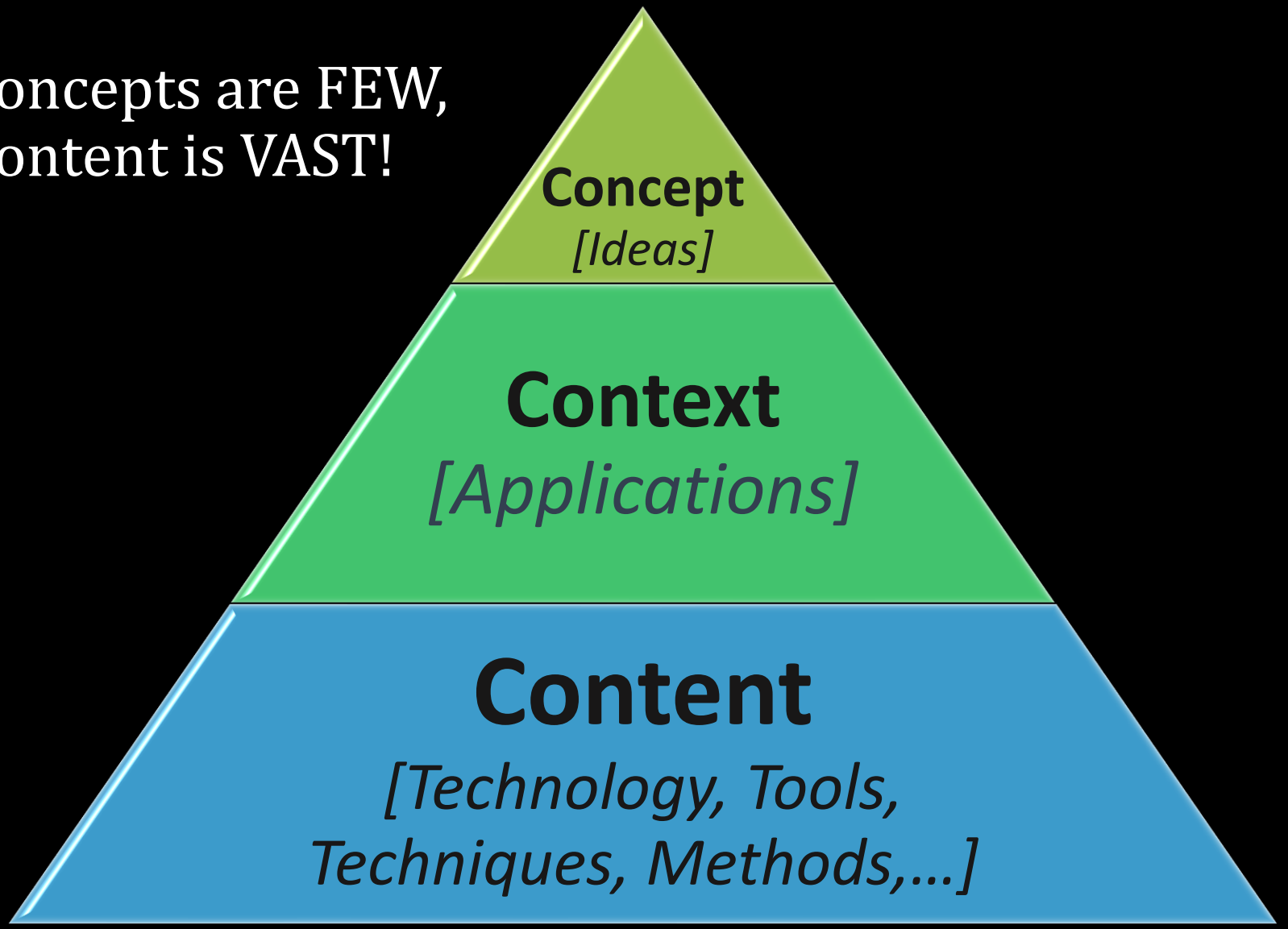
Software Quality Management

- K G Krishna

Let's Focus on **Key Concepts** Driving Quality Across Products & Services

Concepts are FEW,
Content is VAST!

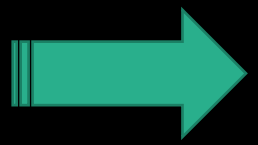


Software Quality Management

Thank You

In the next session, we shall:

Demystify Quality Vocabulary (Engineering, Technology, Methodology, Framework, Process, Model, SQA, SQC,…



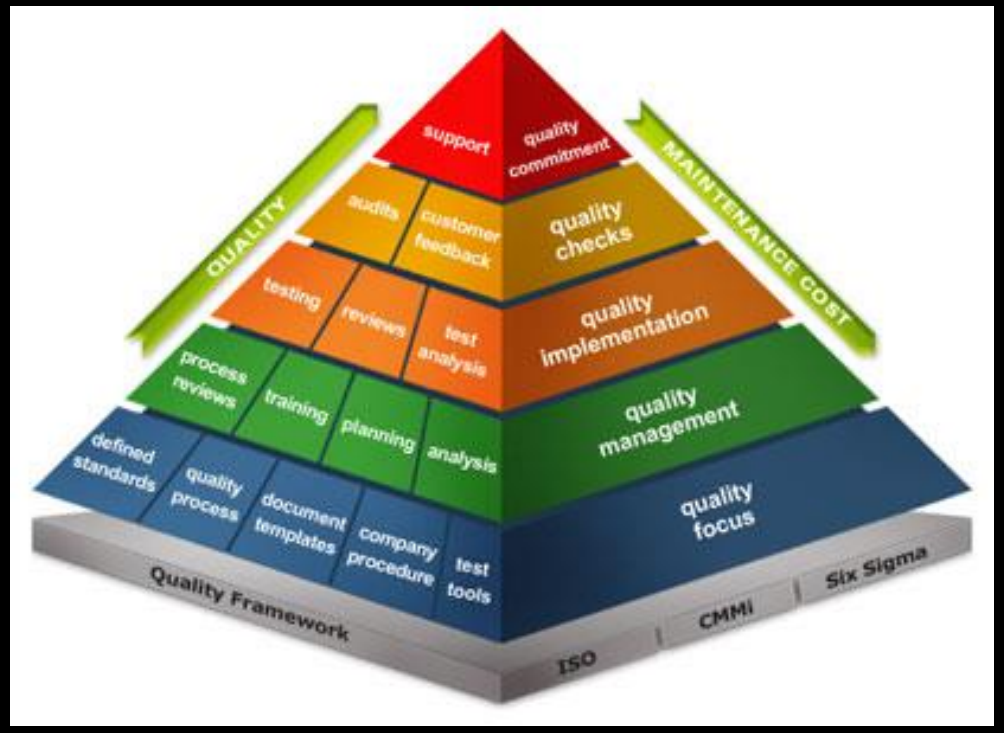
Software Quality Management

- K G Krishna

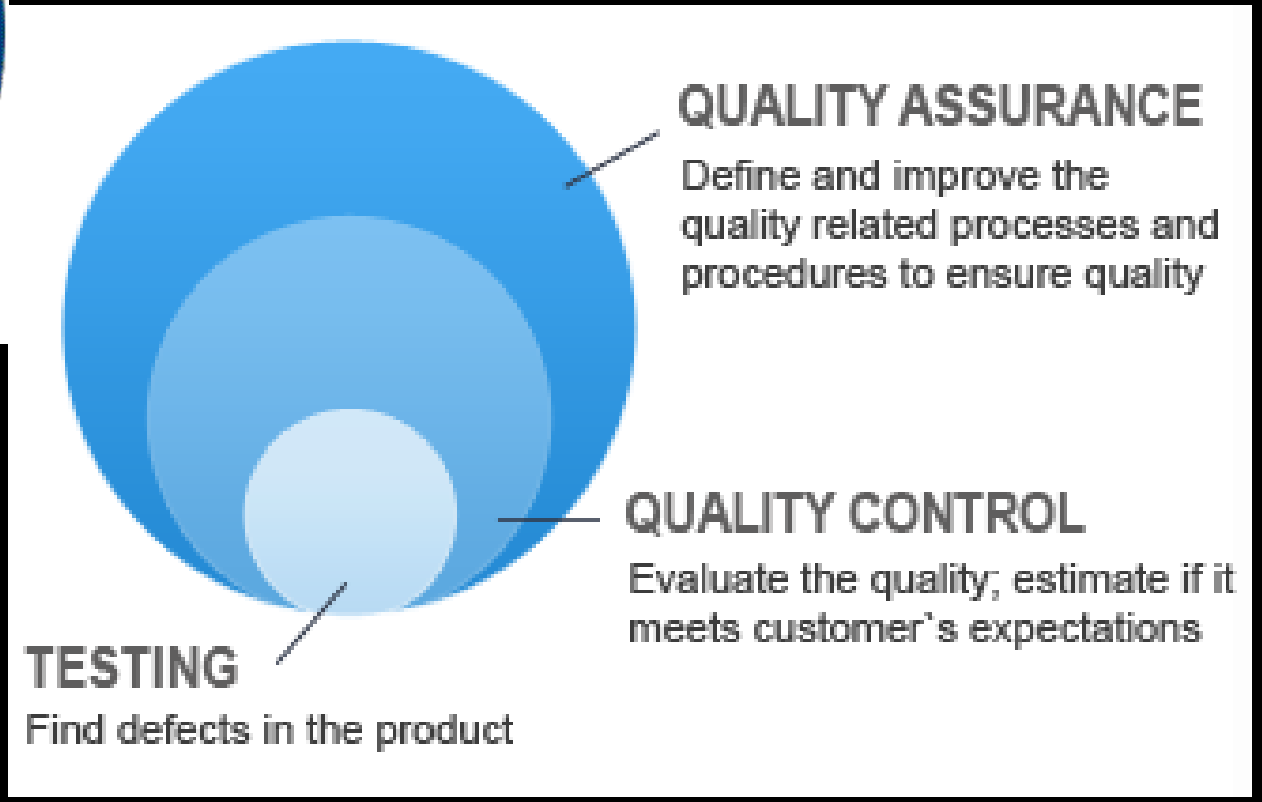
Demystifying Quality Concepts ➔

Let's Demystify Quality Concepts (Vocabulary)

Software Engineering, Technology,
Quality Assurance, Quality Control,…

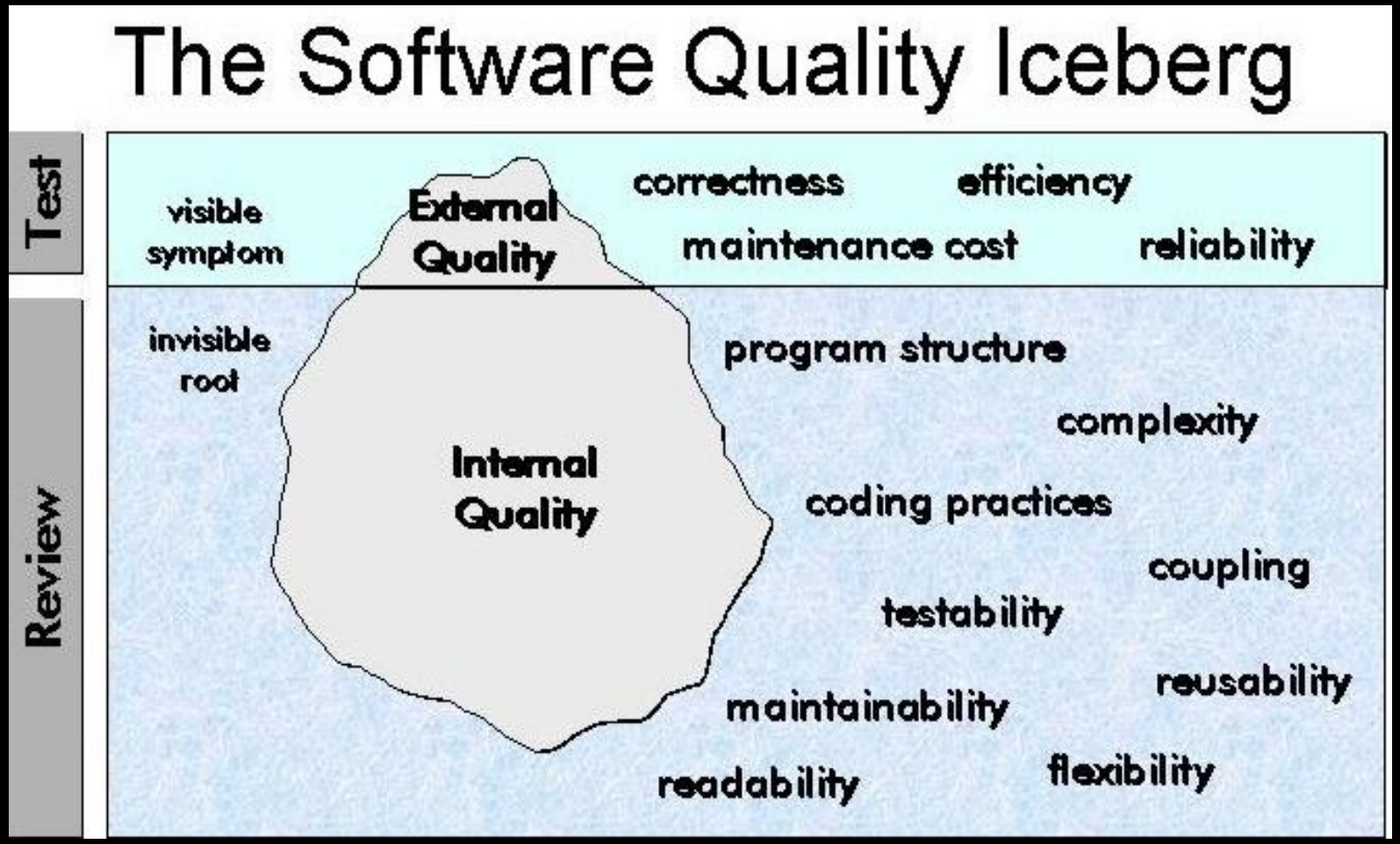


Quality Control vs. Quality Assurance



http://www.highlandsinfotech.com/

The ‘Invisible’ Part of Software Quality



Source courtesy: http://software-quality.blogspot.in/

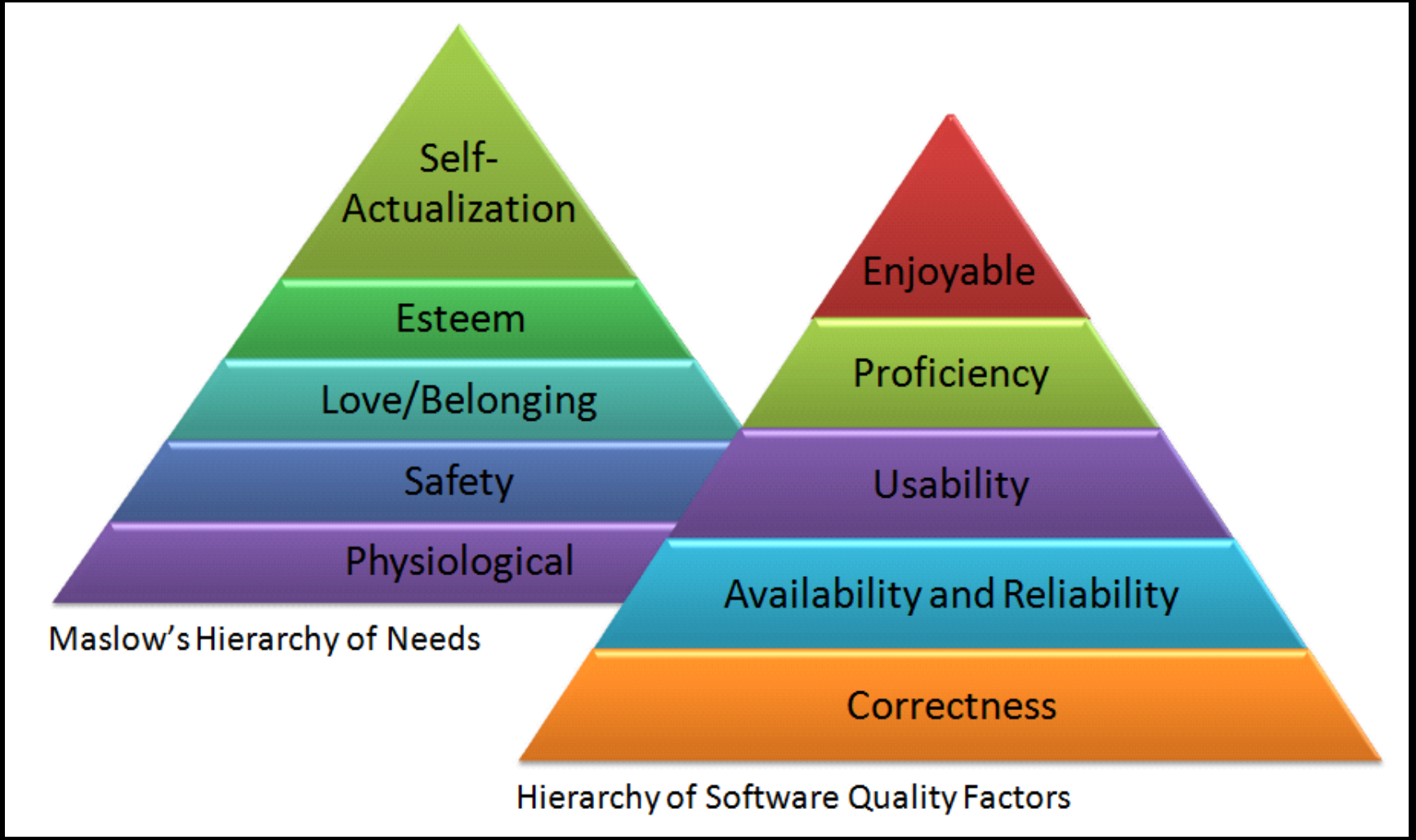
Formal Definitions of 'Software Quality'

(**IEEE** 610.12-1990) Standard Glossary of Software Engineering Terminology:
"the degree to which a system. component, or process meets (1) specified requirements, and (2) customer or user needs or expectations"

(**ISO** 9003-3-1991) Guidelines for the application of ISO 9001 to the Development, Supply and Maintenance of Software:
"the totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs"

The ‘Business’ of Software Quality?

The *Nirvana* of Quality!



Source courtesy: http://sce2.umkc.edu/

"Quality means doing it right when no one is looking"

He was a relentless technological innovator, based on his commitment to creating “the best possible goods at the lowest possible price.” He focused on making ongoing changes in design and production that would drive down costs while improving the product. For instance, the inaugural Model T, released for sale in 1908, cost $825 (about $22,000 in present-day dollars); by 1916, he had reduced the cost by more than half – to $360, while increasing safety, reliability and speed. And for better or worse, Henry Ford – more than any other individual – made us a nation of car owners; at one point over half the families in America owned a Ford motor car.

“There is one rule for the industrialist and that is: make the best quality goods possible at the lowest cost possible, paying the highest wages possible.” - Henry Ford

We cannot rely on mass inspection to improve quality, though there are times when 100 percent inspection is necessary. As Harold S. Dodge said many years ago, 'You cannot inspect quality into a product.' The quality is there or it isn't by the time it's inspected.

(W. Edwards Deming)

Cost is more important than quality but quality is the best way to reduce cost.

— Genichi Taguchi —

AZ QUOTES

Quality has to be caused, not controlled.

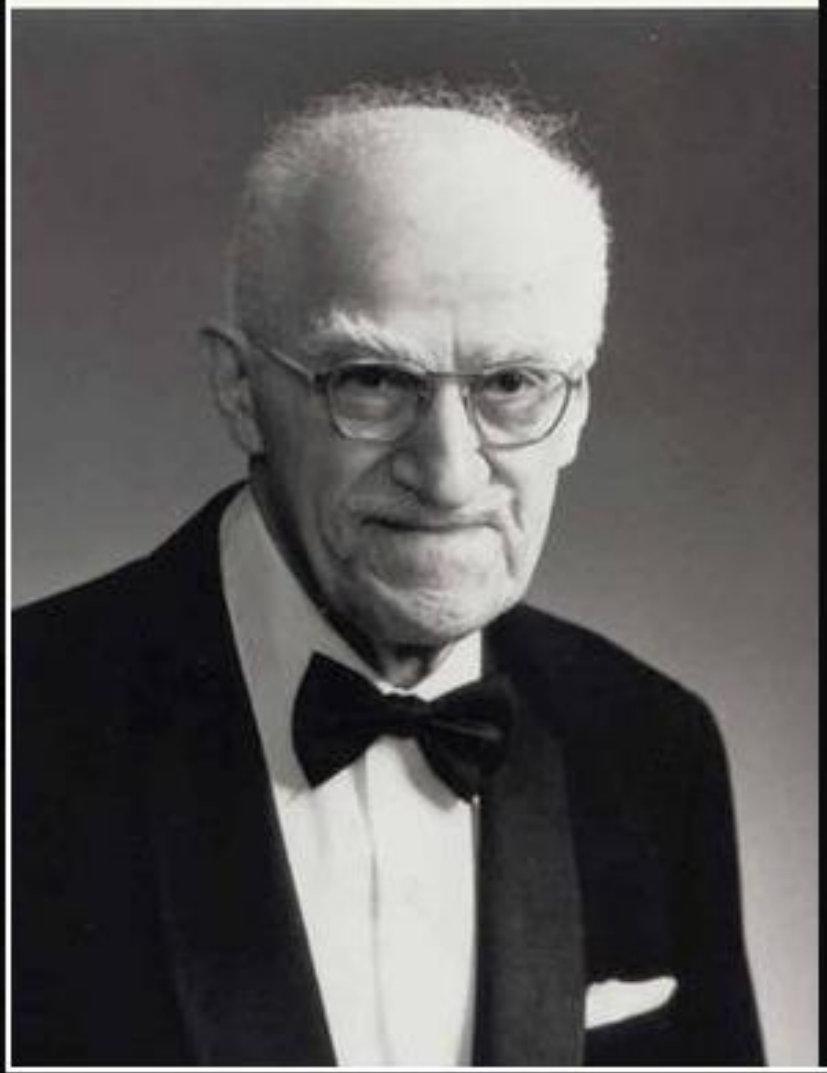
— Phil Crosby —

AZ QUOTES

As with many other things, there is a surprising amount of prejudice against quality control, but the proof of the pudding is still in the eating.

— Kaoru Ishikawa —

AZ QUOTES



All improvement happens project by project and in no other way.

— Joseph M. Juran —

Cost is more important than quality but quality is the best way to reduce cost.

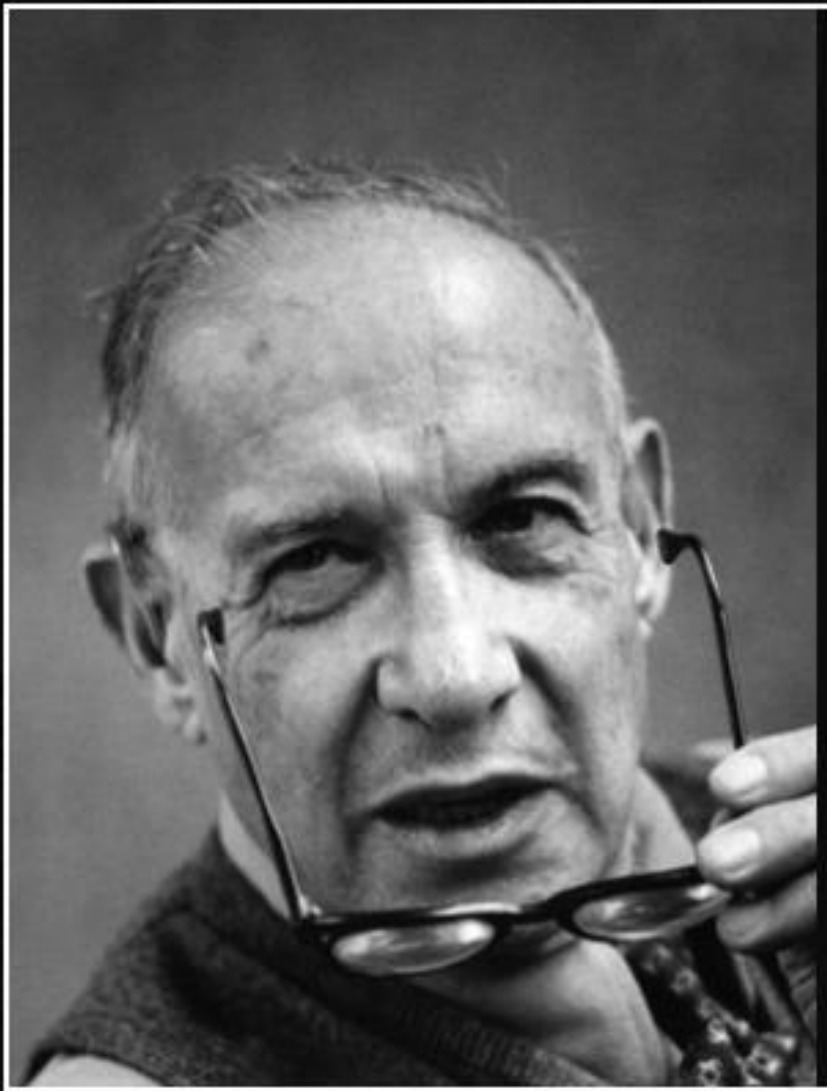
— Genichi Taguchi —

AZ QUOTES

The message of the Kaizen strategy is that not a day should go by without some kind of improvement being made somewhere in the company.

— Masaaki Imai —

AZ QUOTES



Management is doing things right; leadership is doing the right things.

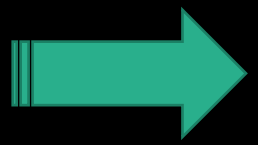
— Peter Drucker —

AZ QUOTES

Thank You

In the next session, we shall:

Discuss 'working' definition of Quality in the content of Software Engineering and overview of various Software Development Models…



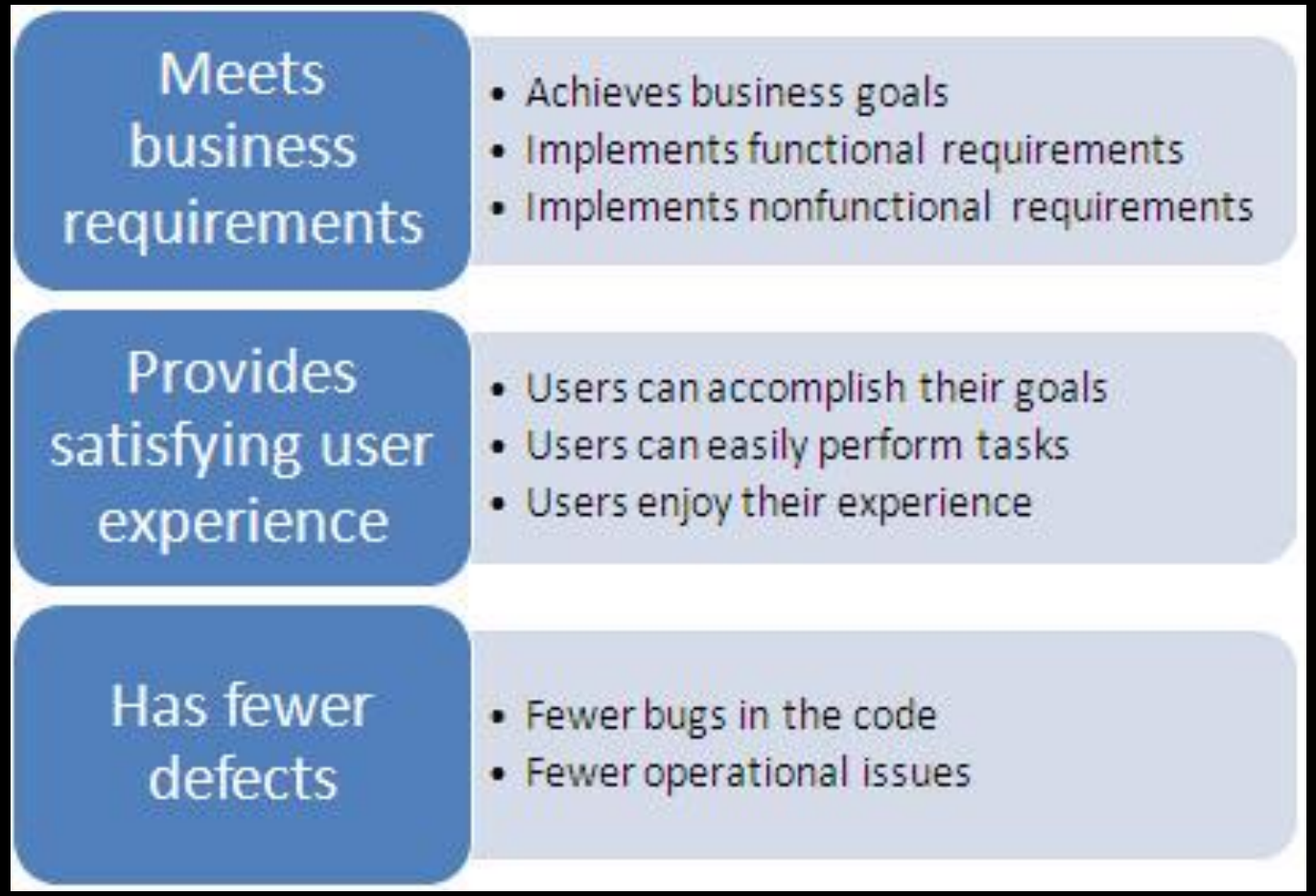
Thank You

Software Quality Management

- K G Krishna

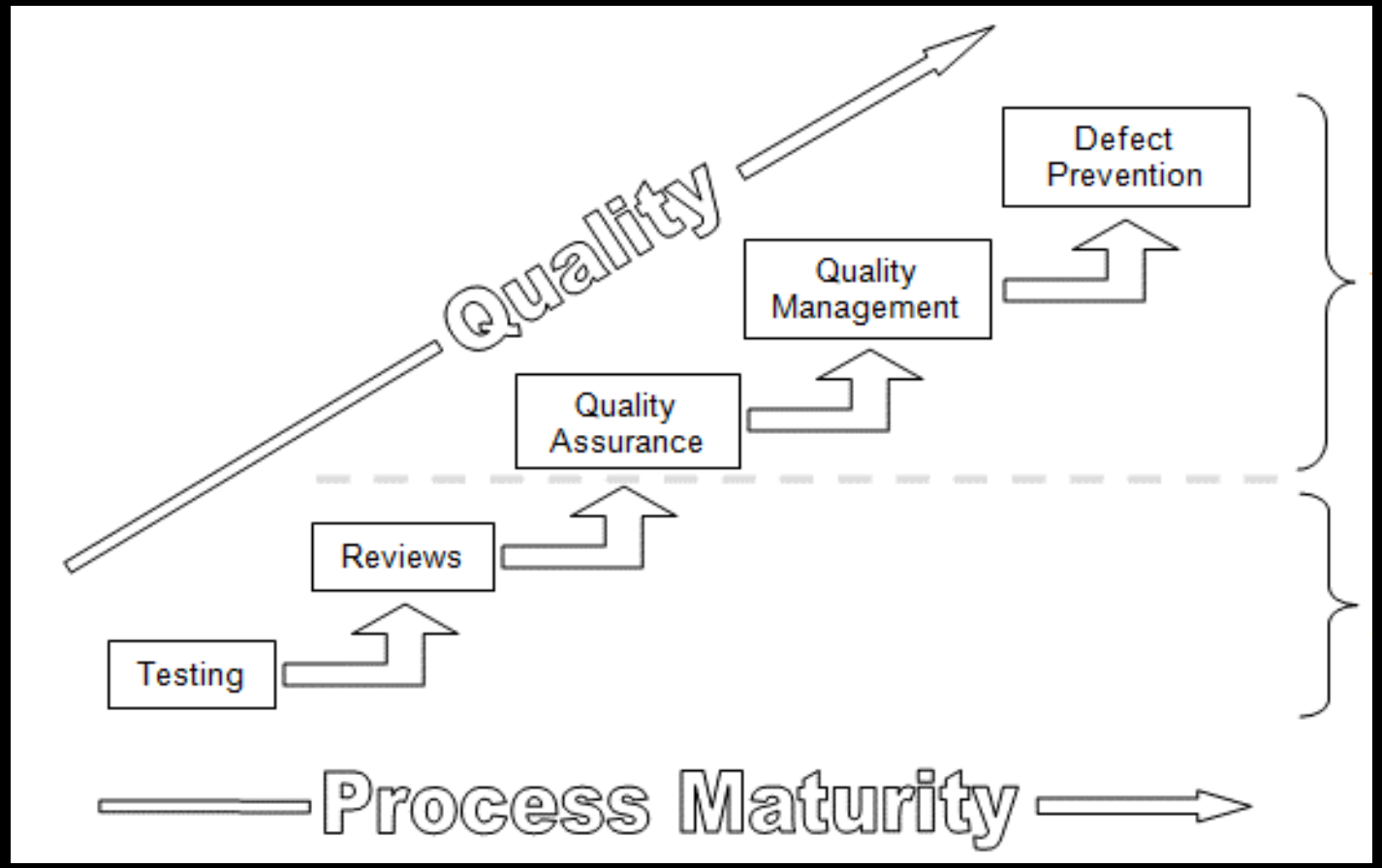
Software Quality Processes ➔

To Summarize, Quality is:

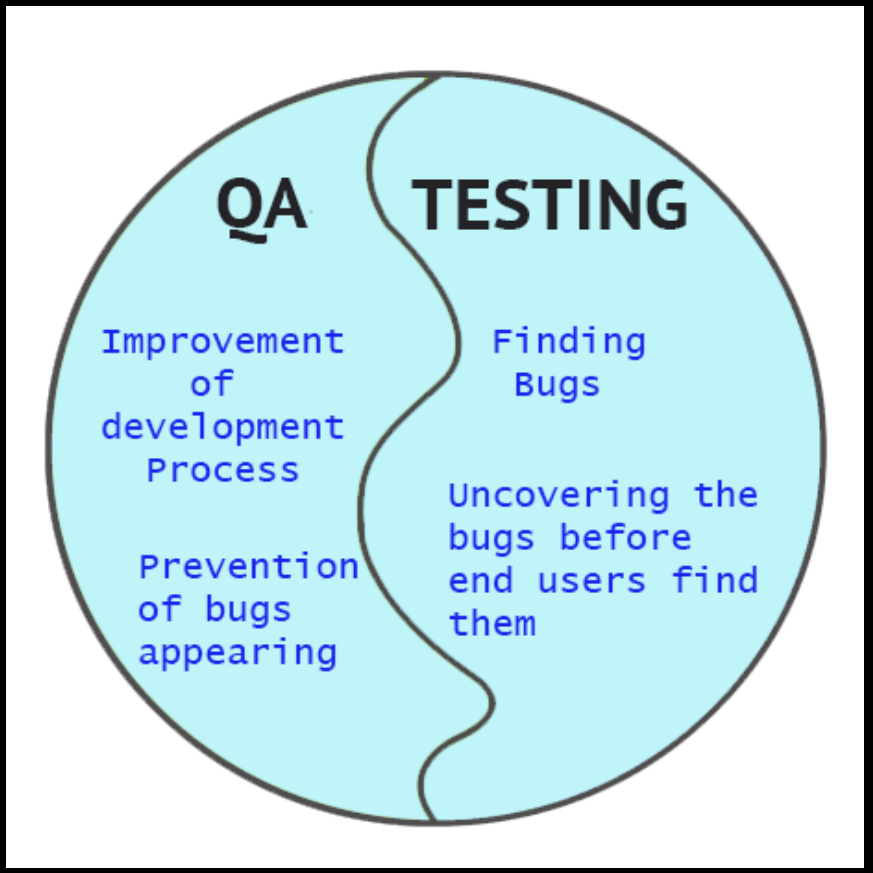


http://blogs.forrester.com/mike_gualtieri

Quality Process (QMS) Maturity
(QC → QA → QM → …Defect Prevention …→ Innovation?)

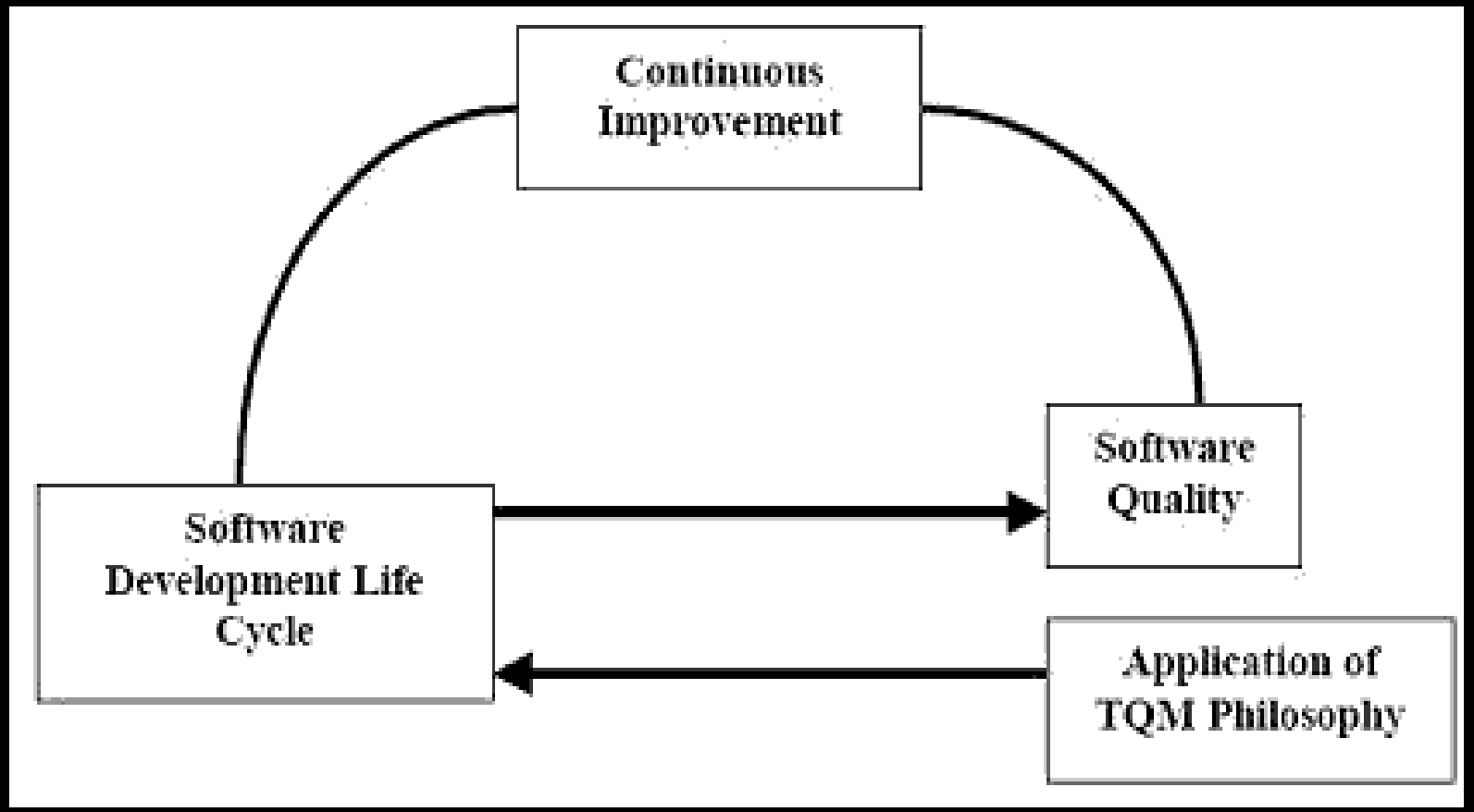


Source courtesy: http://sce2.umkc.edu/BIT/burrise/pl/software_quality_management/



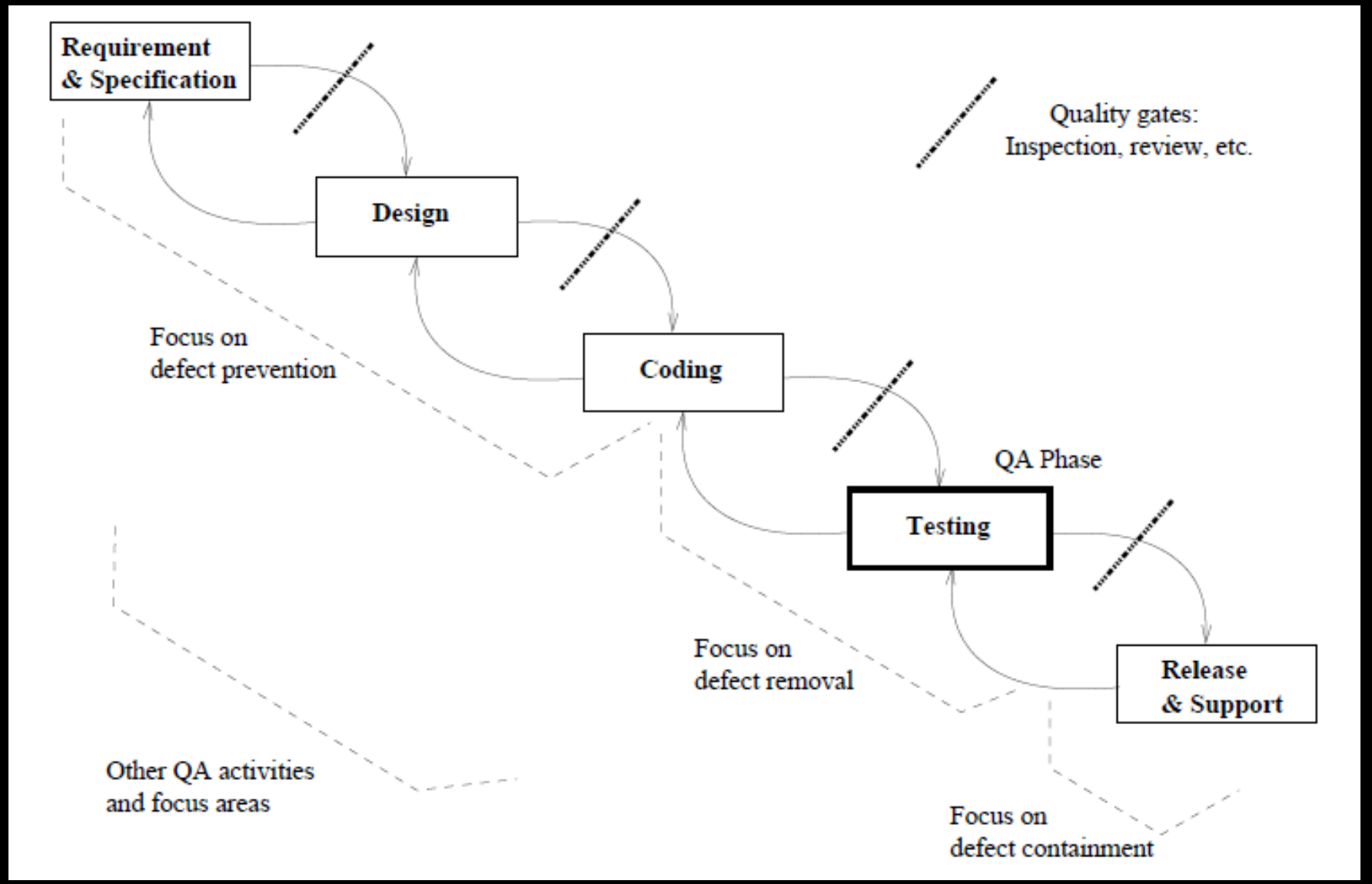
Source courtesy: http://dtechviews.com/software-testing-and-quality-assurance-report/

Continuous Improvement:
(Backbone of QMS/TQM)



Source courtesy: http://www.freetutes.com/systemanalysis/sa9-software-quality-assurance.html

Waterfall Model: *Traditional* Process



QA Scattered throughout all Processes

- **QA Throughout the Process**
  - Defect prevention in early phases
  - Focused defect removal in testing phase
  - Defect containment in late phases
  - Phase transitions: inspection/review/etc. (Gated reviews,…)

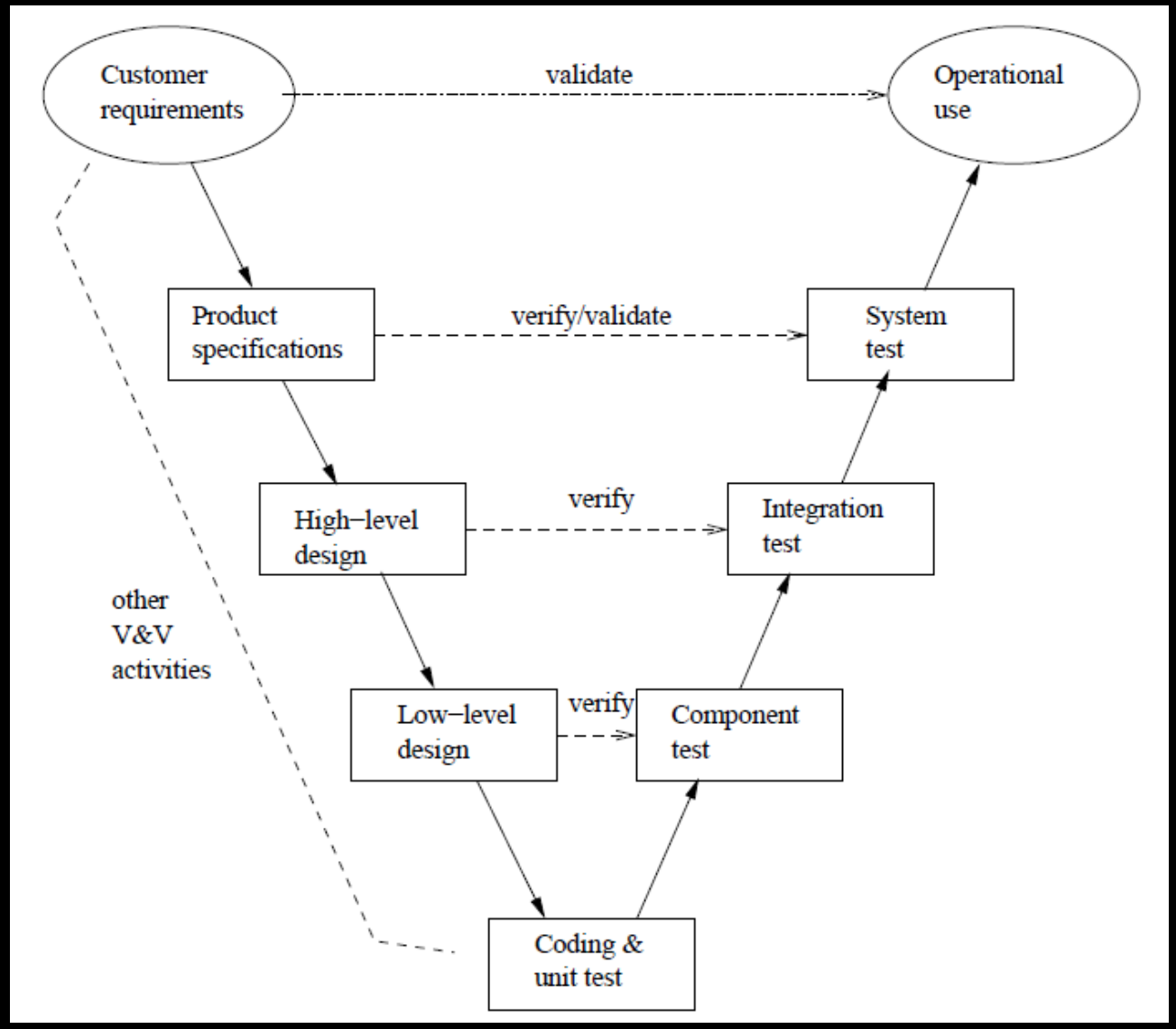
- **Process variations (: waterfall) and QA:**
  - iterative: QA in iterations/increments
  - spiral: QA and risk management
  - .XP: test-driven development

- **QA in maintenance processes:**
  - focus on defect handling;
  - some defect containment activities for critical or highly-dependable systems; data for future QA activities

Core QA Activities:
(Verification & Validation)

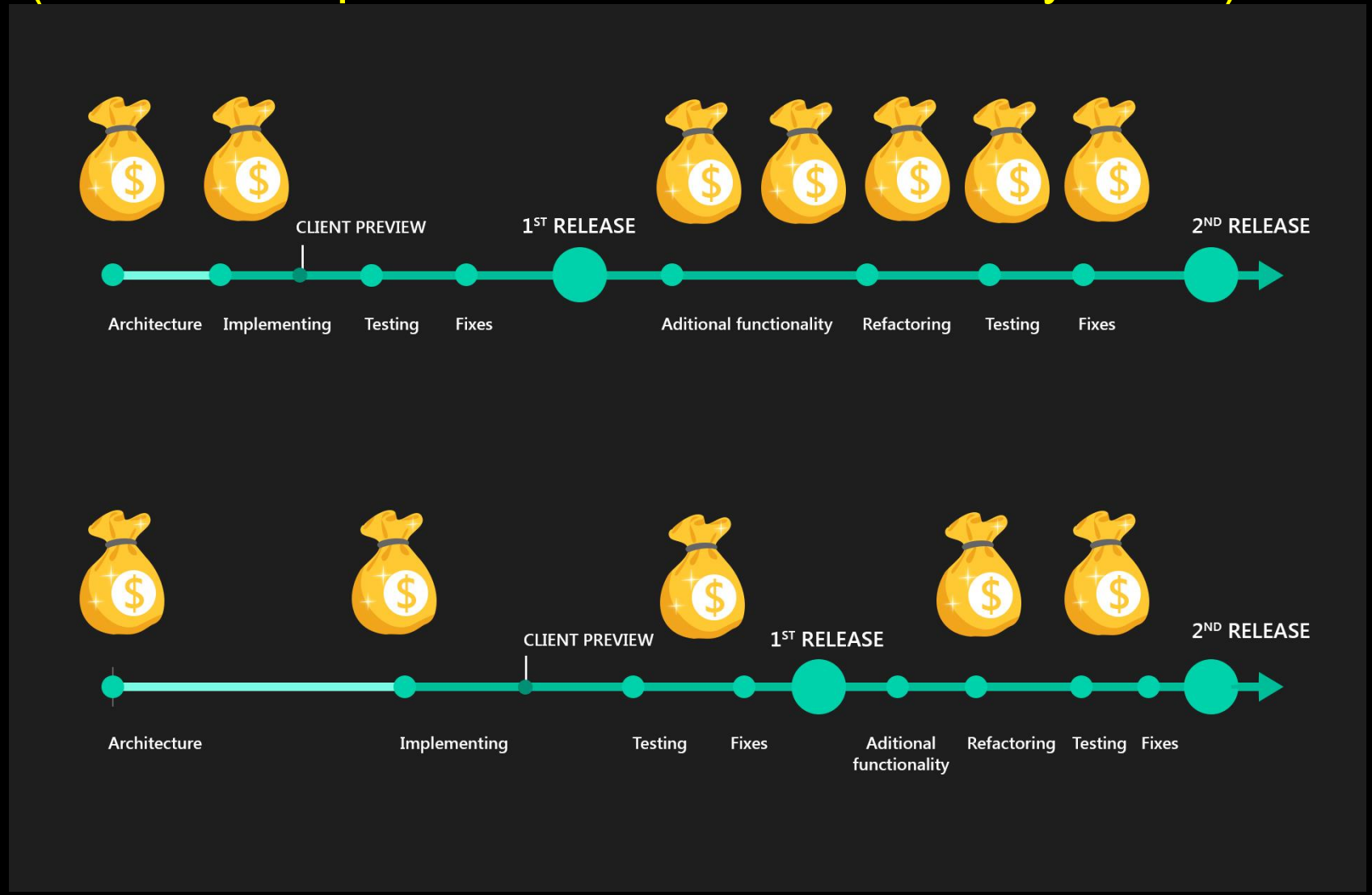
- **Validation: w.r.t. requirement (what?)**
  - appropriate/fit-for-use/ “right thing"?
  - scenario and usage inspection/testing;
  - system/integration/acceptance testing;
  - beta testing and operational support.
- **Verification: w.r.t. specification/design (how?)**
  - correct/ “doing things right"?
  - design as specification for components;
  - structural and functional testing;
  - inspections and formal verification.

V-Process Model: QA Activities



- **V&V in V-model:**
  - V-model as bent-over waterfall
  - left-arm: implementation (& V&V)
  - right-arm: testing (& V&V)
  - Verification: more internal focus
  - Validation: more external focus

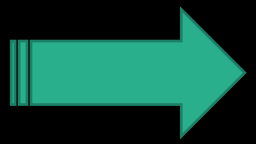
Quality vs. Business Deliverables:
(Customer-specified Milestone-linked Payments)



http://hypersense-software.com/blog/2015/04/it-excellence/

Thank You

In the next session, we shall Discuss:

Software Testing 

Software Quality Management

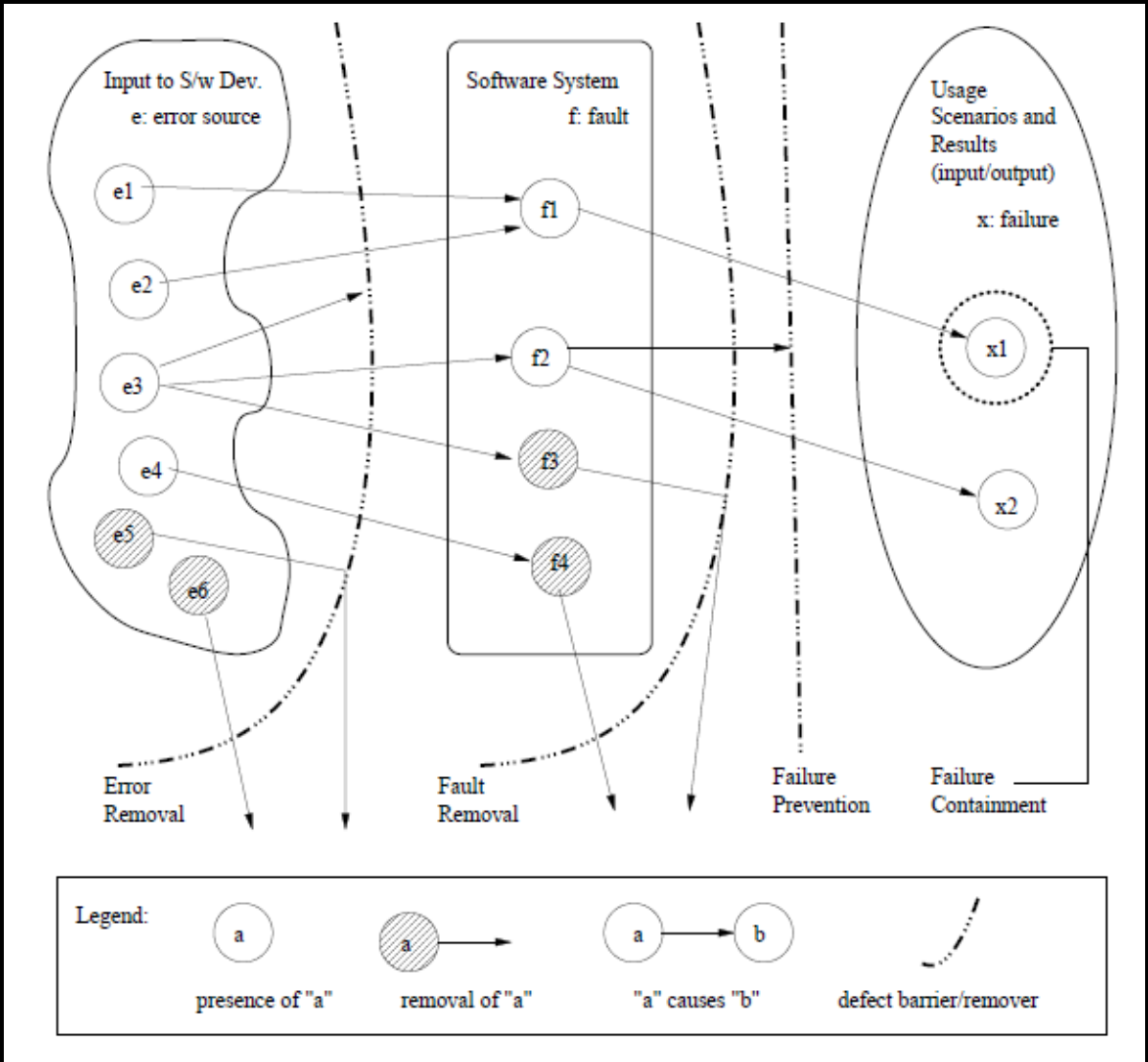
- K G Krishna

*"**Process** enables ordinary people to perform extraordinary things."*

SQA Activities

- Focus on Defect Prevention (QA) vs. Defect Detection/Fix (QC)
  - ✓"Prevention (earlier in the development) is better than Cure (bug-detect and fix in later stages by Testing)
- SQA Processes ensure systematic prevention of defects throughout the development life-cycle (e.g., SQA activities in parallel to development activities in V-Process or test-driven development in Agile Methods)
- QMS (ISO 9000/CMMI) insist on 'Continuous Improvement' Process for periodic assessment and improvement of SQA processes in the organization

What is a 'Defect' *(Error/Bug, Fault, Failure)* ?



Defect Prevention means…

- Preventing fault injection
  - error blocking (error does NOT lead to faults)
  - error source removal (reviews, training,…)
- Removal of faults (pre: detection)
  - inspection: faults discovered/removed
  - testing: failures trace back to faults
- Failure prevention and containment:
  - local failure does NOT lead to global failure (via dynamic measures to tolerate faults)
- Low failure impact => safety assurance

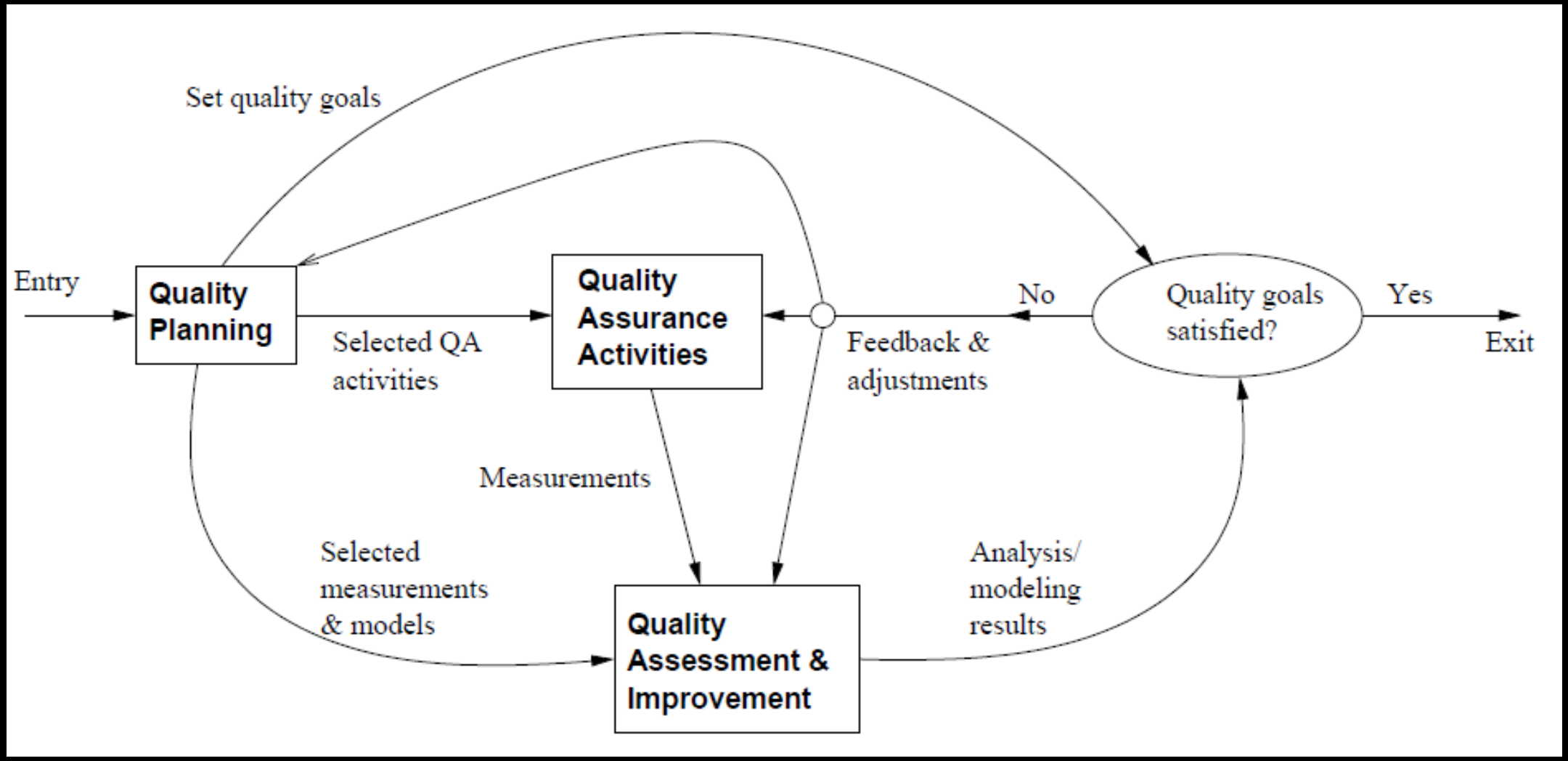
Software Testing:
(Defect Detection & Fix)

- Dependent on Product/Process characteristics:
  - Product type, language, etc.
  - Scale/order: unit, component, system, : :
  - Who: self, independent, 3rd party
- What to check:
  - Verification vs. Validation (Requirements)
  - External specifications (black-box)
  - Internal implementation (code: white/clear-box)
- Criteria: when to stop?
  - Coverage of specs/structures.
  - Reliability via usage-based testing

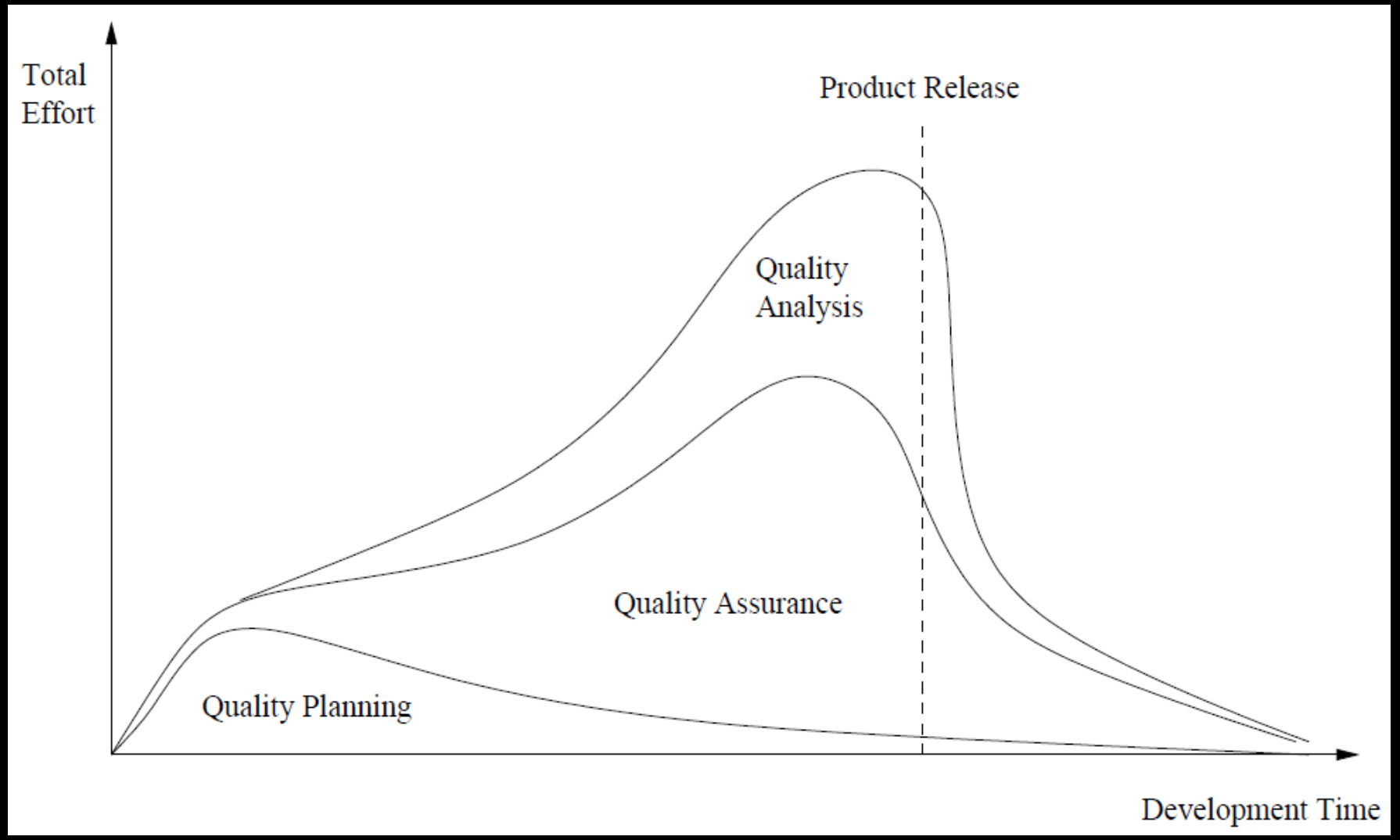
Defect Measurement and Analysis

- Defect Measurement
  - where injected/found?
  - type/severity/impact?
  - Classification of defects
  - consistent interpretation
  - timely defect reporting
- Defect Analysis
  - data and historical baselines
  - goal: assessment/prediction/improvement
  - causal/risk/reliability/etc. analyses

SQA: “The Big Picture”

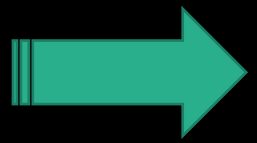


SQA Effort in Typical Waterfall Model



Thank You

In the next session, we shall look at the most common SQA activity:

Software Testing 

Software Quality Management

- K G Krishna

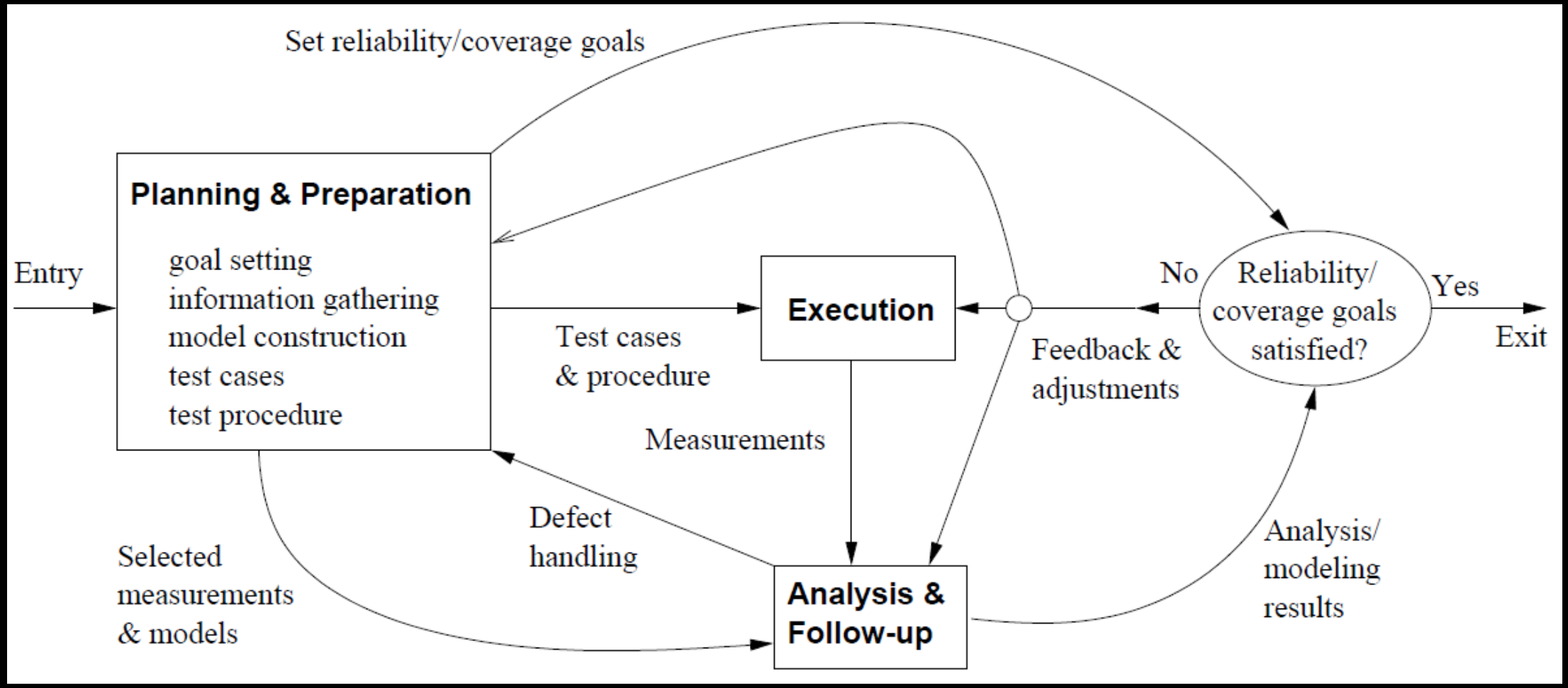
Software Testing:
(Defect Detection & Fix)

- Dependent on Product/Process characteristics:
  - Product type, language, etc.
  - Scale/order: unit, component, system, : :
  - Who: self, independent, 3rd party
- What to check:
  - Verification vs. Validation (Requirements)
  - External specifications (black-box)
  - Internal implementation (white/clear-box)
- Criteria: when to stop?
  - Coverage of specs/structures.
  - Reliability via usage-based testing

Testing: Defect Detection & Removal

- Failure observation ➔fault removal.
- Defect detection ➔ defect fixing
- The ‘curse’ of software: flexibility of software (ease of change; continuous change)!

Software Testing: The Process



Test Planning, Preparation & Execution

- Test planning:
  - goal setting based on customers' quality perspectives and expectations.
  - overall strategy based on the above and product / environmental characteristics.
- Test preparation:
  - preparing test cases/suites (typically based on formal models)
  - preparing test procedure
- Test execution
  - allocating test time (& resources)
  - invoking test (tools)
  - identifying system failures (measurement & analysis)
  - key to execution: handling both *normal vs. abnormal* cases

Testing Perspective: Functional & Structural

- Functional testing (Black-Box):
  - tests external functions (as described by external specs.)
  - black-box in nature; (functional mapping: input –output without involving internal knowledge)
- Structural testing (White-Box):
  - tests internal implementations (components and structures)
  - white-box in nature (“white" = seeing through, internal elements visible)
- Grey-box (mixed black-/white-) testing:
  - ex: procedures in modules
    - procedures individually as black box;
    - procedure interconnection ~ white-box at module level.

White-Box Testing:

- Requires Program component/structure knowledge
  - statement/component checklist
  - path (control flow) testing
  - data (flow) dependency testing
- Applicability
  - test in small/early
  - dual role of programmers/testers
  - can be modelled after specifications
- Criterion for stopping
  - mostly coverage (program/statements) goals
  - occasionally quality/reliability goals.

Black-Box Testing:

- Input/output behaviour
  - specification checklist.
  - testing expected/specified behaviour (FSMs,…)
  - white-box techniques on specification (functional execution path testing)
- Applicability
  - late in testing: system testing etc.
  - suitable for IV&V (Independent V&V)
  - compatible with OO/Reuse paradigm
- Criteria: when to stop
  - traditional: functional coverage
  - usage-based: reliability target

The ‘Big’ Question: When To Stop Testing?

- Resource-based criteria:
  - Stop when you run out of time / money ?
  - ?
- Quality-based criteria:
  - Stop when quality goals reached.
  - Direct quality measure: reliability (resemble actual customer usage profile)
  - Indirect quality measure: coverage.
  - Other surrogate: activity completion (??)

Usage-Based Testing (UBT)

- Usage-based statistical testing:
  - actual usage and scenarios/information
  - captured in operational proles (OPs)
  - simulated in testing environment (if too numerous, opt for random sampling)

- Applicability
  - final stages of testing.
  - particularly system/acceptance testing.
  - use with s/w reliability engineering.

- Termination criteria: *reliability* goals

Coverage-Based Testing (CBT)

- Coverage-based testing:
  - systematic testing based on formal (BBT/WBT) models and techniques
  - coverage measures defined for models
  - testing managed by coverage goals
- Applicability
  - all stages of testing.
  - particularly unit and component testing.
  - later phases at high abstraction levels.
- Termination criteria: *coverage* goals

Systematic Testing based on Formal Models

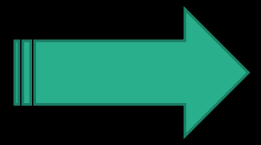
- Steps in Systematic Testing
  - Formalized strategies/goals based on formal models managed by termination criteria

- Steps in model construction and usage:
  - Define the model, usually represented as graphs and relations.
  - “Check" individual elements:
  - “Test": derive (sensitize) test cases and then execute them.
  - Result checking and follow-up

Thank You

In the next session, we shall look at:

Testing Techniques in detail…



Software Quality Management

- K G Krishna

Testing Techniques ➔

Common Testing Techniques…

❖**Coverage and Usage Testing based on Checklists and Partitions**

- Checklist-Based Testing
- Partitions and Partition Testing
- Usage-Based Testing with OPs (Operational Profiles)

Checklists for Testing?

- **Ad hoc testing:**
  - "run-and-observe"
  - How to start the run?
  - Areas/focuses of "observations"? *(Implicit checklists)*
- **Explicit checklists:**
  - Function/features (external)
  - Implementation (internal)
  - Standards to be followed, etc.
  - A combination of the above…

- **Function/feature (external) checklists:**
  - Black-box in nature
  - List of major functions expected
- **Implementation (internal) checklists:**
  - White-box in nature
  - At different levels of abstraction
  - e.g., lists of modules/components/etc.
  - statement coverage as covering a list

Example:

- abnormal termination
- backup and restore
- communication
- co-existence
- file I/O
- gateway
- index management
- installation
- logging and recovery
- locking
- migration
- stress

Partition-Based Testing

- Examples: solving $ax^2 + bx + c = 0$,
  - ▷ solution: $r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
  - ▷ input: a, b, c; output: r.
  - ▷ 32 bits floating point numbers used
  - ▷ input combinations:

$$2^{32} \times 2^{32} \times 2^{32} = 2^{96}$$

- 3 solution partitions (Table 8.3, p.108):

Test Case	Condition $d = b^2 - 4ac$	Input a	b	c
1	$d > 0$	1	2	−1
2	$d = 0$	1	2	1
3	$d < 0$	1	2	3

- **Partitions: a special type of checklists**
  - Mutually exclusive → no overlaps.
  - Collectively exhaustive → coverage.
    - (address the above two problems of checklists while complexity of interactions is addressed by FSMs)

Partition of set S into subsets $G_1, G_2, \ldots, G_n$ ($G_i \subset S$):

▷ G_i's are mutually exclusive:

$$\forall i, j,\ i \neq j \Rightarrow G_i \cap G_j = \emptyset$$

▷ G_i's are collectively exhaustive:

$$\bigcup_{i=1}^{n} G_i = S.$$

Each G_i forms an equivalent class

Partition-Based Testing (contd.)

- **Partition-Based Testing**
  - Sampling from partitioned subsets.
  - Coverage of partitions: non-uniform?
  - Testing based on related problems:
    - usage-related problems?
    - boundary problems?
  - Testing based on level/hierarchy/etc.?

- **Usage-related problems?**
  - More use ➔ failures more likely
  - Usage information in testing (Musa's Operational Profiles)

- **Boundary problems?**
  - Input domain boundary testing

Usage-Based Statistical Testing (UBST)

- **Usage based statistical testing (UBST) to ensure reliability.**
  - ***Reliability:*** Probability of failure-free operation for a specific time period or a given set of input under a specific environment
    - *Reliability*: customer view of quality
    - *Probability*: statistical modeling
    - *Time/input/environment*: OP

- **OP: Operational Profile**
  - Quantitative characterization of the way a (software) system will be used.
  - Generate/execute test cases for UBST
  - Realistic reliability assessment
  - Development decisions/priorities

Advantages of USBT:

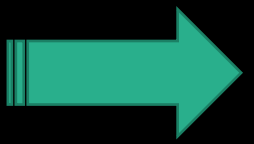
- **Primary Benefits**:
  - Overall reliability management.
  - Focus on high leverage parts
  - Productivity and schedule gains:
    (same effort on most-used parts; reduced effort on lesser-used parts)
    (reduction of 56% system testing cost; or 11.5% overall cost)

- **Introducing new product**
  - Highly-used features quickly
  - Lesser-used: subsequent releases
  - Better communications/customer relations
  - Customer perspective & involvement (closer ties to customers)
  - More precise requirement/specification
  - Better training focus

- **High return on investment:**
  - OP cost, “average" 1 person-month / 10 developers, 100KLOC, 18 months
  - (sub-linear increase for large ones; Cost-benefit ratio: 10)

Thank You

In the next session, we shall look at another important SQA Activity:

Reviews & Inspections 

Software Quality Management

- K G Krishna

Reviews & Inspections ➔

Software Review

A **software review** is *"A process or meeting during which a software product is examined by a project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval".*
- "IEEE Standard 1028-1997 for Software Reviews, clause 3.5"

Software peer reviews are conducted by the author of the work product, or by one or more colleagues of the author, to evaluate the technical content and/or quality of the work.

Software management reviews are conducted by management representatives to evaluate the status of work done and to make decisions regarding downstream activities

Software audit reviews are conducted by personnel external to the software project, to evaluate compliance with specifications, standards, contractual agreements, or other criteria

Source: Wikipedia, 2016

{Software Reviews, Inspections, Walkthroughs}

Code review is systematic examination (often as peer review) of computer source code. Pair programming is a type of code review where two persons develop code together at the same workstation.

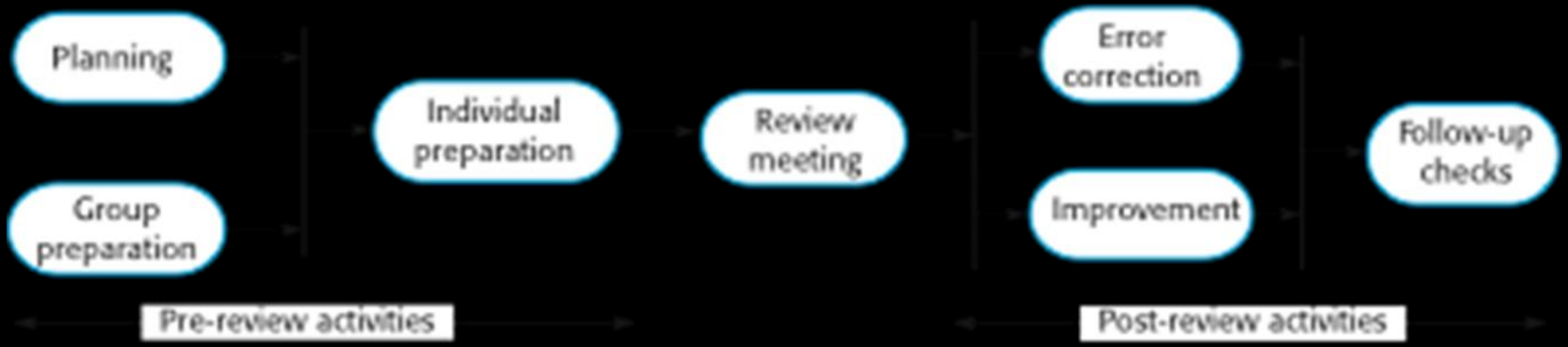
Inspection is a very formal type of peer review where the reviewers are following a well-defined process to find defects.

Walkthrough is a form of peer review where the author leads members of the development team and other interested parties through a software product and the participants ask questions and make comments about defects.

Technical review is a form of peer review in which a team of qualified personnel examines the suitability of the software product for its intended use and identifies discrepancies from specifications and standards.

Source: Wikipedia, 2016

Review Process



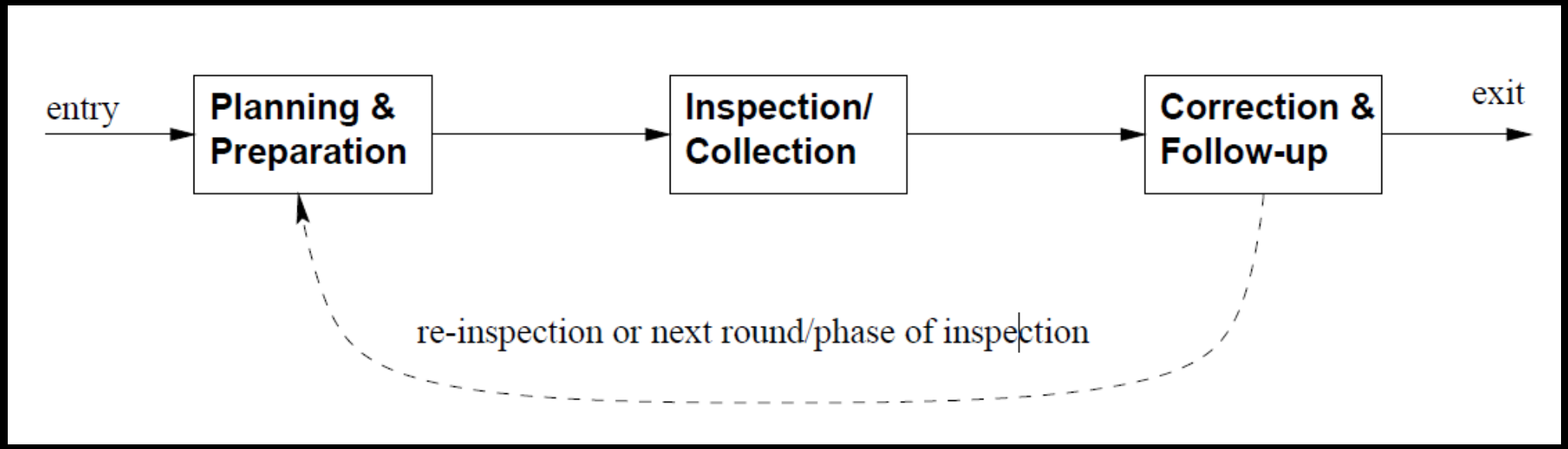
Reviews & Inspections in Agile Methods

- The review process in agile software development is usually informal.
  - In Scrum, for example, there is a review meeting after each iteration (Sprint) of the software has been completed (a sprint review), where quality issues and problems may be discussed.
- In extreme programming (XP), pair programming ensures that code is constantly being examined and reviewed by another team member; rely on team members cooperating to check each other's code with informal guidelines, such as 'check before check-in'
- XP relies on individuals taking the initiative to improve and refactor code.
- Agile approaches are not usually standards-driven, so issues of standards compliance are not usually considered.
- XP practitioners argue that pair programming is an effective substitute for inspection as this is, in effect, a continual inspection process.

Inspection as part of QA

- Performed throughout the software process
  - Coding phase: code inspection
  - Design phase: design inspection
  - Inspection in other phases and at transitions from one phase to another
- What to Inspect
  - Program code (typical)
  - Requirement/design/other documents (aka ‘Reviews’)
  - Charts/models/diagrams/tables/etc.
- Characteristics of Inspection
  - People dependent
  - Not waiting for implemented system.
  - Complementary to other QA activities.

Inspection Process



- Planning and preparation (individual)
- Collection (group/meeting)
- Repair (follow-up)

Fagan Inspection

[First introduced by Fagan at IBM, and subsequently lead to other variations]

- Six steps of Fagan inspection:
  1. Planning
  2. Overview (1-to-n meeting)
  3. Preparation (individual inspection)
  4. Inspection (n-to-n meeting)
  5. Rework
  6. Follow-up

Fagan Inspection (6 Steps)

1. **Planning**
   - Entry criteria: what to inspect, Team size: about 4 persons, Developers/testers from similar projects
   - Inspectors not authors
2. **Overview**
   - Author-inspectors meeting; General background information (functional/structural/info., intentions)
   - Assign individual tasks: coverage of important areas, moderate overlap,…
3. **Preparation or individual inspection**
   - Independent analysis/examination; Code as well as other document
   - Individual results: questions/guesses, potential defects
4. **Inspection (collection)**
   - Meeting to collect/consolidate individual inspection results; Team leader/meeting moderator
   - Reader/presenter: summarize/paraphrase
   - Defect identification, but not solutions, to ensure inspection effectiveness
   - No more than 2 hours; Inspection report
5. **Rework**
   - Author's response; Defect fixing (solutions)
6. **Follow-up**
   - Resolution verification by moderator; Re-inspection, if required

Program Inspection Check-List: An Example

Fault class	Inspection check
Data faults	• Are all program variables initialized before their values are used? • Have all constants been named? • Should the upper bound of arrays be equal to the size of the array or Size -1? • If character strings are used, is a delimiter explicitly assigned? • Is there any possibility of buffer overflow?
Control faults	• For each conditional statement, is the condition correct? • Is each loop certain to terminate? • Are compound statements correctly bracketed? • In case statements, are all possible cases accounted for? • If a break is required after each case in case statements, has it been included?
Input/output faults	• Are all input variables used? • Are all output variables assigned a value before they are output? • Can unexpected inputs cause corruption?
Interface faults	• Do all function and method calls have the correct number of parameters? • Do formal and actual parameter types match? • Are the parameters in the right order? • If components access shared memory, do they have the same model of the shared memory structure?

Other Inspection Methods…

(variations to Fagan Inspection in size/scope and formality

- Two-person inspection / Meetingless inspections
- Phased inspections / N-fold inspections (N independent teams)
- Informal check/review/walkthrough
- Active design reviews (Architecture/Design Docment)
- Inspection for program correctness
- Code reading / Code reading with stepwise abstraction

"Active Reviews for Intermediate Design (ARID) are an easy, lightweight evaluation approach for software architecture that concentrates on suitability and does not require complete documentation."

- http://www.sei.cmu.edu/architecture/tools/evaluate/arid.cfm

Code Reading: Formal Inspection

```
1    input(x);
2    if(x > 0) then
3        y ← x;
4    else
5        y ← −x;
6    output(y);
```

```
1        y ← x;
2    if(x > 0) then
3    else
4    output(y);
5        y ← −x;
6    input(x);
```

- **Code reading**
  - focus on code / optional meetings
- **Code reading by stepwise abstraction**
  - top-down decomposition and bottom-up abstraction

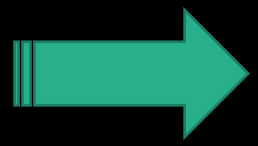
Reviews & Inspections: Summary

- Manually Intensive
- Wide applicability (across documents, reviews and Interfaces, …)
- Dynamic, complex problems and Interactions – Hard to track
- Hard to automate

Thank You

In the next session, we shall look at:

Quality Planning, Goals/Metrics,…



Software Quality Management

- K G Krishna

SQA: Fault-Tolerance Techniques ➔

A Check on SQA Vocabulary…

- Defect:: Focus of QA (Defect Elimination, Defect Removal, Defect Reduction)
- Defects contribute to Fault in the system (Lurking in the system, if not addressed may lead to undesired behaviour or major catastrophe/accident in case of real time systems)
- Fault Tolerance, Fault Containment (QA strategy for dealing with *hidden* defects)
- Hazard: Potential cause (condition) for accident (Hazard Analysis, Hazard Source Identification → Hazard Control/Reduction, Hazard Resolution
- Safety: The property of being accident-free for (embedded/RT) software systems
- Safety Engineering: Engineering (Design, Development) focusing on Safety considerations (fail-safe: failing in such a way not leading to major harm/accident)
- Accident: (post hazard scenario): Failures with severe consequences (Accident Analysis, Damage Reduction/Control, Safe/Escape route,…)

Fault tolerance as part of SQA

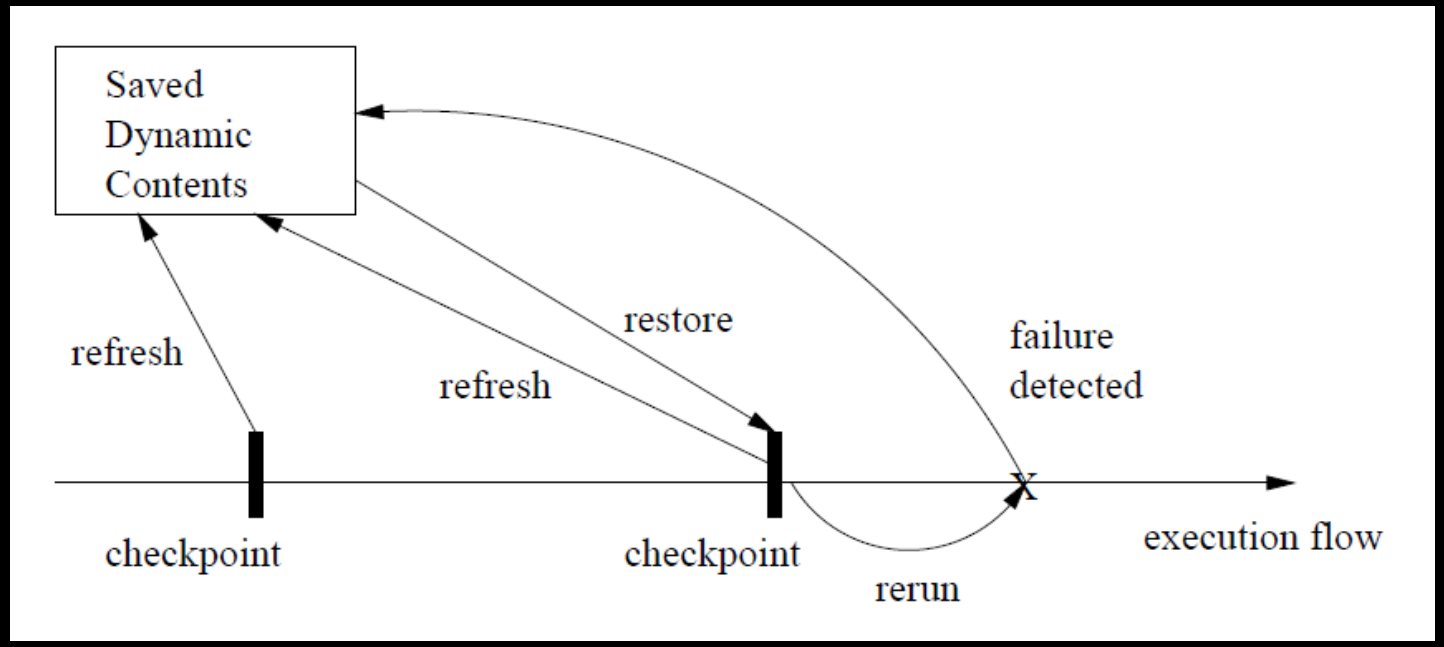
- Duplication: over time or components
- High cost, high reliability
- Run-time/dynamic focus
- Involves FT design and implementation
- Complementary to other QA activities

❖ **FT Philosophy**

- Local faults not lead to system failures
- Duplication/redundancy used
- redo ➔recovery block (RB)
- Parallel redundancy
  - N version programming (NVP)

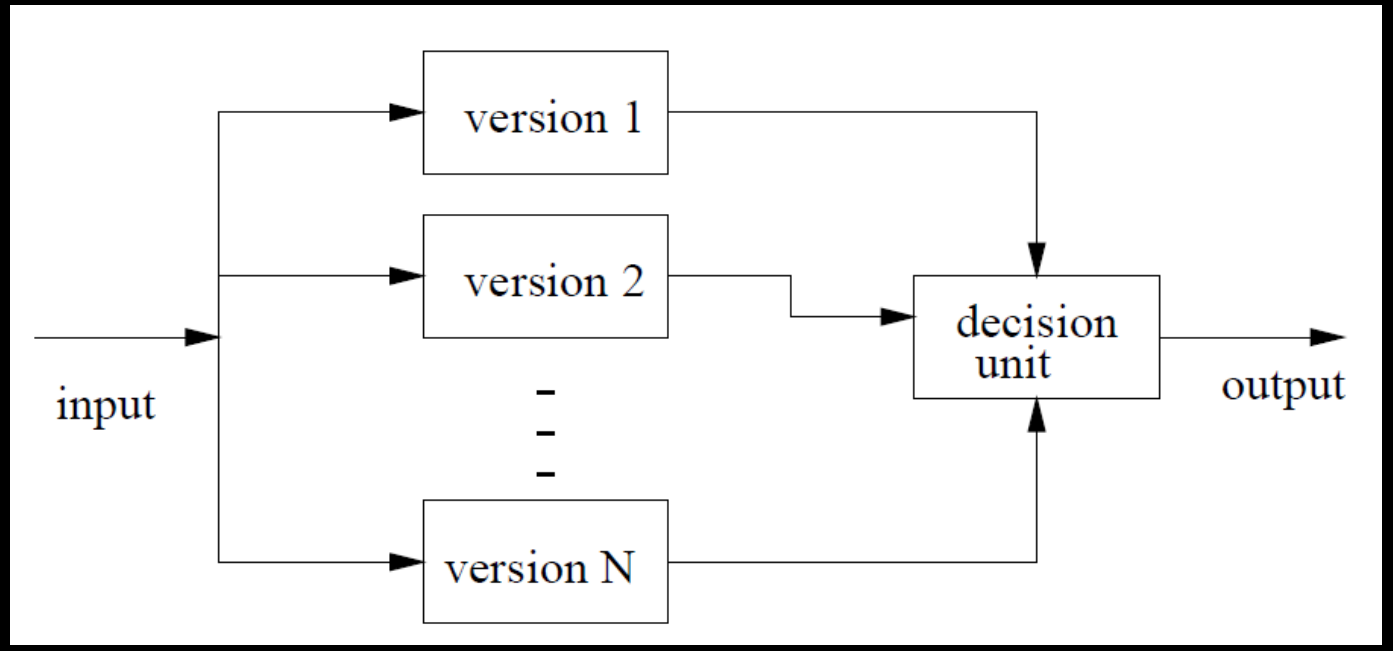
Fault Tolerance: Recovery Blocks

- Periodic checkpointing
  - too often: expensive checkpointing; too rare: expensive recovery; smart/incremental checkpointing
- Problem detection/acceptance test
  - exceptions due to in/external causes; periodic vs event-triggered
- Rollback (recovery)
  - external disturbance: environment?
  - internal faults: tolerate/correct?



Fault Toleranace: N-Version Programming (NVP)

- Multiple independent versions
  - Multiple: parallel vs backup?; How to ensure independence?
- Support environment:
  - Concurrent execution; switching; voting/decision algorithms
- Correction/recovery?
  - p-out-of-n reliability; in conjunction with RB



Implementing NVP

- Ways to ensure independence:
  - People diversity: type, background, training, teams, etc.
  - Process variations
  - Technology: methods/tools/PL/etc.
  - End result/product:
    - design diversity: high potential
    - implementation diversity: limited
- Ways to ensure design diversity:
  - People/teams
  - Algorithm/language/data structure
  - Software development methods
  - Tools and environments
  - Testing methods and tools (!)
  - Formal/near-formal specifications

Fault Tolerance in Safety (& Security)

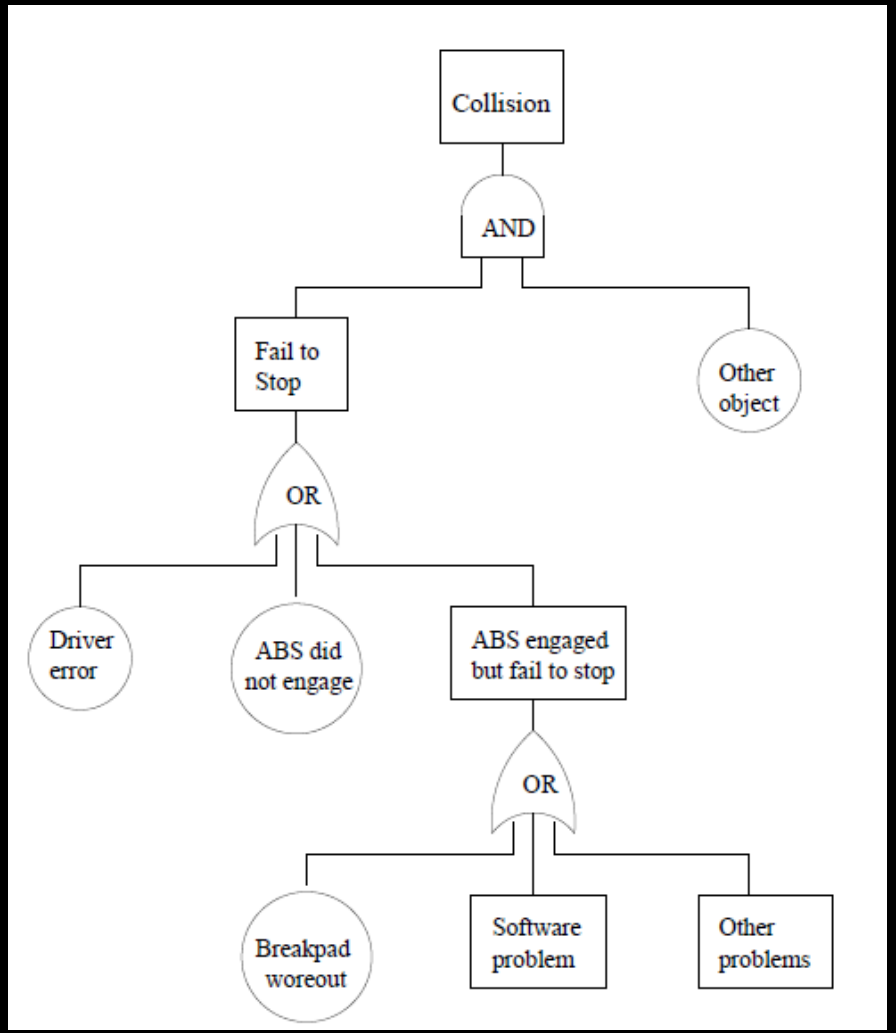
- Extending FT idea for safety:
  - FT: tolerate fault
  - Extend: tolerate failure
  - Safety: accident free
  - Weaken error-fault-failure-accident link
- FT in SSE (software safety engineering):
  - Too expensive for regular systems
  - As hazard reduction technique in SSE
  - Other related SSE techniques: general redundancy; substitution/choice of modules; barriers and locks; FT analysis
- Hacker-Proofing (Software Security Engineering)????
  (emerging trend to design software for security)

Analysis for Fault Tolerance (FTA)

- Safety Engineering: Hazard Analysis
  - Ex: FTA *(Fault Tree Analysis)* of Automobile Accident

FTA construction:

- Starts with top event/accident (backward search)
- Decomposition of events or conditions
- Stop when further development not required or not possible (atomic)
- Focus on controllable events/elements

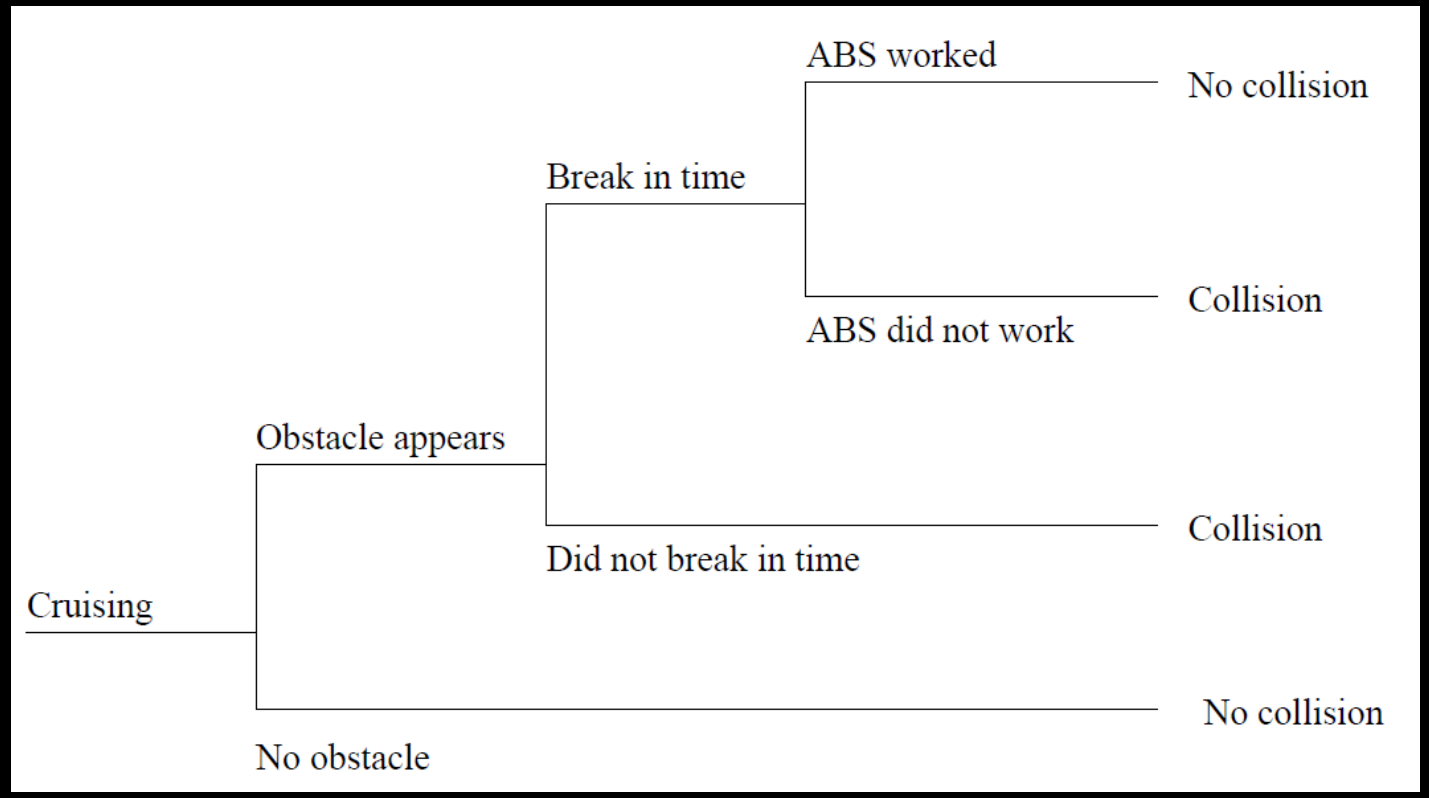


Safety Engineering: Hazard Analysis

ETA *(Event Tree Analysis)* of Automobile Accident

Event tree analysis (forward search)

- Recreate accident sequence/scenario
- Critical path analysis
- Used in hazard resolution (esp. in hazard reduction/control, e.g. creating barriers, isolation and containment



Safety Engineering: Using FTA vs ETA

❖ Using FTA:

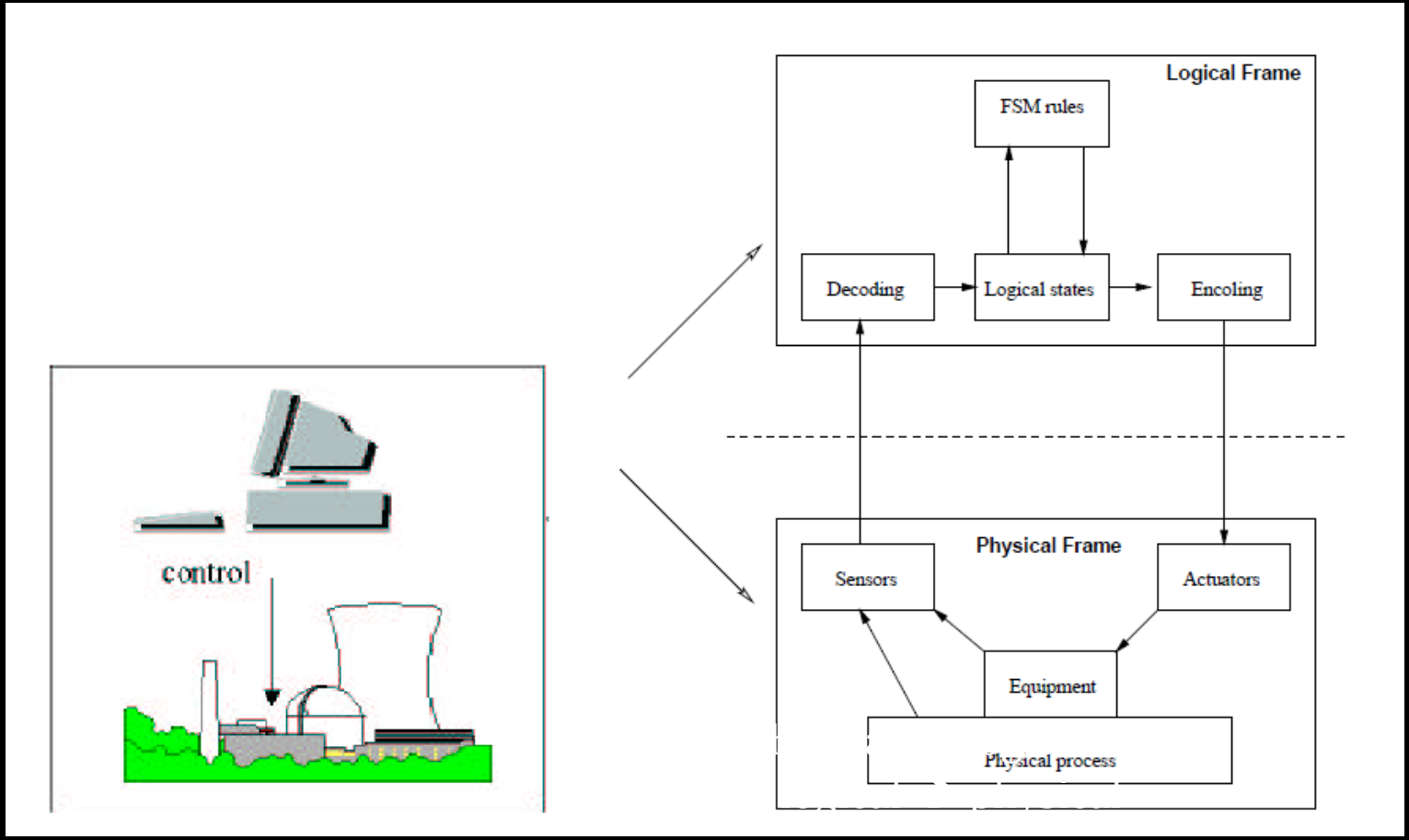
- Hazard identification
  - logical composition (vs. temporal composition in ETA)
- Hazard resolution
  - component replacement etc.; focused safety verification; negate logical relation

❖ When to Use ETA

- (FTA focusses on static analysis; static logical conditions) vs. Dynamic aspect of accidents
- Timing and temporal relations
- In Real-time control systems
- ETA provide additional Info.
  - Used in Hazard Resolution (in hazard reduction/control as in creating barriers, isolation and containment)

Software Safety: TFM (Two Frame Model)

- physical frame: nuclear reactor
- logical frame: computer controller



When Do We Use TFM

[Failure/Hazard Sources & Scenario Analysis]

❖TFM characteristics:

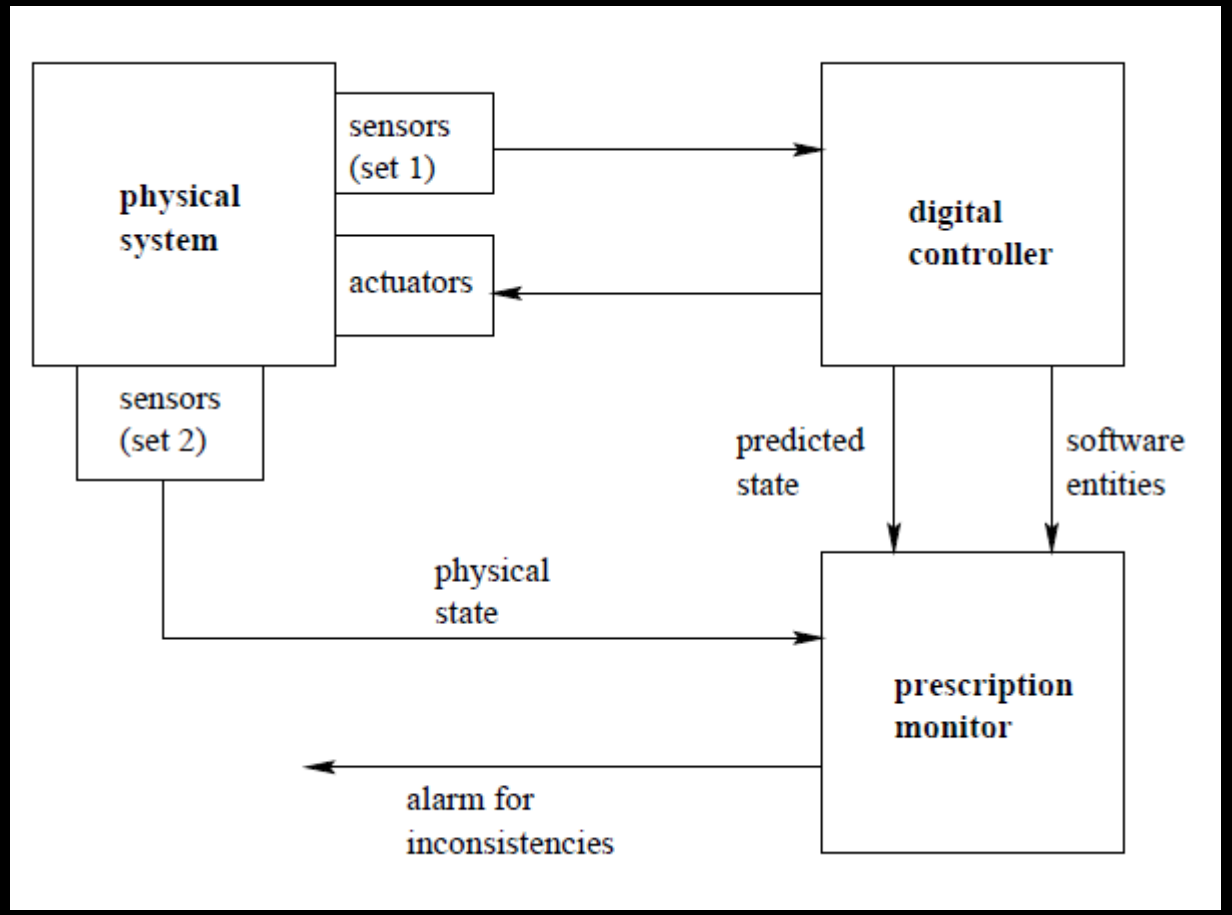
- Interaction between the two frames
- Nondeterministic state transitions and encoding/decoding functions
- Focuses on symmetry/consistency between the two frames.

❖Usage of TFM

- Hardware/equipment failures
- Software failures
- Communication/interface failures
  - Causes of communication/interface hazards:
    - Inconsistency between frames.
    - Sources of inconsistencies

Fault Simulation: Prescription Monitor

- Inconsistencies detected by PM by Fault Seeding
  - Erroneous user input
  - Wrong data types or values
  - Programming errors
  - Wrong reading of sensors



Software Fault Tolerance: Summary

❖ Software Fault Tolerance

- Duplication and redundancy.
- Techniques: RB, NVP, and variations.
- Cost effectiveness concerns.

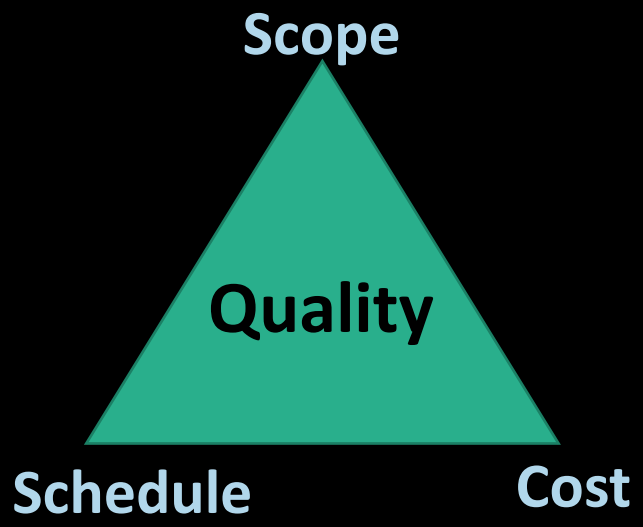
❖ Software Safety Engineering

- Analysis to identify hazard
- Design for safety
- Safety constraints and verification
- Cost and application concerns.

Thank You

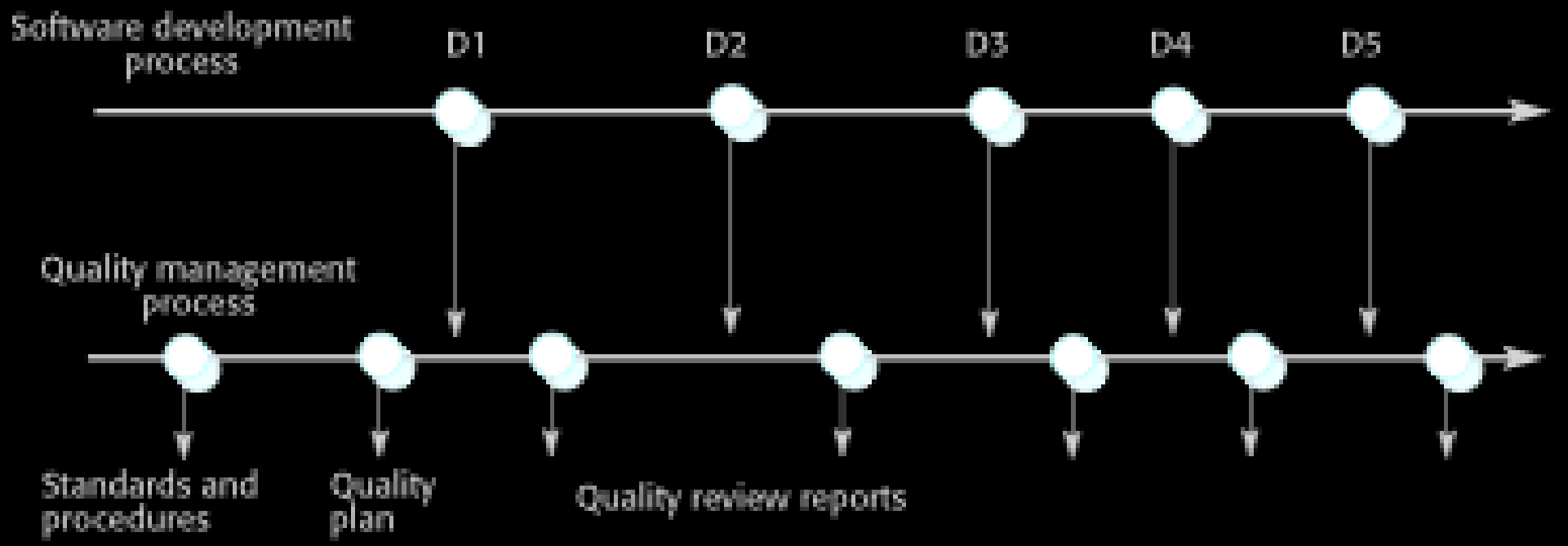
Software Quality Management

- K G Krishna



Quality Planning in Projects ➔

Software Development vs Quality Planning



Quality Planning

- A quality plan sets out the desired product qualities and how these are assessed and defines the most significant quality attributes.
- The quality plan should define the quality assessment process.
- It should set out which organisational standards should be applied and, where necessary, define new standards to be used.
- Quality Plan Structure
  - Product introduction; Product plans; Process descriptions; Quality goals
  - Risks and risk management
  - …
- Quality plans should be short, succinct documents
  - Should be a living document accessible to the Project Team for dynamic updates whenever required

Desired Quality Attributes (Goals)
(To be explicitly documented in Quality Plan)

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Quality (Project) Goals

- Answering the question '*What do we have to do to have a success?*'
- *Informally*, the objective of a project can be defined by completing the statement:

  The project will be regarded as a success if……. & ……

- Focus on *what* will be put in place, rather than *how* activities will be carried out

Product or Process Standards
(To be followed in the Project)

- Standards define the required attributes of a product or process.
- Standards may be international, national, organizational or project standards.
- Product standards define characteristics that all software components should exhibit e.g. a common programming style.
- Process standards define how the software process should be implemented
- Encapsulation of best practice/lessons-learnt- avoids repetition of past mistakes.
- They are a framework for defining what quality means in a particular setting i.e. organization's view of quality.
- They provide continuity - new staff can understand the organisation by understanding the standards that are used.

Product & Process Standards

Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of new code for system building
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

Quality Management System (QMS) Standards

- ✓ ISO 9001:2000 - CMMI (SQA Model)
- ✓ ISO / IEC 15504 (Assessment Model)
- ✓ ISO 12207 (Process Model)

Quality Planning in Projects

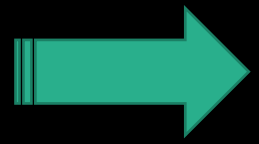
(Template of Quality Plan Document)

IEEE Std SQA Plan

Thank You

In the next session, we shall look at:

Defining Metrics & Baselining



"What We Can't Measure, We Can't Manage"

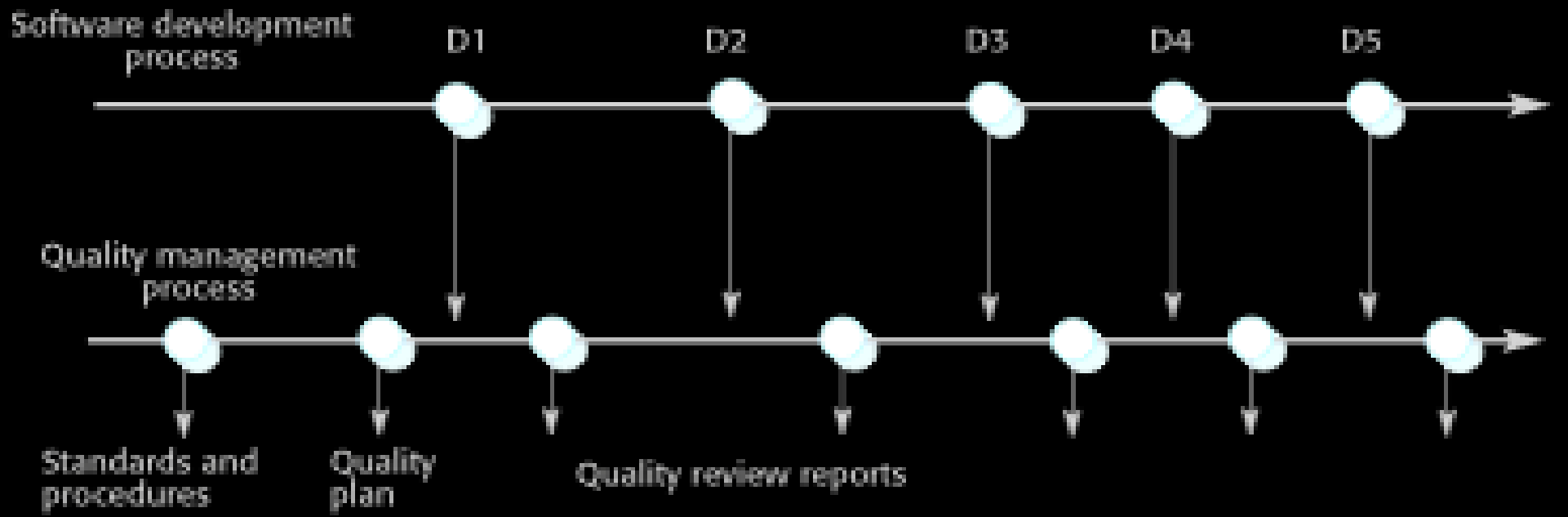
Software Quality Management

- K G Krishna

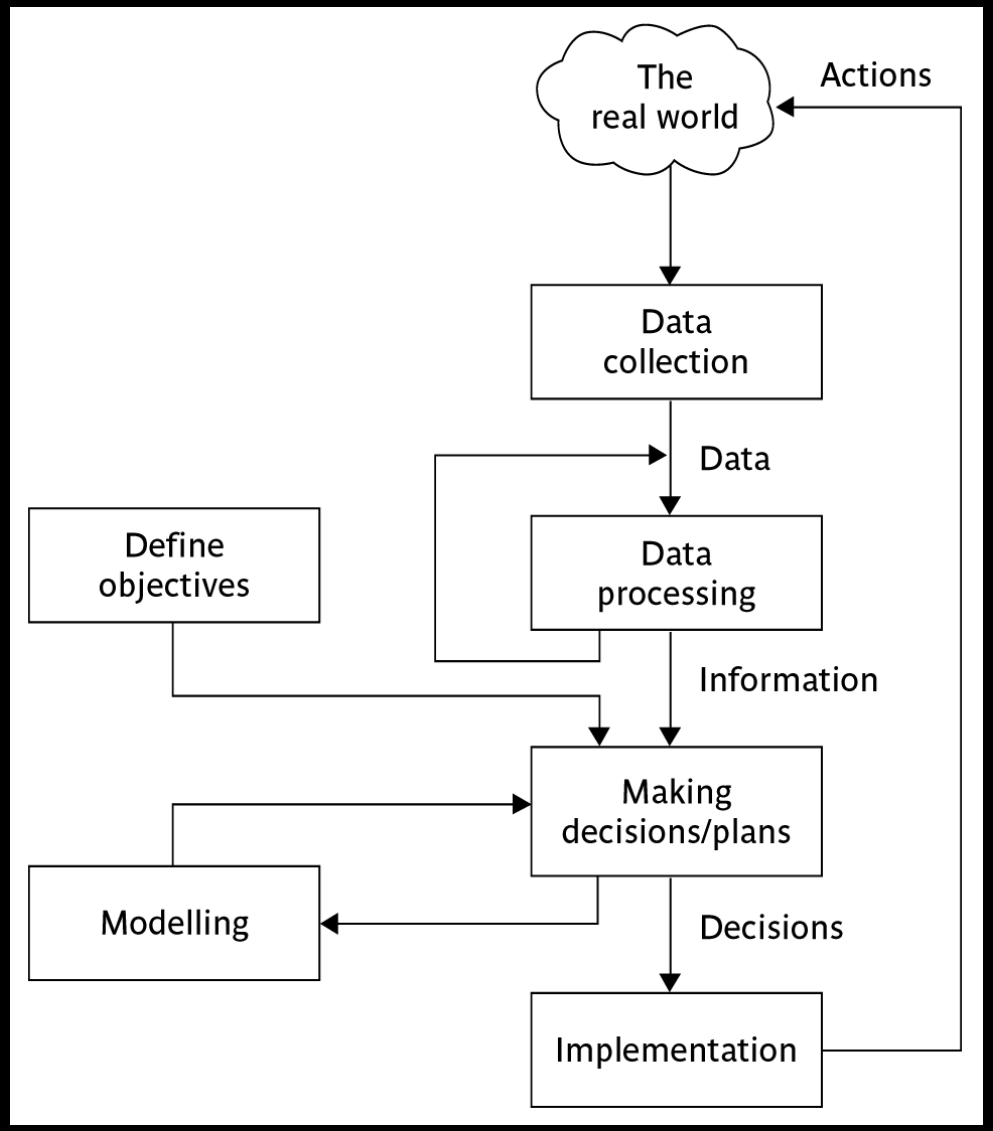
Defining Quality Metrics & Baselining ➔

Project / Quality Management

“To Measure = To Control = To Manage”



“What We Can’t Measure, We Can’t Manage”



Source courtesy: Software project management (5e) - introduction © The McGraw-Hill Companies, 2009

Metrics for Management Control

Data – the raw details

e.g. *'6,000 documents processed at location X'*

Information – the data is processed to produce something meaningful and useful

e.g. 'productivity is 100 documents a day'

Comparison with objectives/goals

e.g. we will not meet target of processing all documents by 31st March

Modelling – working out the probable outcomes of various decisions

e.g. if we employ two more staff at location X how quickly can we get the documents processed?

Implementation – carrying out the remedial actions that have been decided upon

Defining Metrics
(*for Performance Assessment & Tracking*)

- ✓ **S - Specific** (or Significant) – metrics are specific and targeted to area being measured. For example, What or who is involved? What do I want to accomplish? What are the reasons, purpose or benefits? What are the requirements and constraints? Developing too many metrics can lead to excessive overhead and inefficiencies.
- ✓ **M - Measurable** (or Manageable, Meaningful) – metrics can be collected that are accurate and complete. Is this something we can really measure?
- ✓ **A - Achievable** (or Appropriate, Attainable) – desired metrics can be readily measured. Answers this question, "How can this be accomplished?" Metrics should not be developed if one cannot collect accurate or complete data.
- ✓ **R - Relevant** (or Realistic, Results-Oriented) – things which are not important are not measured. For example, Is this worthwhile? Does this match our needs? Metrics should be of sufficient complexity to be relevant but simple enough to comprehend.
- ✓ **T - Timely** (or Tangible, Tractable) - metrics are those for which data are available when needed. When are these metrics needed?

Elements of Metrics

- Objective - goal or objective that the metric supports (for example, business process).
- Description - what the metric is about; why the metric is useful.
- Data Source - where the data was obtained (and often by what means); how the metric is calculated; quality of the data.
- Metric Owner - individual responsible for providing the metric.
- Reporting Period - period for which the metric is applicable or relevant
- Reporting Frequency - frequency in which the metric is measured
- Present Value - current or present value of the metric as most recently measured.
- Previous Value - value of the metric from the last measurement period.
- Baseline - normally expected value for a metric (an average for example); may also include lowest and highest possible value for the metric.
- Trend - direction the metric is moving; should indicate is increased or decreased values are better and the progress (improving, getting worse, etc.)
- Thresholds - expected ranges of the metrics; this determines where the metric is with respect to target value or values (determines acceptable values for "green" status) and danger values or values (determines unacceptable values; generally the delineation between "yellow" and "red" status).

Courtesy: https://confluence.cornell.edu/display/metrics/Successful+Metrics

Metrics – Related Terms/Acronyms

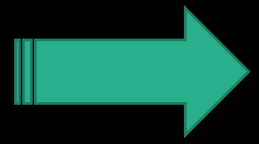
Term	Acronym	Definition
Balanced Scorecard	**BSC**	Report often used to keep track of the performance of specific activities with focus on performance objectives and performance targets.
Key Performance Indicator	**KPI**	Metrics which are used to measure the achievement of each critical success factor. Should be relevant to a specific objective or business process. For example, average incident response time, number of repeated incidents, incidents completed without escalation.
Lagging Indicator		Metric which refers to past events. For example, number of incidents resolved by the Service Desk within 24 hours.
Leading Indicator		Metric which refers to future events. For example, number of currently open incidents.
Critical Success Factor	**CSF**	Factor that represents something must happen to achieve success. For example, effectiveness of Service Desk operations.
Metric		A standard, reportable measure and reported to assess performance in a particular area.
Dashboard		A mechanism (generally graphical) to provide at a-glance views of key performance indicators.

Courtesy: https://confluence.cornell.edu/display/metrics/Successful+Metrics

Thank You

In the next session, we shall look at:

Software Product Metrics & Defect Propagation



Software Quality Management

- K G Krishna

Software Product Metrics & Defect Propagation ➔

Metrics - A Check on Vocabulary…

- Measure
- Indicator: "anything that enables prediction of future (trends)…" (mostly used in Finance and Business)
- *Lead Indicator*: "signal future events " (like yellow-light before coming green-light, though not as accurate as that)
- *Lagging Indicator*: "follows an event (like yellow-light is lagging indicator for green because it tails green-light; uemployment rate is a lagging indicator of economy because if the economy is poor, it leads to unemployment)
- Index, Metric
- Metrics: Project vs. Process vs. Product
- Metrics Dabatabase
- Benchmarking: Processes & Products

“Measure vs. Metrics”

(multiple interpretations?)

- Measurement - A measure (rating, sizing, etc.) of one thing. Examples are cost, function points, effort, time, etc.
- Metric - A result of taking two or more measurements to create a value. Using the measurement examples, you can get cost per function point, function points per unit of time, etc.

Basically, the measurement is something you need to create a metric that can then be used for reporting and analysing. A measurement generally will not provide much value or meaning until it is combined with other measurements. The metrics provide the value.

Software Measurement and Metrics

- Software measurement is concerned with deriving a numeric value for an attribute of a software product or process.
- This allows for objective comparisons between techniques and processes.
- Although some companies have introduced measurement programmes, most organisations still don’t make systematic use of software measurement.
- There are few established standards in this area.

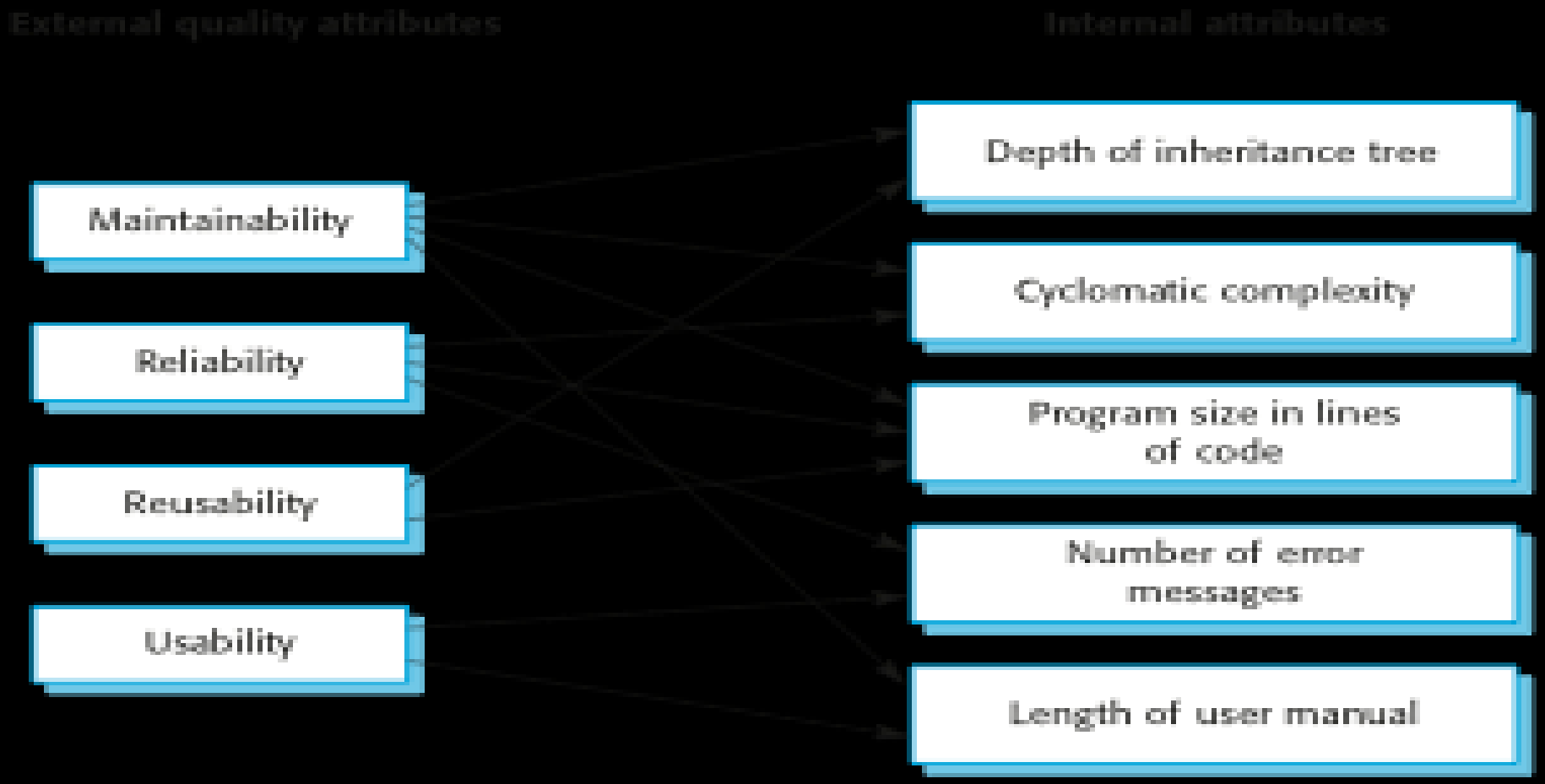
Software Metric

- Any type of measurement which relates to a software system, process or related documentation
  - Lines of code in a program, the Fog index, number of person-days required to develop a component.
- Allow the software and the software process to be quantified.
- May be used to predict product attributes or to control the software process.
- Product metrics can be used for general predictions or to identify anomalous components.

Metrics assumptions

- A software property can be measured.
- The relationship exists between what we can measure and what we want to know. We can only measure internal attributes but are often more interested in external software attributes.
- This relationship has been formalised and validated.
- It may be difficult to relate what can be measured to desirable external quality attributes.

Relationships between Internal and External software attributes



Problems with Measurement in IT industry

- It is impossible to quantify the return on investment of introducing an organizational metrics program.
- There are no standards for software metrics or standardized processes for measurement and analysis.
- In many companies, software processes are not standardized and are poorly defined and controlled.
- Most work on software measurement has focused on code-based metrics and plan-driven development processes. However, more and more software is now developed by configuring ERP systems or COTS or using Agile Methods
- Introducing measurement adds additional overhead to processes.

Product metrics

- A quality metric should be a predictor of product quality.
- Classes of product metric
  - Dynamic metrics which are collected by measurements made of a program in execution; Dynamic metrics are closely related to software quality attributes; It is relatively easy to measure the response time of a system (performance attribute) or the number of failures (reliability attribute).
  - Dynamic metrics help assess efficiency and reliability

  - Static metrics which are collected by measurements made of the system representations; Static metrics have an indirect relationship with quality attributes; You need to try and derive a relationship between these metrics and properties such as complexity, understandability and maintainability.
  - Static metrics help assess complexity, understandability and maintainability.

Static software Product Metrics

Software metric	Description
Fan-in/Fan-out	Fan-in is a measure of the number of functions or methods that call another function or method (say X). Fan-out is the number of functions that are called by function X. A high value for fan-in means that X is tightly coupled to the rest of the design and changes to X will have extensive knock-on effects. A high value for fan-out suggests that the overall complexity of X may be high because of the complexity of the control logic needed to coordinate the called components.
Length of code	This is a measure of the size of a program. Generally, the larger the size of the code of a component, the more complex and error-prone that component is likely to be. Length of code has been shown to be one of the most reliable metrics for predicting error-proneness in components.
Cyclomatic complexity	This is a measure of the control complexity of a program. This control complexity may be related to program understandability.
Length of identifiers	This is a measure of the average length of identifiers (names for variables, classes, methods, etc.) in a program. The longer the identifiers, the more likely they are to be meaningful and hence the more understandable the program.
Depth of conditional nesting	This is a measure of the depth of nesting of if-statements in a program. Deeply nested if-statements are hard to understand and potentially error-prone.
Fog index	This is a measure of the average length of words and sentences in documents. The higher the value of a document's Fog index, the more difficult the document is to understand.

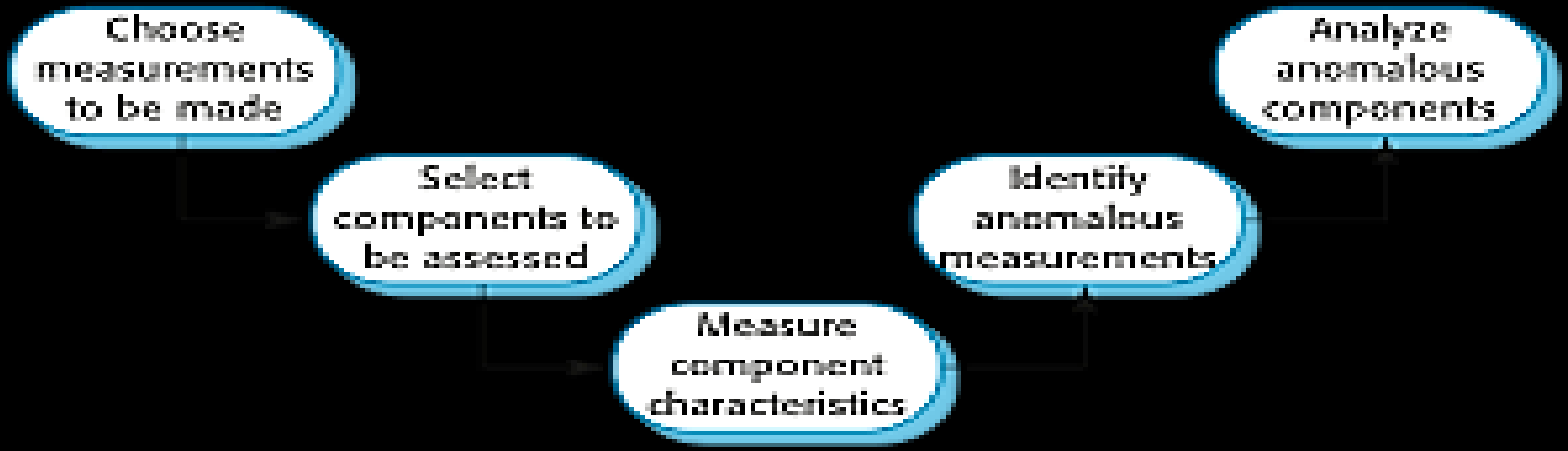
The Object-Oriented Metrics suite

Object-oriented metric	Description
Weighted methods per class (WMC)	This is the number of methods in each class, weighted by the complexity of each method. Therefore, a simple method may have a complexity of 1, and a large and complex method a much higher value. The larger the value for this metric, the more complex the object class. Complex objects are more likely to be difficult to understand. They may not be logically cohesive, so cannot be reused effectively as superclasses in an inheritance tree.
Depth of inheritance tree (DIT)	This represents the number of discrete levels in the inheritance tree where subclasses inherit attributes and operations (methods) from superclasses. The deeper the inheritance tree, the more complex the design. Many object classes may have to be understood to understand the object classes at the leaves of the tree.
Number of children (NOC)	This is a measure of the number of immediate subclasses in a class. It measures the breadth of a class hierarchy, whereas DIT measures its depth. A high value for NOC may indicate greater reuse. It may mean that more effort should be made in validating base classes because of the number of subclasses that depend on them.

Software Component Analysis

- System component can be analyzed separately using a range of metrics.
- The values of these metrics may then compared for different components and, perhaps, with historical measurement data collected on previous projects.
- Anomalous measurements, which deviate significantly from the norm, may imply that there are problems with the quality of these components.

The Process of Product Measurement



Defect Propagation & Defect Removal Efficiency (DRE)

- DRE is useful to measure the effectiveness of a particular test such as acceptance, unit, or system testing.

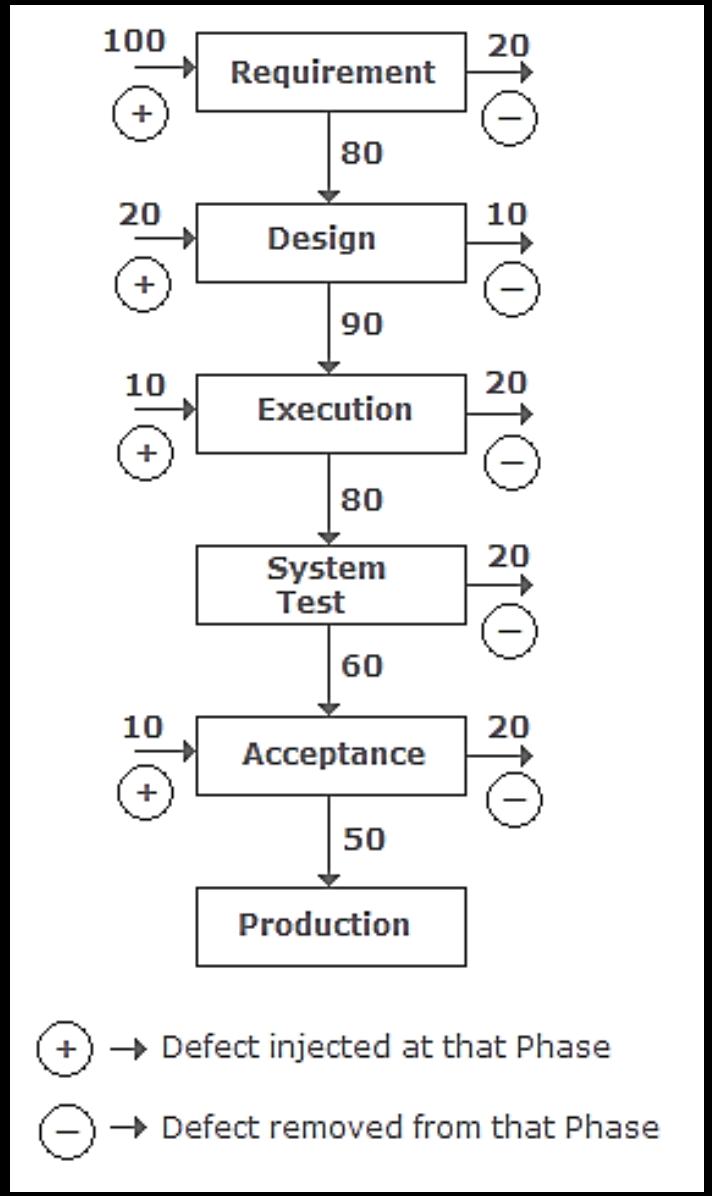
$$DRE = \frac{\text{Number of Bugs while Testing}}{\text{Number of Bugs while Testing} + \text{Number of Bugs found by User}}$$

Number of Bugs while Testing = Bugs found during Development process while testing it

Number of Bugs found by User = These are the Defect detected by the end user

The success of DRE depends on several factors.

- Severity and distribution of bugs must be taken into account.
- Second, how do we confirm when the customer has found all the bugs. This is normally achieved by looking at the history of the customer.



Source courtesy: http://www.indiabix.com/technical/software-testing/metrics/

Defect Propagation (spillage) & ‘spoilage’

Phase Created	Requirement	Design	Execution	Testing	Production
Requirement	0	1	2	3	4
Design	0	0	1	2	3
Execution	0	0	0	1	2

- Defect age is also called a phase age or phage. One of the most important things to remember in testing is that the later we find a defect the more it costs to fix it. Defect age and defect spoilage metrics work with the same fundamental, i.e., how late you found the defect. So the first thing we need to define is what is the scale of the defect age according to phases. For instance, the following table defines the scale according to phases. So, for instance, requirement defects, if found in the design phase, have a scale of 1, and the same defect, if propagated until the production phase, goes up to a scale of 4.

$$\text{Spoilage} = \frac{\text{Sum of number of Defects x Discovered Phase}}{\text{Total number of Defects}}$$

Thank You

In the next session, we shall look at:

Software Quality Management Systems ➔

Software Quality Management

- K G Krishna

Quality Management Systems ➔

Overview of

ISO 9001:2000 & CMMI (SQA Model)
ISO / IEC 15504 (Assessment Model)
ISO 12207 (Process Model)

ISO 9001:2000 Quality Management System (QMS)

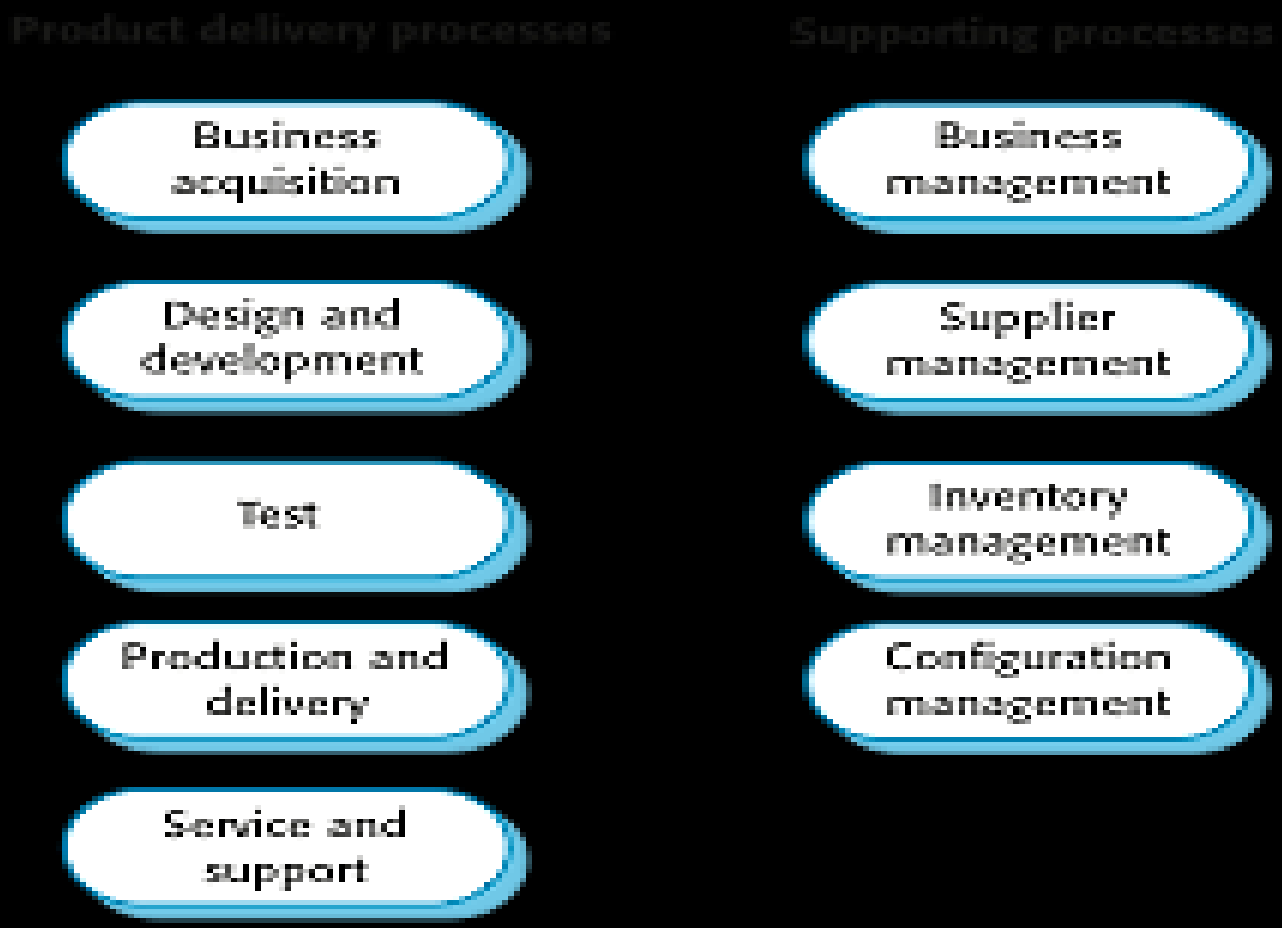
- ISO 9001 is one of a family of standards that specify the characteristics of a good quality management system (QMS)
- Can be applied to the creation of any type of product or service, not just IT and software
- Does NOT set universal product/service standards
- DOES specify the way in which standards are established and monitored

ISO 9001:2000 Principles

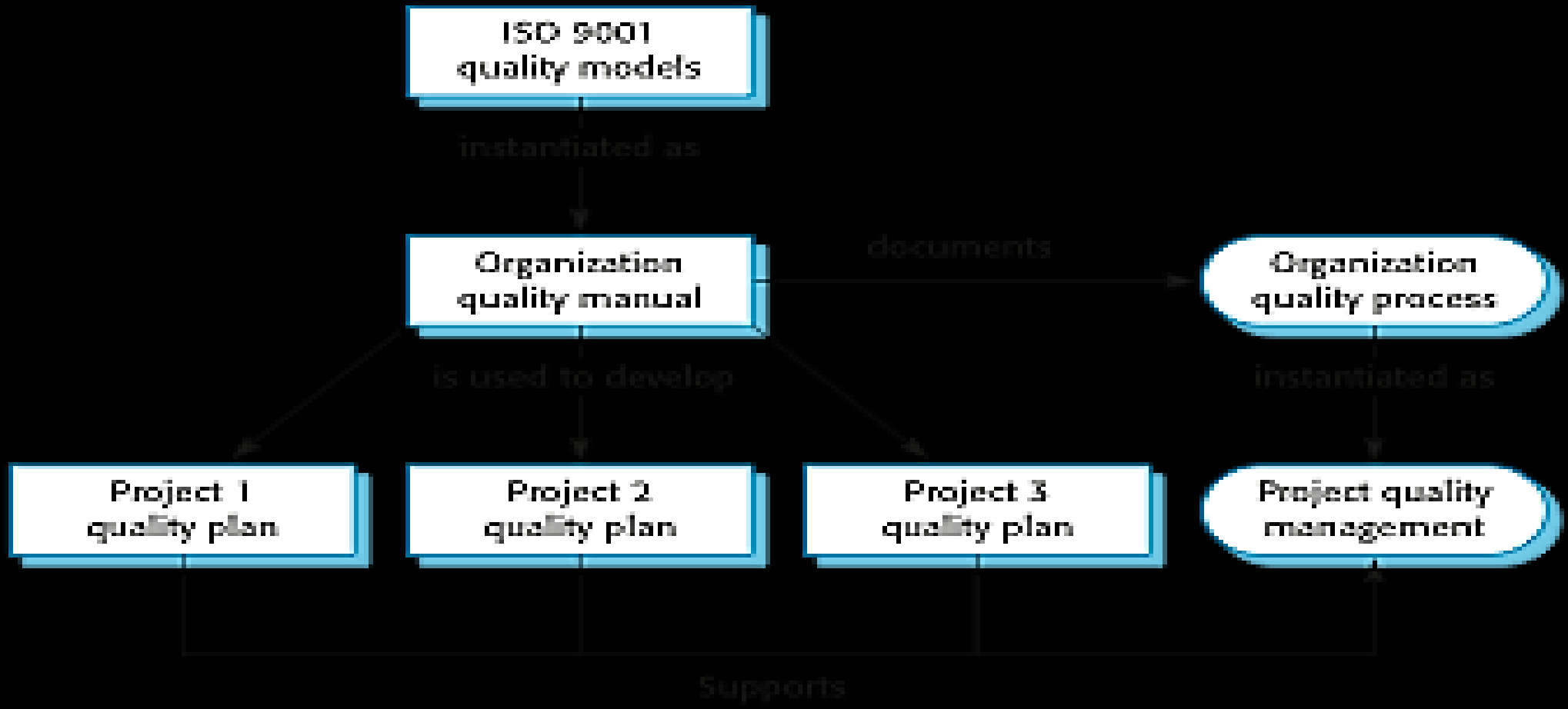
1. Understanding the requirements of the customer
2. Leadership to provide unity of purpose and direction to achieve quality
3. Involvement of staff at all levels
4. Focus on the individual sub-process which create intermediate and deliverable products and services
5. Focus on interrelation of processes that deliver products and services
6. Continuous process improvement
7. Decision-making based on factual evidence
8. Mutually beneficial relationships with suppliers

.

ISO 9001 Core Processes



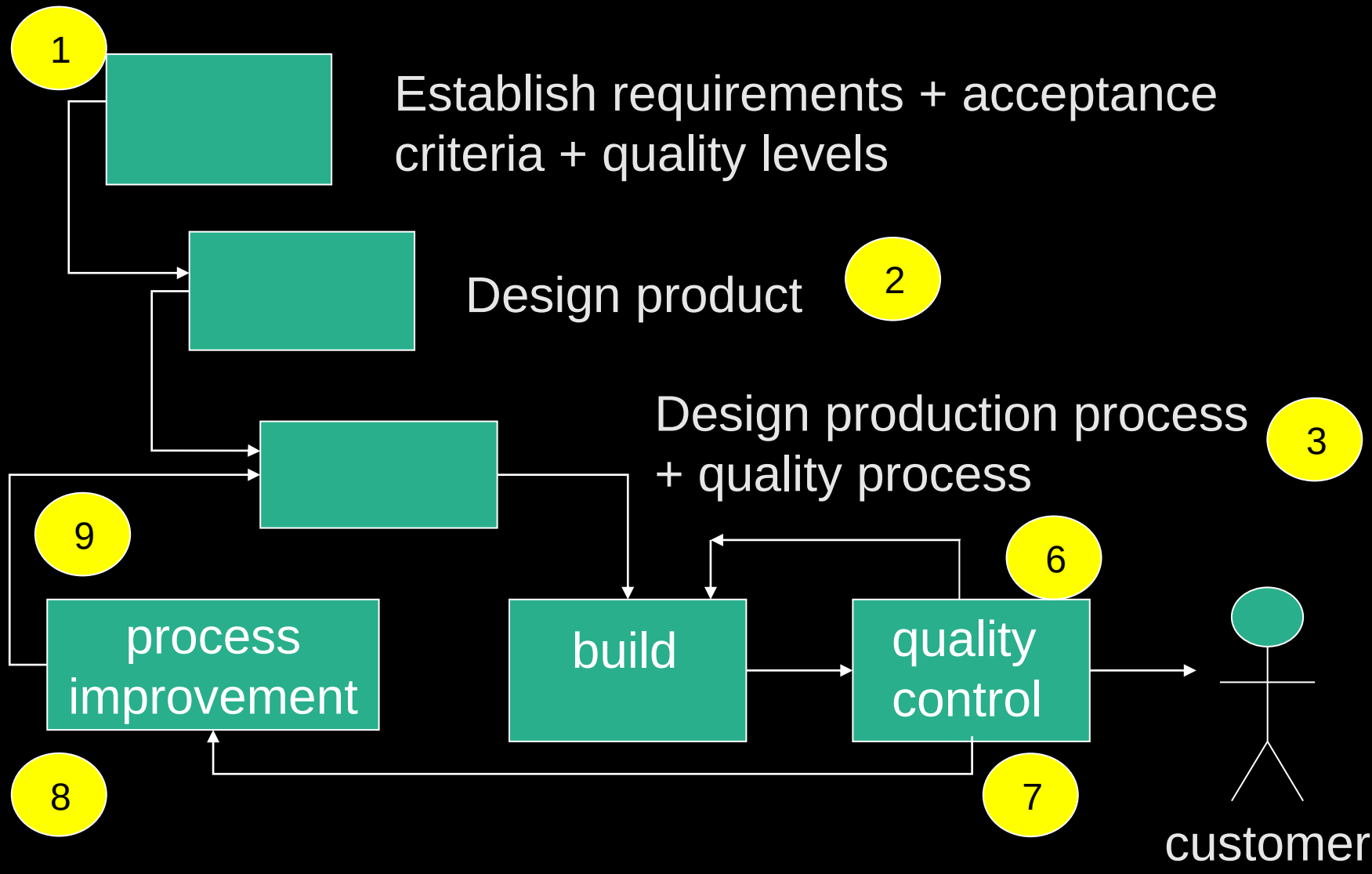
ISO 9001 and Quality Management



ISO 9001 Certification

- Quality standards and procedures should be documented in an organisational quality manual.
- An external body may certify that an organisation's quality manual conforms to ISO 9000 standards.
- Some customers require suppliers to be ISO 9000 certified although the need for flexibility here is increasingly recognised.

ISO 9001:2000 Development Cycle

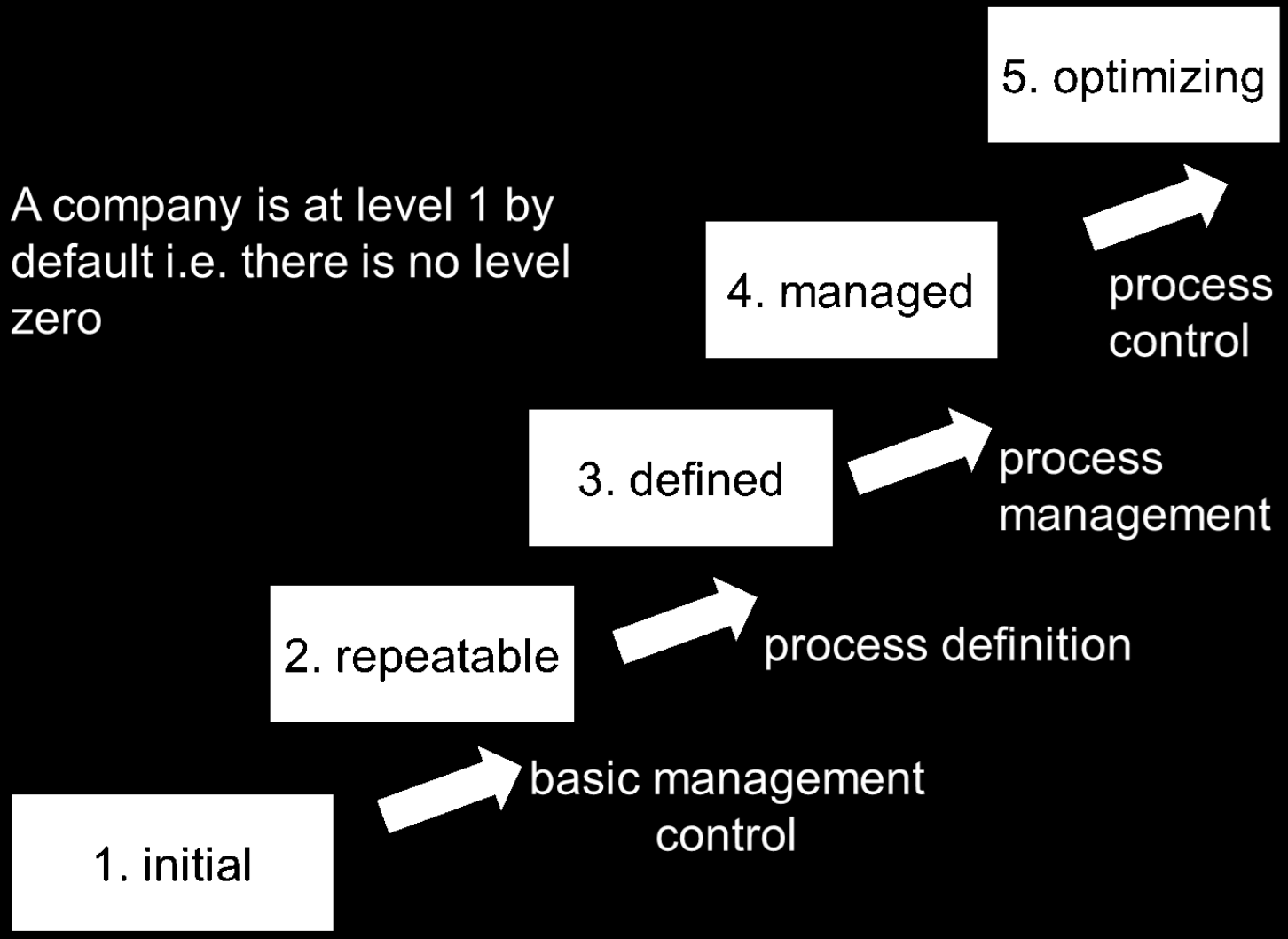


CMMI: Capability Maturity Model *Integrated*

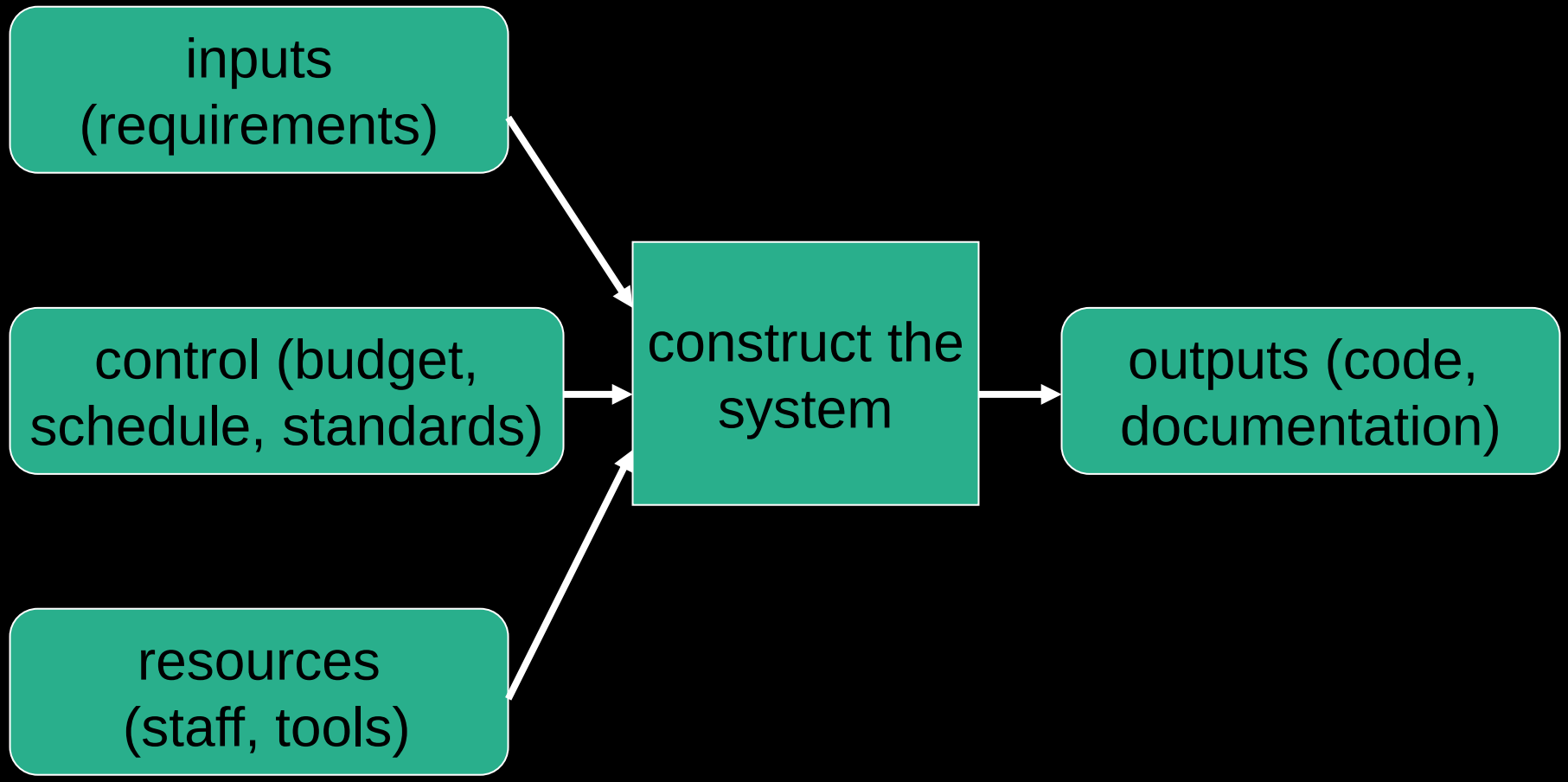
- Created by Software Engineering Institute, Carnegie Mellon University
- CMM developed by SEI for US government to help procurement
- Ref: Watts S. Humphrey ‘Managing the software process’ Addison Wesley
- Assessment is by questionnaire and interview
- Different versions have been developed for different environments e.g. software engineering
- New version CMMI tries to set up a generic model which can be populated differently for different environments

CMMI: Process Maturity Levels

A company is at level 1 by default i.e. there is no level zero



CMMI: A Repeatable model

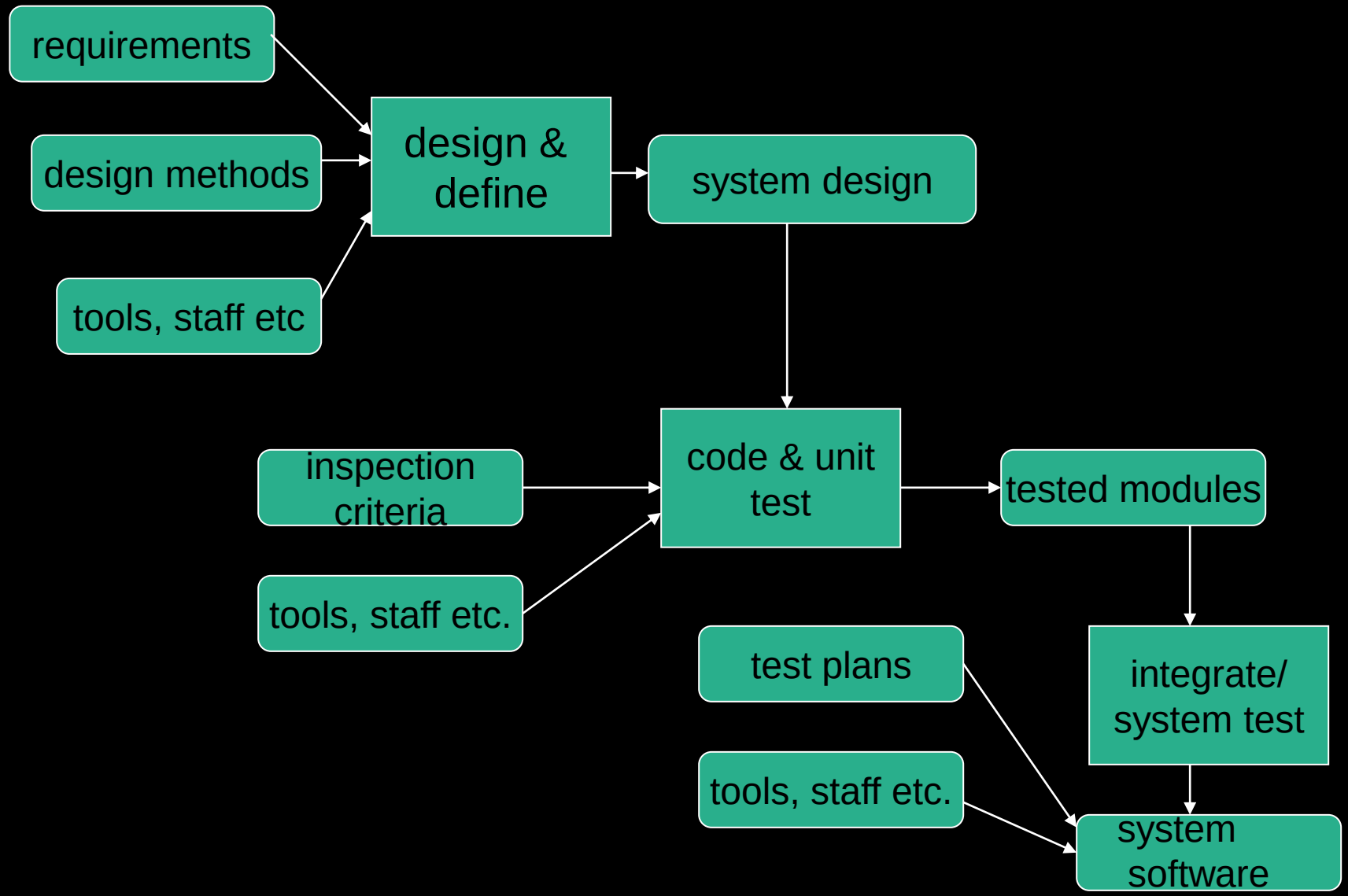


CMMI: Repeatable Model KPAs

To move to this level focus to be on:

- Configuration management
- Quality assurance
- Sub-contract management
- Project planning
- Project tracking and oversight
- Measurement and analysis

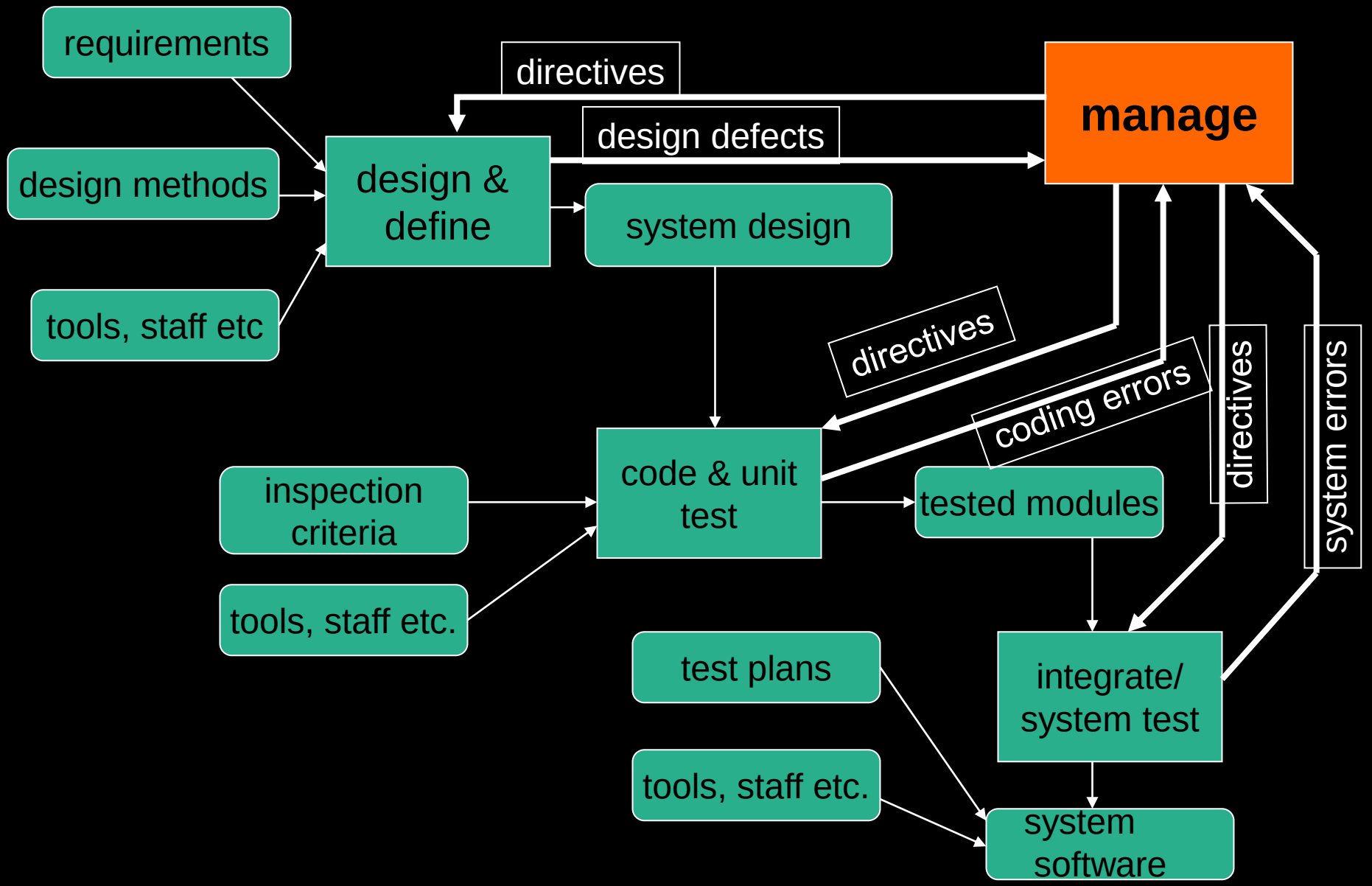
CMMI: A Defined Process



CMMI: Repeatable to Defined KPAs

- Requirements development and technical solution
- Verification and validation
- Product integration
- Risk management
- Organizational training
- Organizational process focus (function)
- Decision analysis and resolution
- Process definition
- Integrated project management

CMMI: a Managed Process

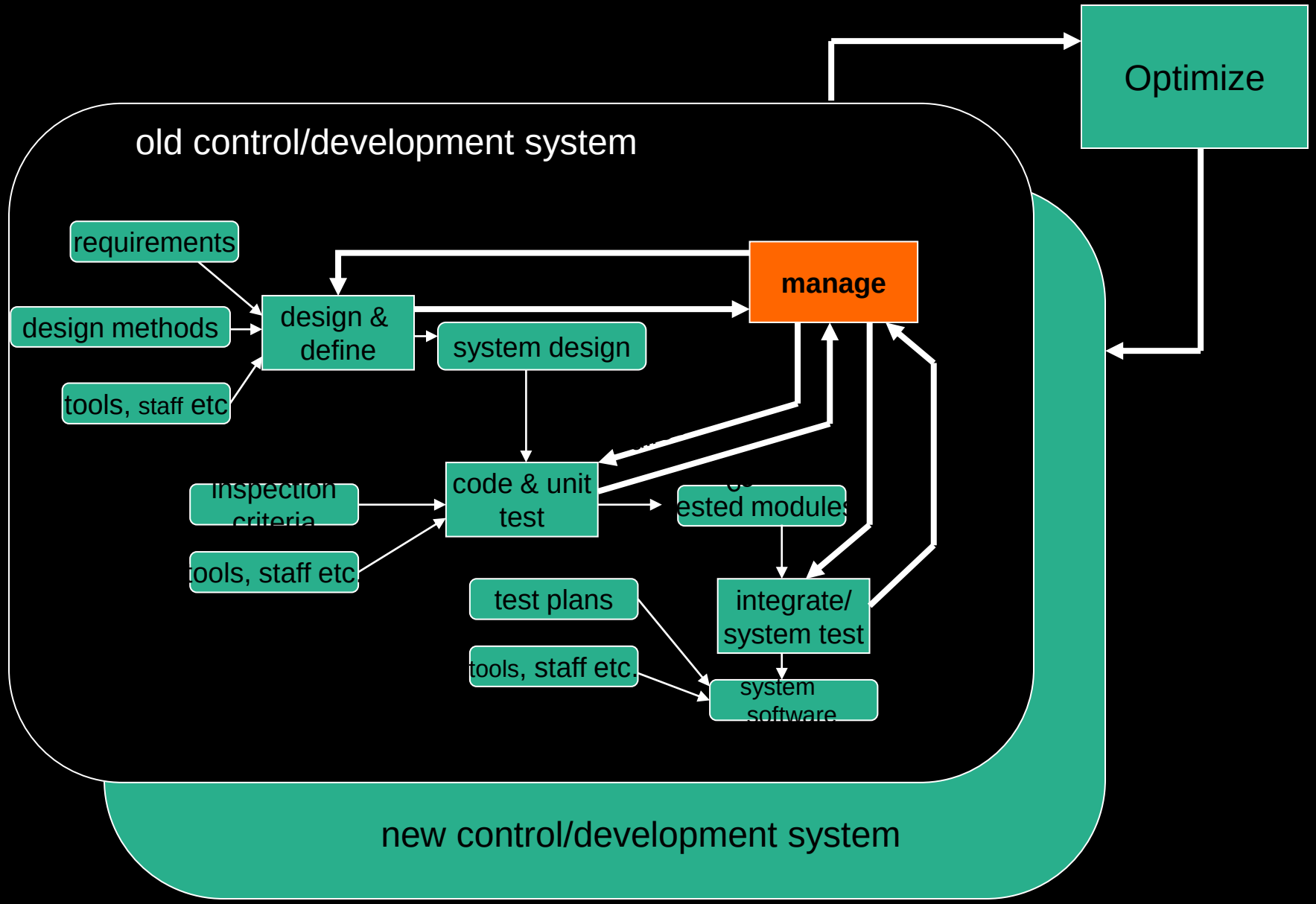


CMMI: Defined to Managed KPAs

Focus on:

- Organizational process performance
- Quantitative project management

CMMI: Level-5 Optimizing



CMMI: Managing to Optimizing KPAs

Focus on:

- Causal analysis and resolution
- Organizational innovation and deployment

Some Questions about CMMI

- suitable only for large organizations?
  - e.g. need for special quality assurance and process improvement groups
- defining processes may not be easy with new technology
  - how can we plan when we've not used the the development method before?
- Higher CMM levels easier with maintenance environments?
- can you jump levels? *("Direct Jump to level 5 in India")*

ISO/IEC 15504 IT Process Assessment

- To provide guidance on the assessment of software development processes
- ISO12207 (Process Reference Model) Defined set of processes that represent good practice to be the benchmark
  - A defined standard approach to development
  - Reflects recognized good practice
  - A benchmark against which the processes to be assessed can be judged

Software Quality Management: Summary

- Software quality management is concerned with ensuring that software has a low number of defects and that it reaches the required standards of maintainability, reliability, portability and so on.
- SQM includes defining standards for processes and products and establishing processes to check that these standards have been followed.
- Software standards are important for quality assurance as they represent an identification of 'best practice'.
- Quality management procedures may be documented in an organizational quality manual, based on the generic model for a quality manual suggested in the ISO 9001 standard.

Thank You

Software Quality Management

- K G Krishna

Overview of 7 QC Tools ➔

7 QC Tools in Software Industry!

These 7 QC Tools pioneered by Japanese for Quality Control in their Manufacturing industry during 60’s have been widely adopted by Managers across the world in all sectors of business in their day-to-day operations

Seven Quality Control Tools

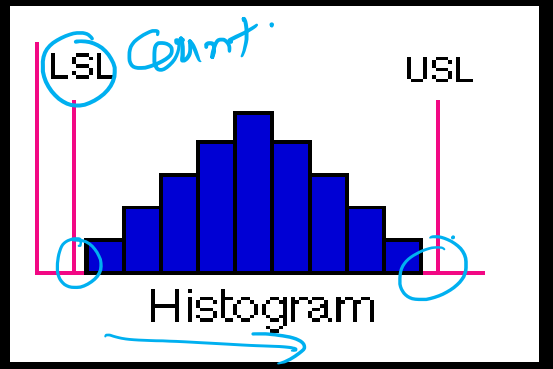
- Kaoru Ishikawa (Japan, 1960) developed seven basic visual tools of quality so that the average person could analyze and interpret data.

- These tools have been used worldwide by companies, managers of all levels and employees.

- The Seven Tools
  - Histograms, Pareto Charts, Cause and Effect Diagrams, Run Charts, Scatter Diagrams, Flow Charts, and Control Charts

Source courtesy: Contents of the above PPT is an extract from net sources-- all credits to their original authors

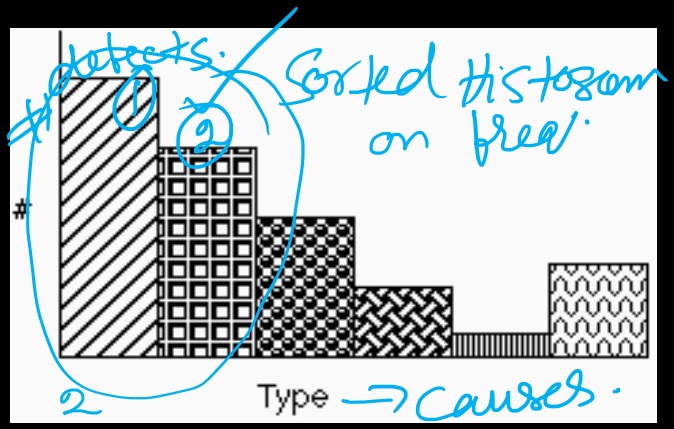
Histograms



- Histogram
  - A histogram is a bar graph that shows frequency data.
  - Histograms provide the easiest way to evaluate the distribution of data.
- Creating a Histogram
  - Collect data and sort it into categories.
  - Then label the data as the independent set or the dependent set.
    - The characteristic you grouped the data by would be the independent variable.
    - The frequency of that set would be the dependent variable.
  - Each mark on either axis should be in equal increments.
  - For each category, find the related frequency and make the horizontal marks to show that frequency.
- Examples of How Histograms Can Be Used
  - Histograms can be used to determine distribution of sales.
  - Say for instance a company wanted to measure the revenues of other companies and wanted to compare numbers.

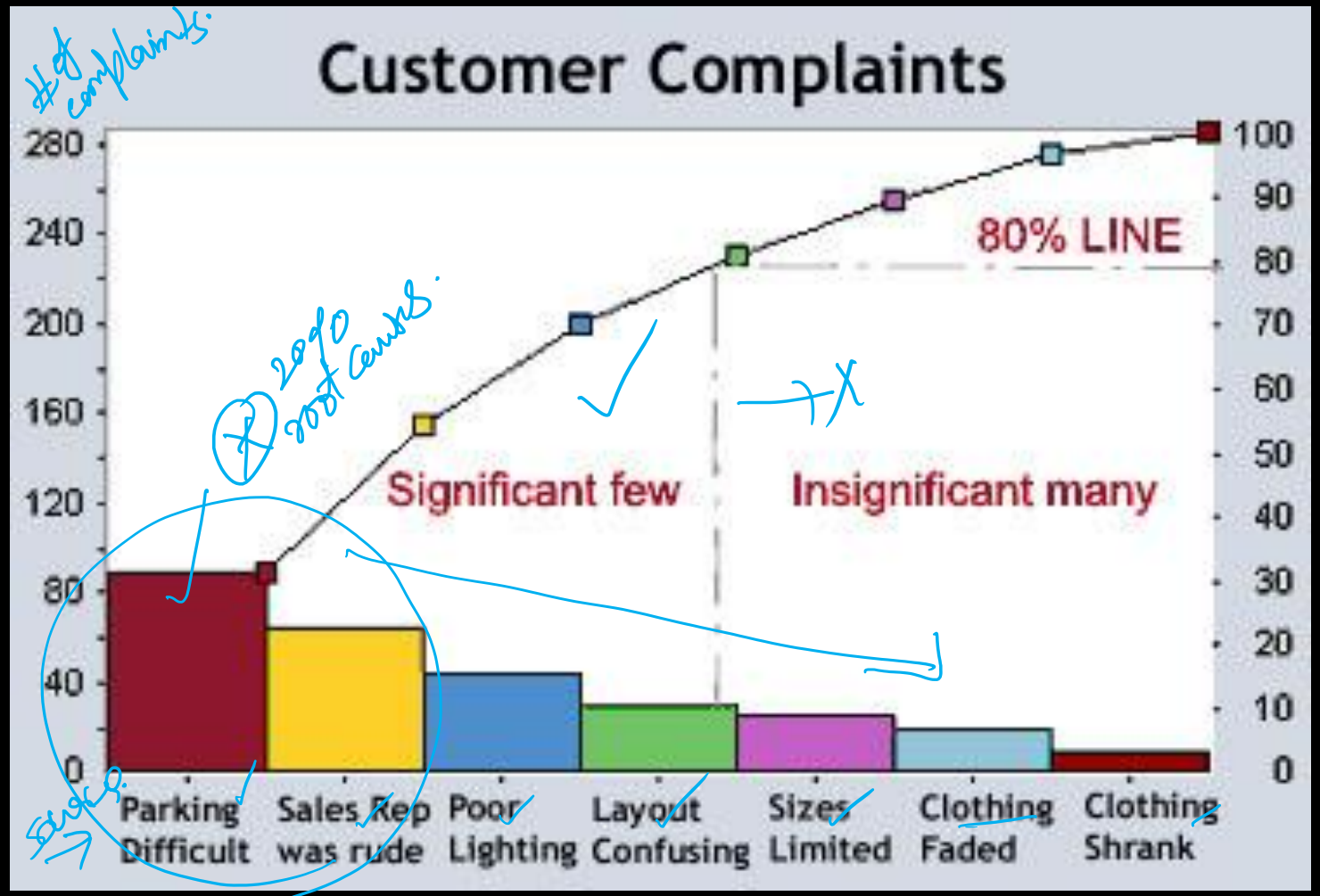
Pareto Charts

80/20 rule.



- Pareto Charts
  - Pareto charts are used to identify and prioritize problems to be solved.
  - They are actually histograms aided by the 80/20 rule adapted by Joseph Juran.
    - Remember the 80/20 rule states that approximately 80% of the problems are created by approximately 20% of the causes.
- Constructing a Pareto Chart
  - First, information must be selected based on types or classifications of defects that occur as a result of a process.
  - The data must be collected and classified into categories.
  - Then a histogram or frequency chart is constructed showing the number of occurrences.
- An Example of Use
  - Pareto Charts are used when products are suffering from different defects but the defects are occurring at a different frequency, or only a few account for most of the defects present, or different defects incur different costs. What we see from that is a product line may experience a range of defects. The manufacturer could concentrate on reducing the defects which make up a bigger percentage of all the defects or focus on eliminating the defect that causes monetary loss.

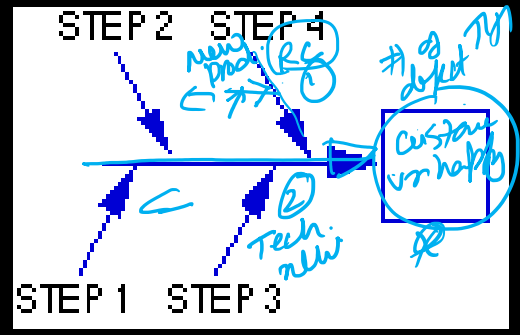
Pareto Chart – An Example



Source courtesy: Contents of the above PPT is an extract from net sources-- all credits to their original authors

Cause and Effect (Fishbone) Diagram

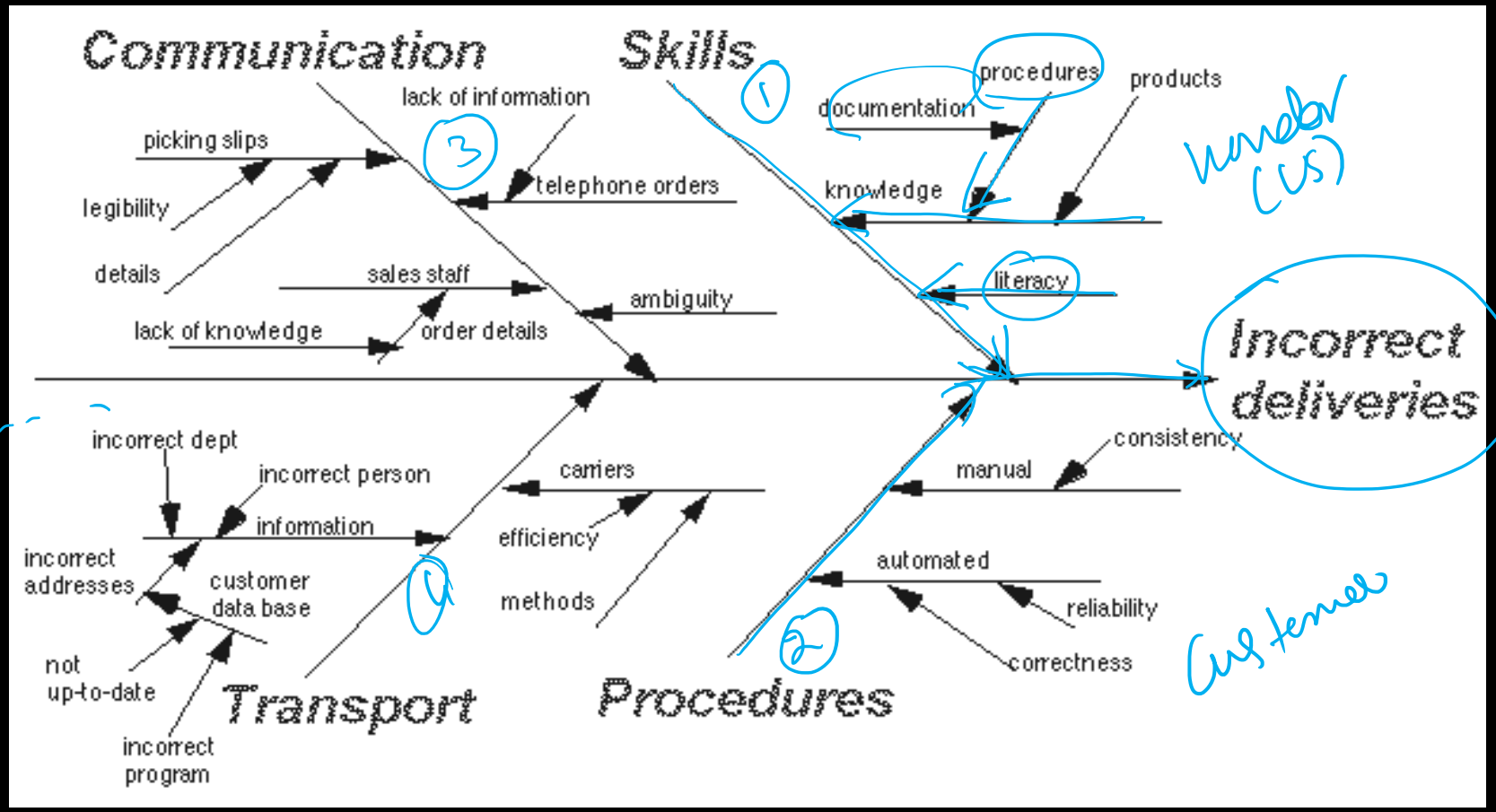
- Cause and Effect Diagram (Ishikawa diagram / Fishbone diagram)
  - It is a tool for discovering all the possible causes for a particular effect.
  - The major purpose of this diagram is to act as a first step in problem solving by creating a list of possible causes.
- Constructing a Cause and Effect Diagram
  - First, clearly identify and define the problem or effect for which the causes must be identified. Place the problem or effect at the right or the head of the diagram.
  - Identify all the broad areas of the problem.
  - Write in all the detailed possible causes in each of the broad areas.
  - Each cause identified should be looked upon for further more specific causes.
  - View the diagram and evaluate the main causes.
  - Set goals and take action on the main causes.



Source courtesy: Contents of the above PPT is an extract from net sources-- all credits to their original authors

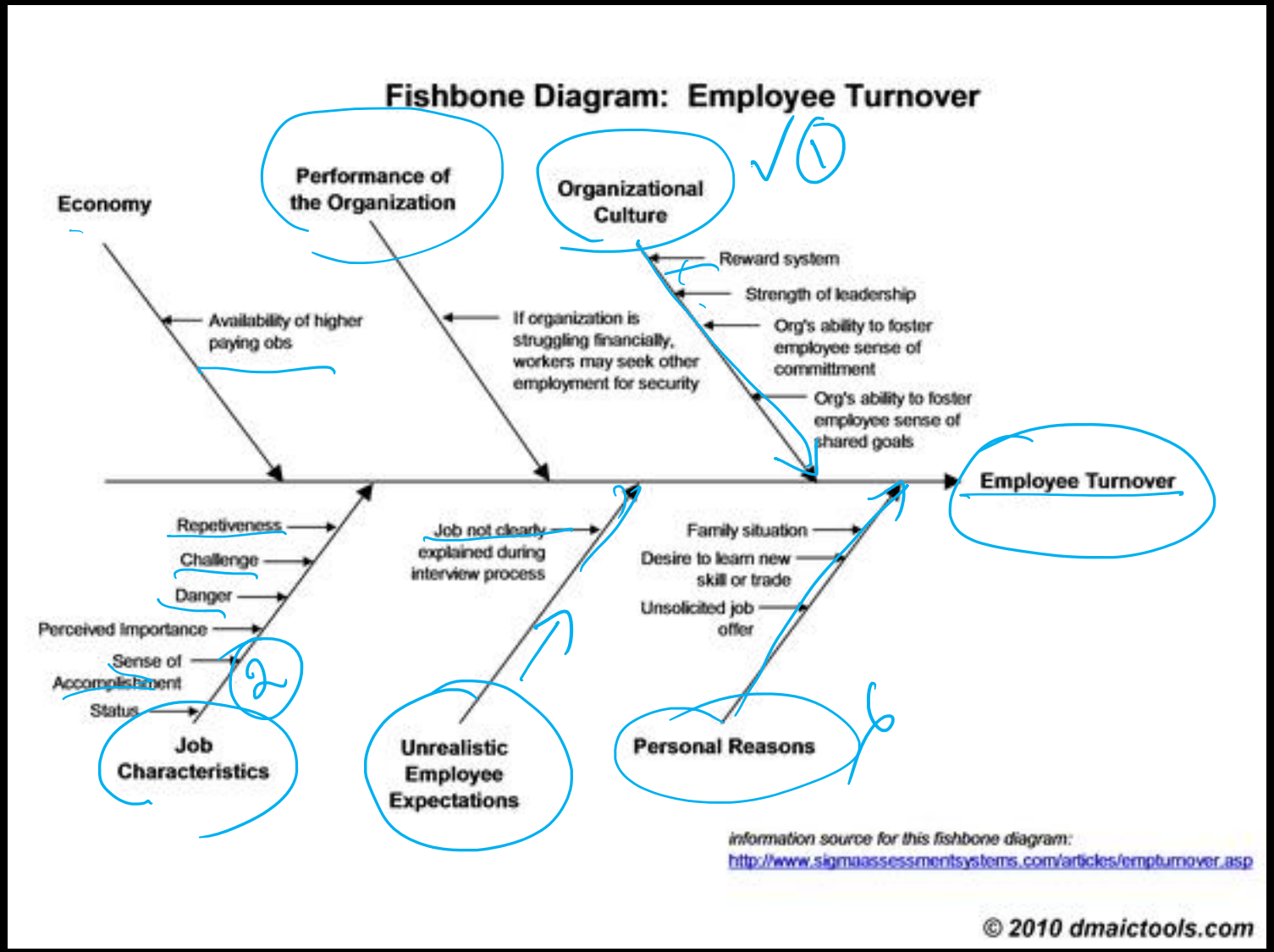
Cause and Effect Diagram – An Example

- When a production team is about to launch a new product, the factors that will affect the final product must be recognized. The fishbone diagram can depict problems before they have a chance to begin.



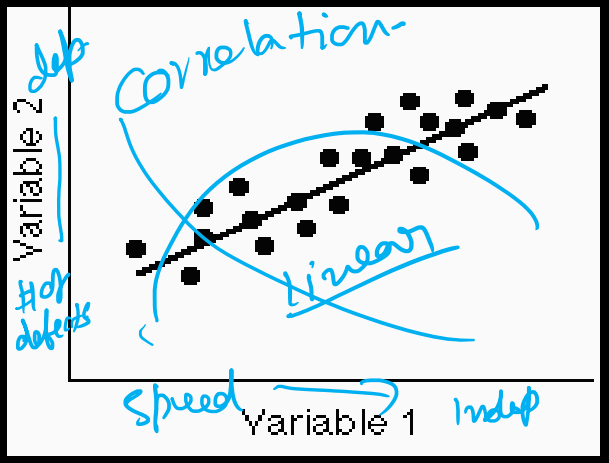
Source courtesy: http://www.hci.com.au/hcisite/toolkit/causeand.htm

Cause and Effect Diagram – An Example



Scatter Diagrams

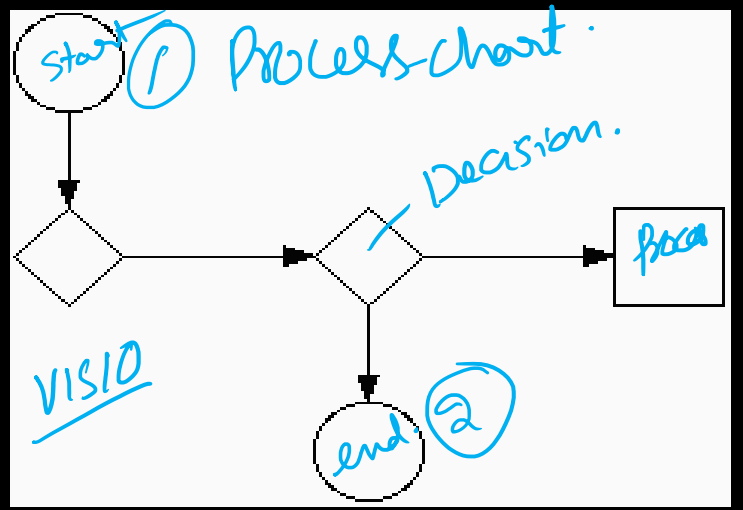
- Scatter Diagrams
  - Scatter Diagrams are used to study and identify the possible relationship between the changes observed in two different sets of variables.
- Constructing a Scatter Diagram
  - First, collect two pieces of data and create a summary table of the data.
  - Draw a diagram labeling the horizontal and vertical axes.
    - It is common that the “cause” variable be labeled on the X axis and the “effect” variable be labeled on the Y axis.
  - Plot the data pairs on the diagram.
  - Interpret the scatter diagram for direction and strength.
- An Example
  - A scatter diagram can be used to identify the relationship between the production speed of an operation and the number of defective parts made.



Source courtesy: Contents of the above PPT is an extract from net sources-- all credits to their original authors

Flow Charts

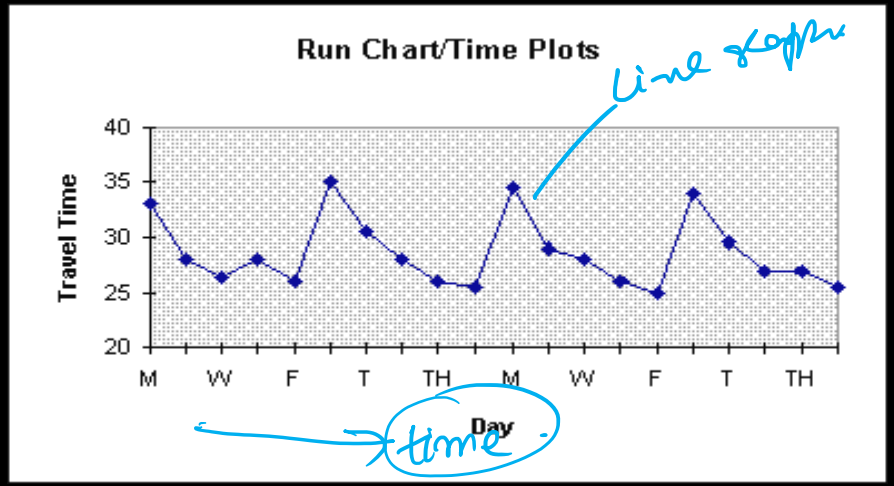
- Flow Chart
  - A flow chart is a pictorial representation showing all of the steps of a process.
- Creating a Flow Chart
  - First, familiarize the participants with the flow chart symbols.
  - Draw the process flow chart and fill it out in detail about each element.
  - Analyze the flow chart. Determine which steps add value and which don't in the process of simplifying the work.
- Examples of Use
  - When the Two separate stages of a process flow chart should be considered:
    - The making of the product vs. The finished product



Source courtesy: Contents of the above PPT is an extract from net sources-- all credits to their original authors

Run Charts

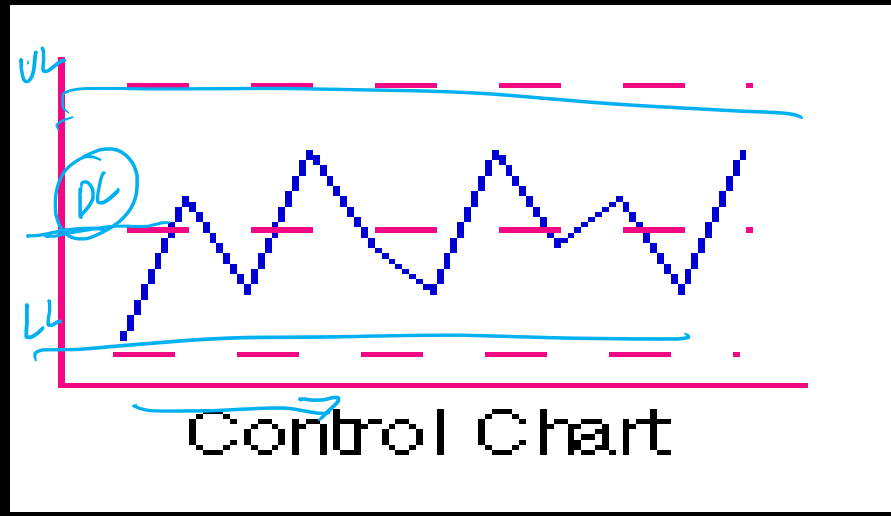
- Run Chart
  - Run charts are used to analyze processes according to time or order.
- Creating a Run Chart
  - Gathering Data (Some type of process or operation must be available to take measurements for analysis.)
  - Organizing Data (Data must be divided into two sets of values X and Y. X values represent time and values of Y represent the measurements taken from the manufacturing process or operation.)
  - Charting Data (Plot the Y values versus the X values.)
  - Interpreting Data (Interpret the data and draw any conclusions that will be beneficial to the process or operation.)
- An Example of Use
  - An organization's desire is to have their product arrive to their customers on time, but they have noticed that it doesn't take the same amount of time each day of the week. They decided to monitor the amount of time it takes to deliver their product over the next few weeks.



Source courtesy: Contents of the above PPT is an extract from net sources-- all credits to their original authors

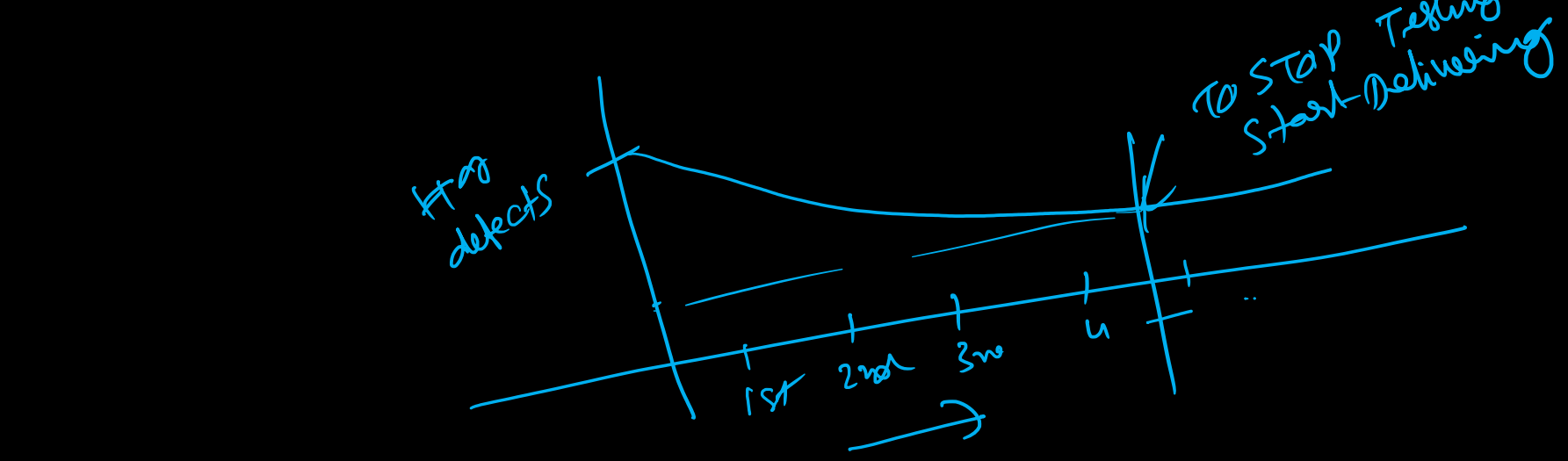
Control Charts

- Control Charts
  - Control charts are used to determine whether a process will produce a product or service with consistent measurable properties.
- Steps Used in Developing Process Control Charts
  - Identify critical operations in the process where inspection might be needed.
  - Identify critical product characteristics.
  - Determine whether the critical product characteristic is a variable or an attribute.
  - Select the appropriate process control chart.
  - Establish the control limits and use the chart to monitor and improve.
  - Update the limits.
- An Example of Use
  - Counting the number of defective products or services
    - Do you count the number of defects in a given product or service?
    - Is the number of units checked or tested constant?



Source courtesy: Contents of the above PPT is an extract from net sources-- all credits to their original authors

Thank You



Software Quality Management

- K G Krishna

‘Six-Sigma Thinking’ ➔

“Six-Sigma Thinking”

- **Understanding Six Sigma**
- History of Six Sigma
- Six Sigma Methodologies & Tools
- Roles & Responsibilities
- How *YOU* can use Six Sigma in your own Orgzn./Projects

“Six-Sigma Thinking” is all about::

- Process-centricity
- Measurement / Metrics
- Productivity / Defect Reduction
- A Methodology for Process Improvement
- Innovation in eliminating ‘*Opportunities for Defects*’

Beyond statistical definition of 6-Sigma (3.4 defects per million opportunities for defects or 99.9997% defect-free), learning about Six-Sigma Methodology inculcates thinking in terms of Process, Metrics orientation and relentless pursuit of excellence in ensuring near ‘defect-free’ delivery.

“Process makes Ordinary People deliver extraordinary results.”

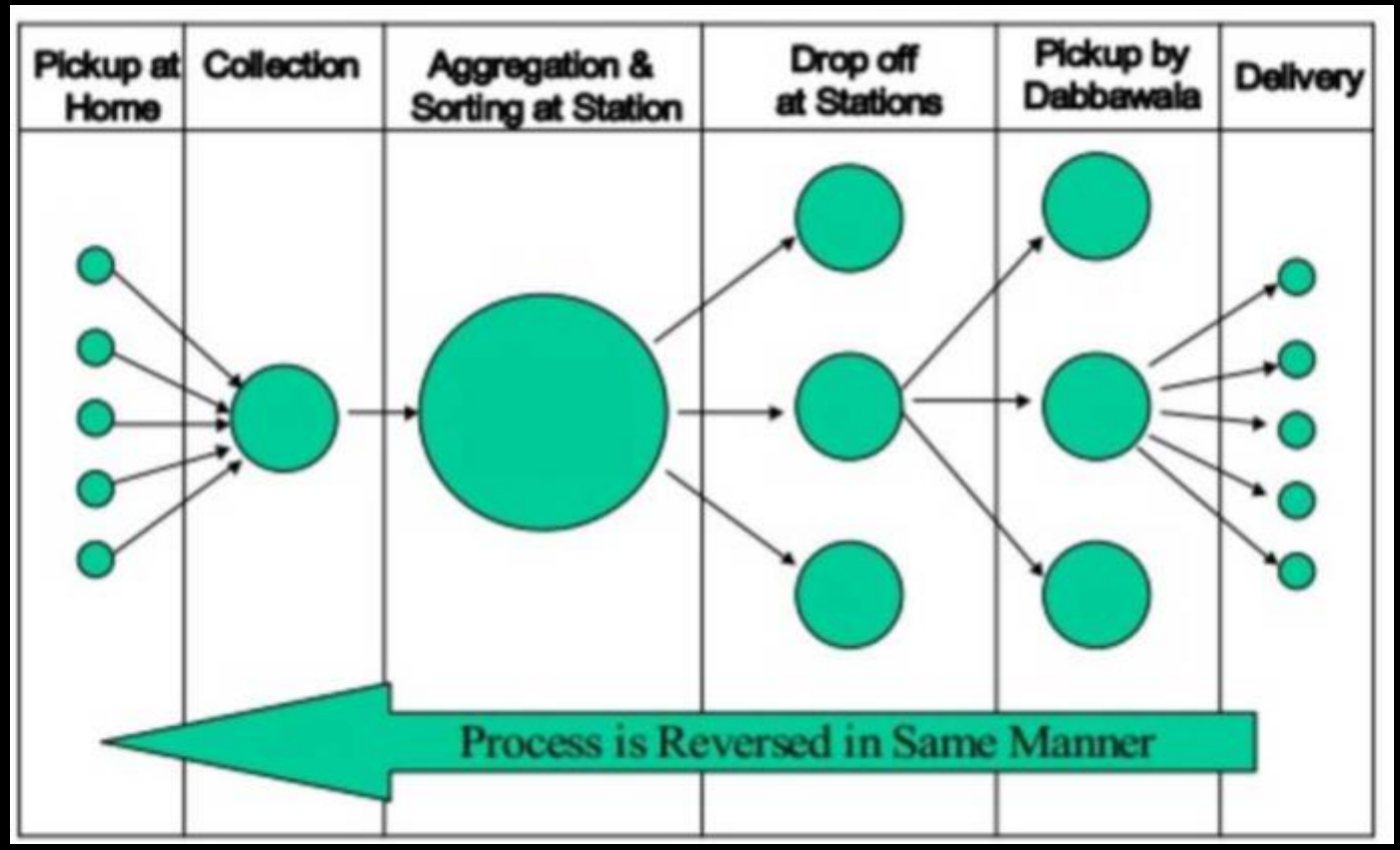
- “The Mumbai *Dabbawalas* have shown with the right system, organization does not need extraordinary talent to achieve extraordinary performance.

- Leaders who see themselves as system architects can obtain the same results by designing their systems/processes.”

http://image.slidesharecdn.com/themumbaidabbawalasharvardbusinessreviewnovember-141027084641-conversion-gate01/95/the-mumbai-dabbawalas-6-638.jpg?cb=1414401871

The Process of Mumbai *Dabbawala*!

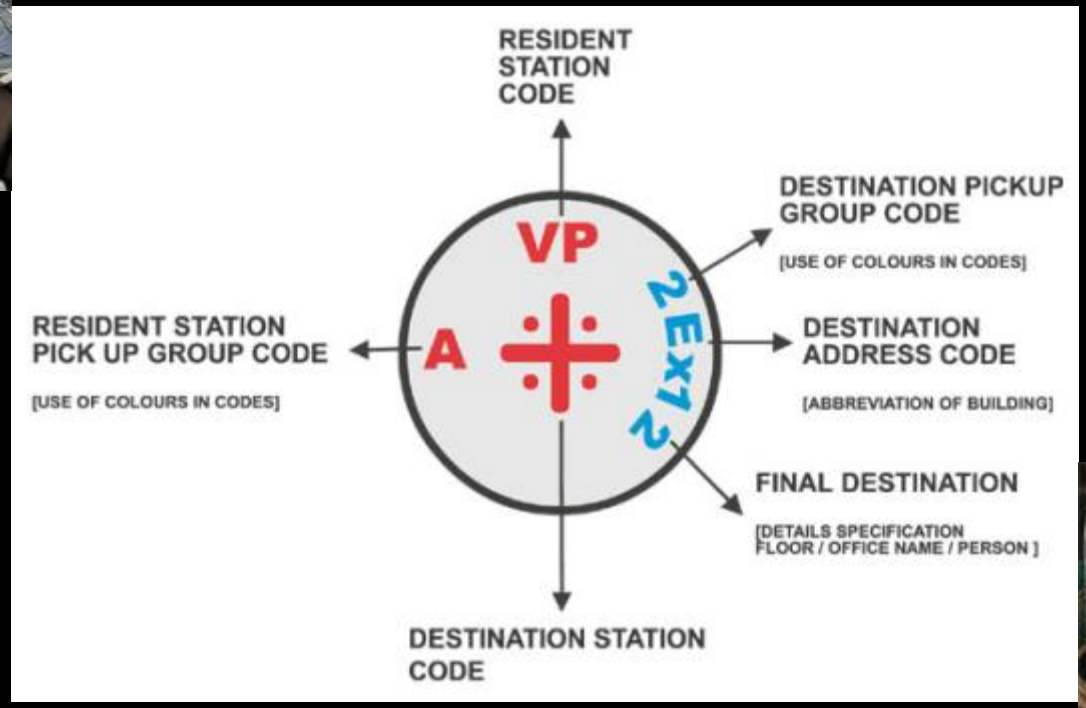
(Involves People, Methods/Process, Culture, Technology)



"…More than their management style, the love and enthusiasm with which they work and carry the food, makes them the lord of this service. They don't just carry food for people; they carry the love that one wants in their food…"

https://www.thirstt.com/droplets/Management-Styles-Learning-the-Dabbawalas-way/53d4d0f300403cff5d75b038

It Starts with Coding of *Dabbas…*



https://hbr.org/resources/images/article_assets/hbr/1211/R1211K_A.jpg

What is extraordinary about *Dabbawalas*?

Nuttan Mumbai Tiffin boxes Suppliers Association

- History-started in 1890
- Charitable trust-Registered in 1956
- Education -85% illiterate
- Total area coverage -60 km/70km
- Employee stength-5000
- No. of dabbas-200000 dabbas i.e. 400000 transaction everyday
- Time taken- 3hrs

MUMBAI DABBAWALAS

- **There are 5,000 or so dabbawalas in the city who have an astounding service record.**
- **Every working day they transport more than 130,000 lunchboxes throughout Mumbai, the world's fourth-most-populous city.**
- **That entails conducting upwards of 260,000 transactions in six hours each day, six days a week, 52 weeks a year (minus holidays), but mistakes are extremely rare.**
- **Amazingly, the dabbawalas—semiliterate workers who largely manage themselves—have achieved that level of performance at very low cost, in an ecofriendly way, without the use of any IT system or even cell phones.**

http://image.slidesharecdn.com/themumbaidabbawalasharvardbusinessreviewnovember-141027084641-conversion-gate01/95/the-mumbai-dabbawalas-4-638.jpg?cb=1414401871

Dabbawalas: Let's Look at Complexity of their Operations

Many-Many Network

Dabbawala makes only One Error 1 in 16 Million

(More than 6 sigma!)

THE DABBAWAL SYSTEM:ON -TIME DELIVERY, EVERY TIME

“Defect Definition?”

“Defect = Undesired Result, Unpleasant Customer Experience,
Deviation from Specifications

“Defects are mostly due to badly designed or Implemented Process”

“Mistake-proofing Process using Six-Sigma Methodology”

“**Metrics orientation** is the key to Management and **Continuous Improvement** of Process

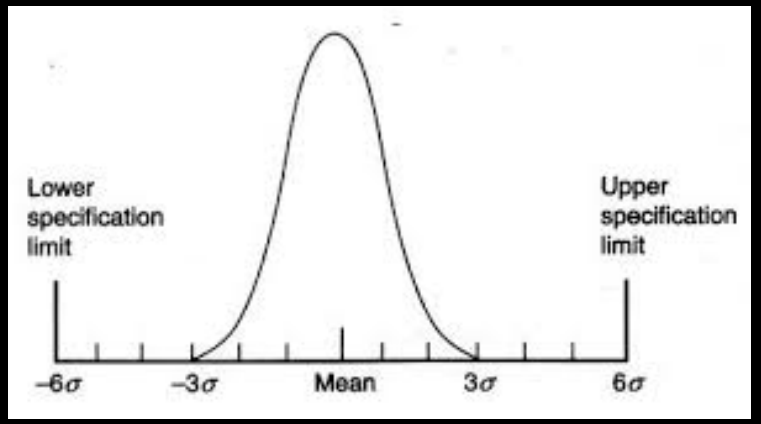
“Measuring is everything!”

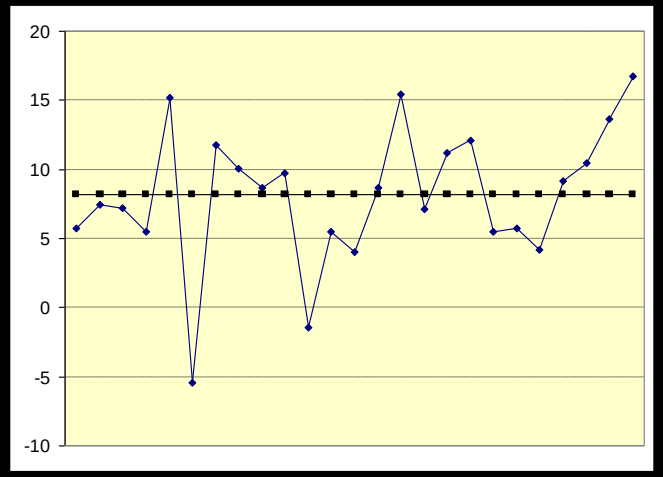
6-Sigma → 3.4 Defects per Million Opportunities-for-Defects

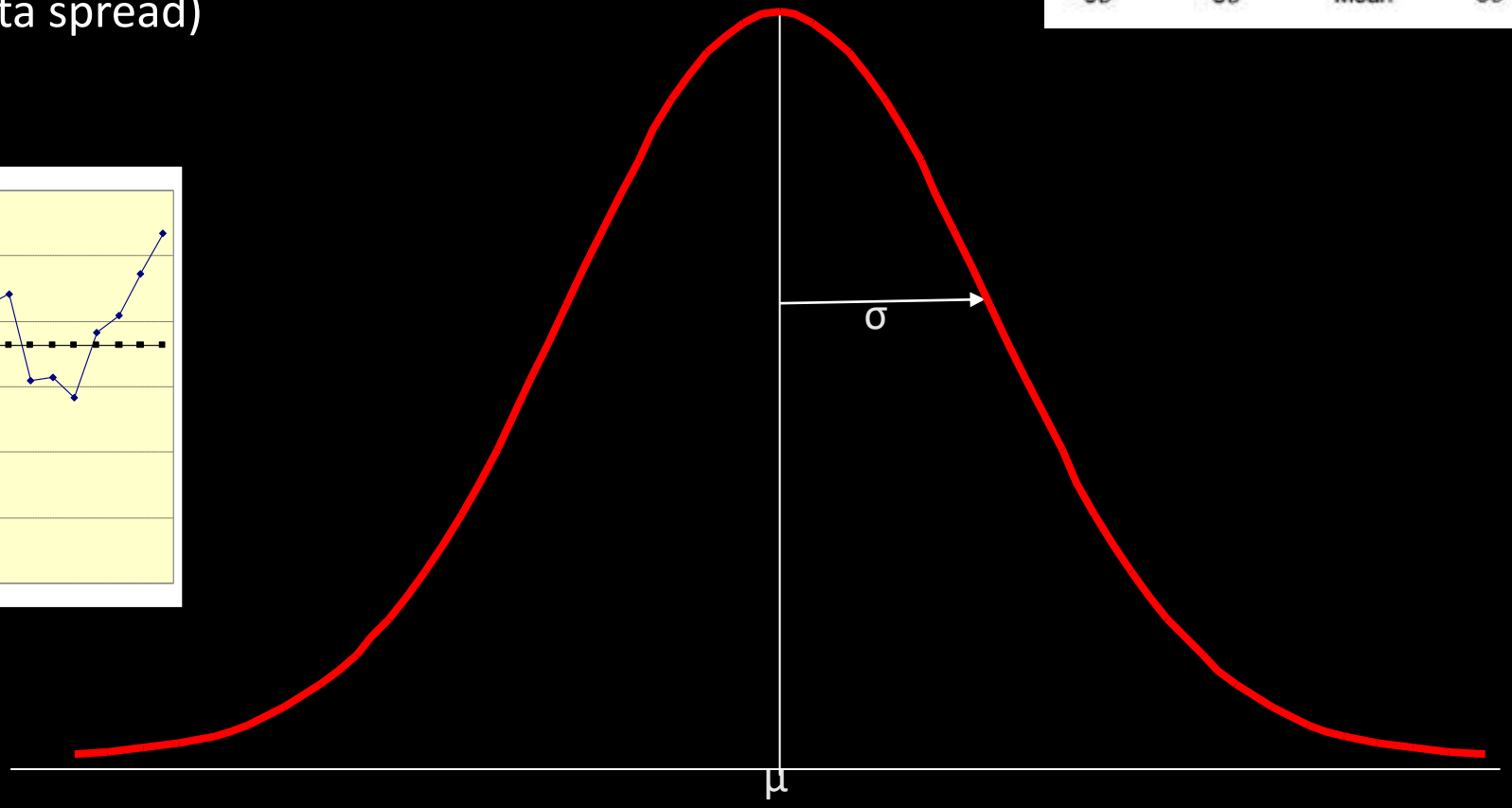
- ✓"Six-Sigma is a quantitative measure of Process Deviation"
- ✓A performance goal - representing 3.4 defects for every million opportunities to make one.
- ✓A statistical measure indicating the number of standard deviations within customer expectations.
- ✓A series of tools and methods used to improve or design products, processes, and/or services.
- ✓A disciplined, fact-based approach to managing a business and its processes.
- ✓A means to promote greater awareness of customer needs, performance measurement, and business improvement.

6-Sigma: What's in a name?

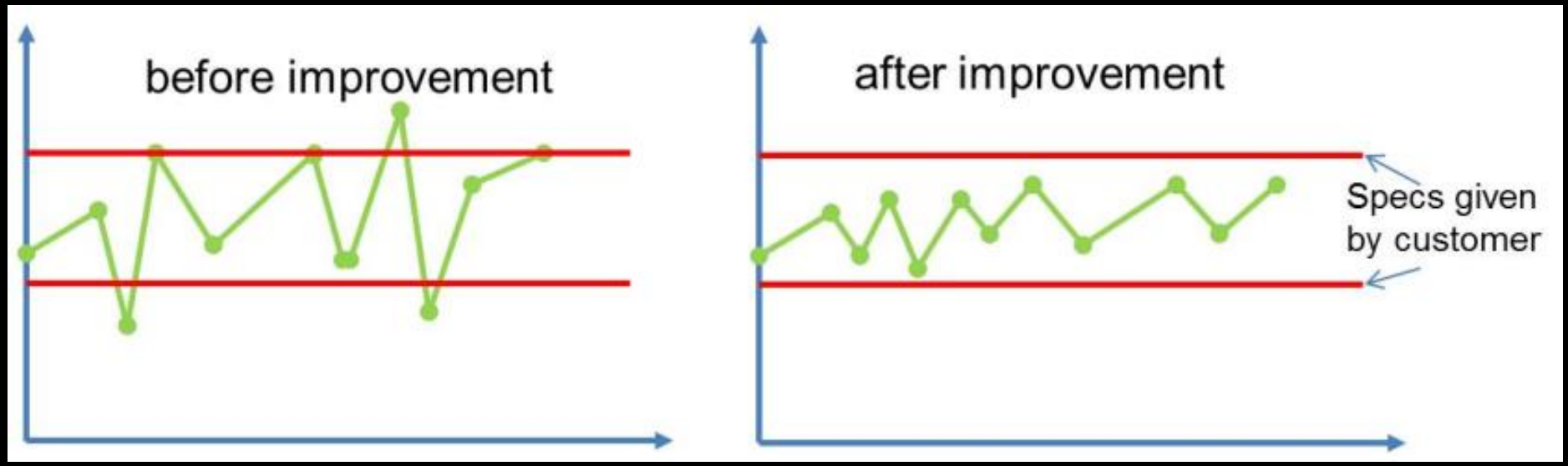
- Sigma is the Greek letter representing the standard deviation of a population of data.
- Sigma is a measure of *variation* (the data spread)







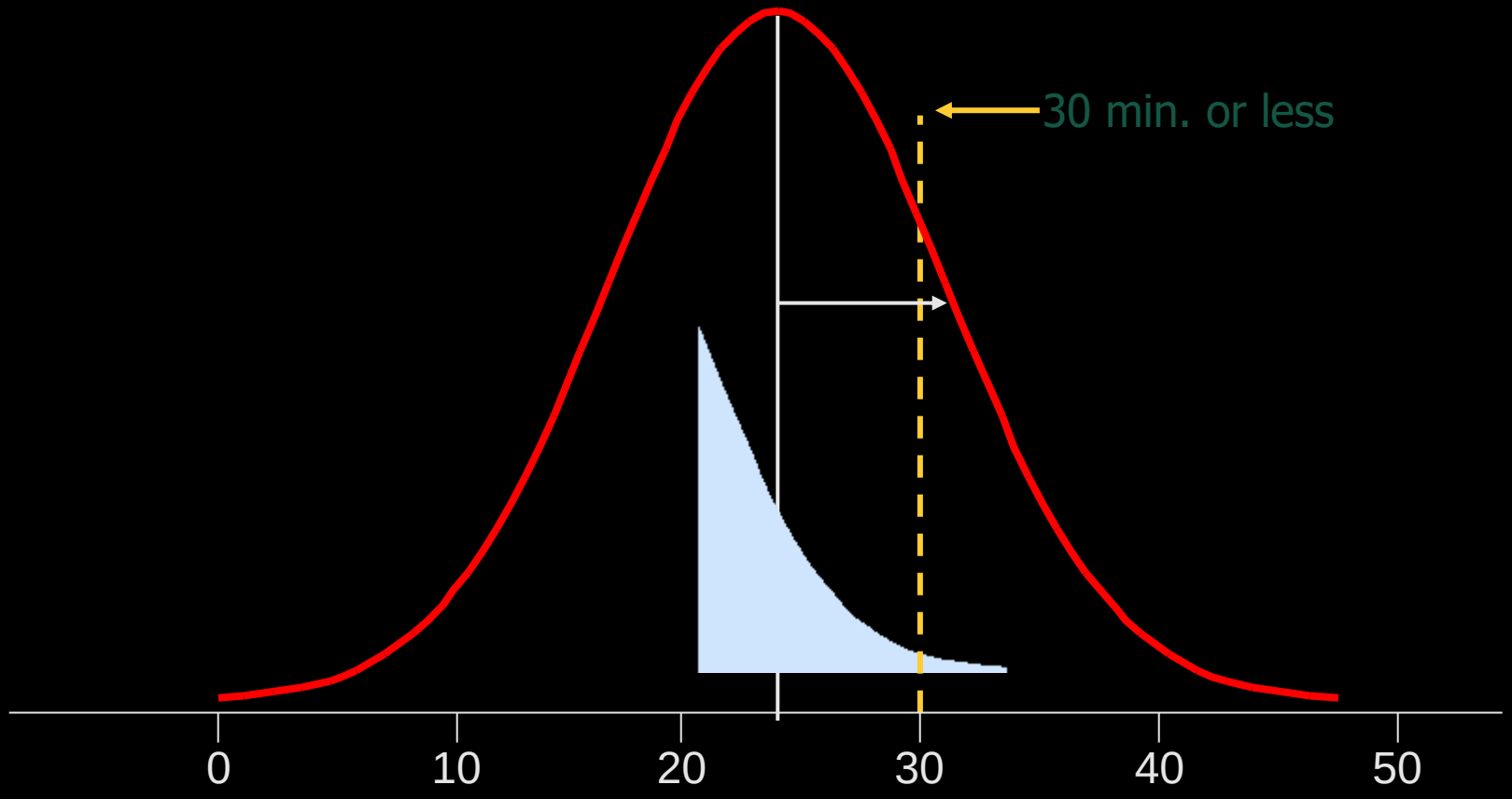
Six-Sigma is about Controlling Variation in a Process



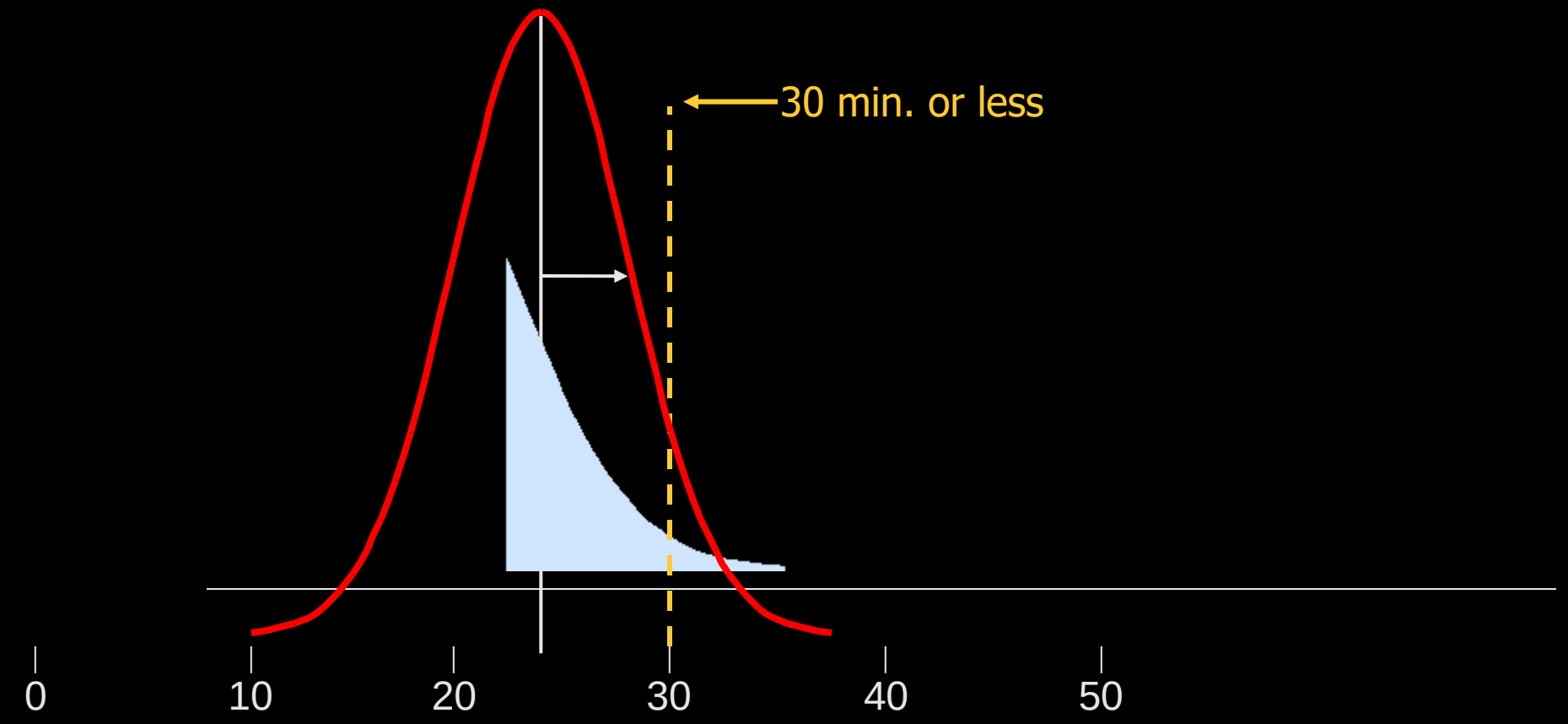
Customers do not feel averages!

Managing by the average doesn't tell the whole story.

The average *and* the variation *together* show what's happening.



Sigma level measures how often we meet (or fail to meet) the requirement(s) of our customer(s).



“Six-Sigma Methodology is a structured framework for undertaking systematic and measurable process improvement for reducing variation in Processes.”

Reducing Defects in end Product/Service

=

Eliminating Opportunities-for-Defects in the Process

“Six-Sigma measure for a Process is…”

“Sigma Rating” ➔

$$\frac{\text{No. of actual Defects observed}}{\text{No. of Opportunities-for-Defects that existed in the Process}}$$

Measuring Process Performance

- Customers want their pizza delivered fast!
- Guarantee = “30 minutes or less”
- What if we measured performance and found an average delivery time of 23.5 minutes?
  - On-time performance is great, right?
  - Our customers must be happy with us, right?

Managing Up the Sigma Scale

Sigma	% Good	% Bad	DPMO
1	30.9%	69.1%	691,462
2	69.1%	30.9%	308,538
3	93.3%	6.7%	66,807
4	99.38%	0.62%	6,210
5	99.977%	0.023%	233
6	99.9997%	0.00034%	3.4

Examples of the Sigma Scale

In a world at 3 sigma. . .

- There are 964 U.S. flight cancellations per day.
- The police make 7 false arrests every 4 minutes.
- In MA, 5,390 newborns are dropped each year.
- In one hour, 47,283 international long distance calls are accidentally disconnected.

In a world at 6 sigma. . .

- 1 U.S. flight is cancelled every 3 weeks.
- There are fewer than 4 false arrests per month.
- 1 newborn is dropped every 4 years in MA.
- It would take more than 2 years to see the same number of dropped international calls.

History of Six-Sigma

The Six Sigma Evolutionary Timeline

1818: Gauss uses the normal curve to explore the mathematics of error analysis for measurement, probability analysis, and hypothesis testing.

1924: Walter A. Shewhart introduces the control chart and the distinction of special vs. common cause variation as contributors to process problems.

1736: French mathematician Abraham de Moivre publishes an article introducing the normal curve.

1896: Italian sociologist Vilfredo Alfredo Pareto introduces the 80/20 rule and the Pareto distribution in *Cours d'Economie Politique*.

1949: U. S. DOD issues Military Procedure MIL-P-1629, *Procedures for Performing a Failure Mode Effects and Criticality Analysis*.

1960: Kaoru Ishikawa introduces his now famous cause-and-effect diagram.

1941: Alex Osborn, head of BBDO Advertising, fathers a widely-adopted set of rules for "brainstorming".

1970s: Dr. Noriaki Kano introduces his two-dimensional quality model and the three types of quality.

1986: Bill Smith, a senior engineer and scientist introduces the concept of Six Sigma at Motorola

1995: Jack Welch launches Six Sigma at GE.

AlliedSignal

MOTOROLA

1994: Larry Bossidy launches Six Sigma at Allied Signal.

Six-Sigma Companies

Six-Sigma in Financial Services

Six-Sigma Methodologies & Tools

DMAIC – The Improvement Methodology

Define	Measure	Analyze	Improve	Control
Objective: DEFINE the opportunity	**Objective**: MEASURE current performance	**Objective**: ANALYZE the root causes of problems	**Objective**: IMPROVE the process to eliminate root causes	**Objective**: CONTROL the process to sustain the gains.
Key Define Tools: • Cost of Poor Quality (COPQ) • Voice of the Stakeholder (VOS) • Project Charter • As-Is Process Map(s) • Primary Metric (Y)	**Key Measure Tools**: • Critical to Quality Requirements (CTQs) • Sample Plan • Capability Analysis • Failure Modes and Effect Analysis (FMEA)	**Key Analyze Tools**: • Histograms, Boxplots, Multi-Vari Charts, etc. • Hypothesis Tests • Regression Analysis	**Key Improve Tools**: • Solution Selection Matrix • To-Be Process Map(s)	**Key Control Tools**: • Control Charts • Contingency and/or Action Plan(s)

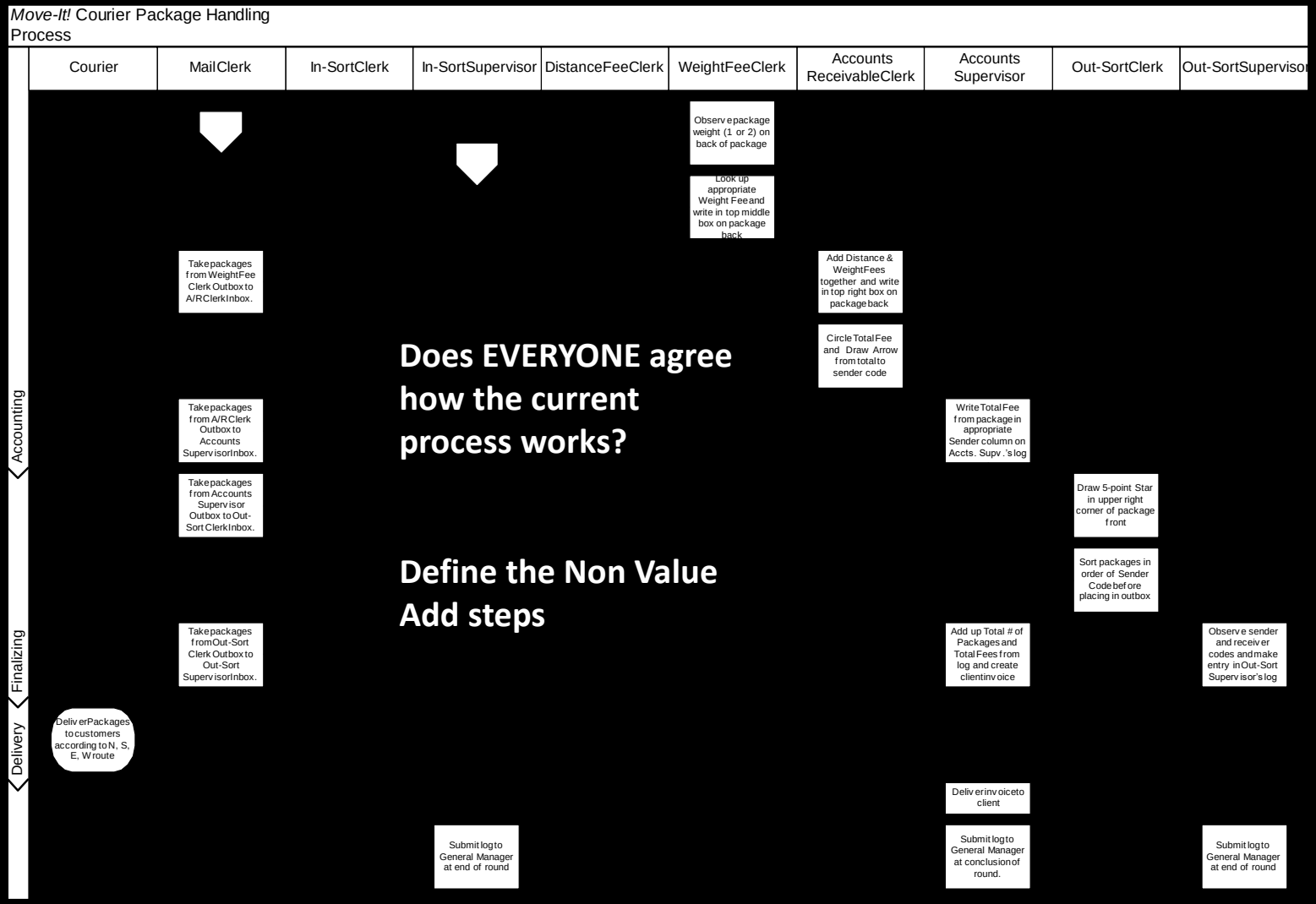
Define – DMAIC Project

What is the project?

- What is the problem? The “problem” is the Output (a “Y” in a math equation Y=f(x1,x2,x3) etc).
- What is the cost of this problem
- Who are the stake holders / decision makers
- Align resources and expectations

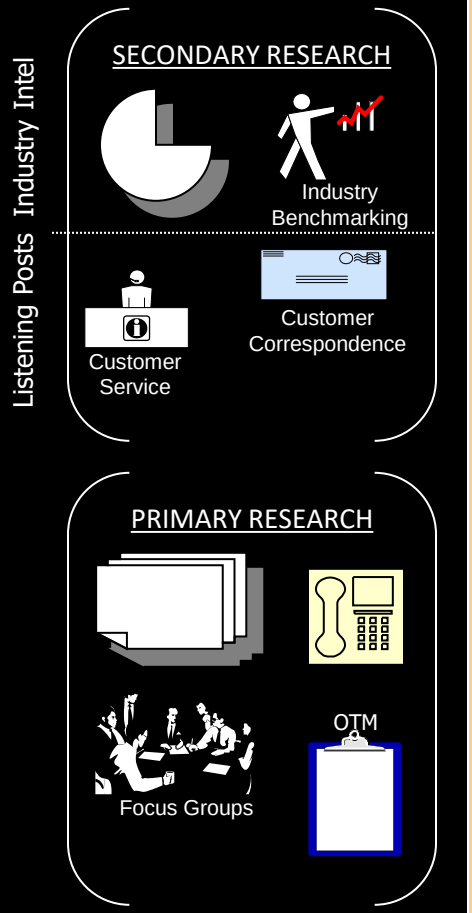
Define – As-Is Process

How does our existing process work?



Define – Customer Requirements

What are the CTQs? What motivates the customer?

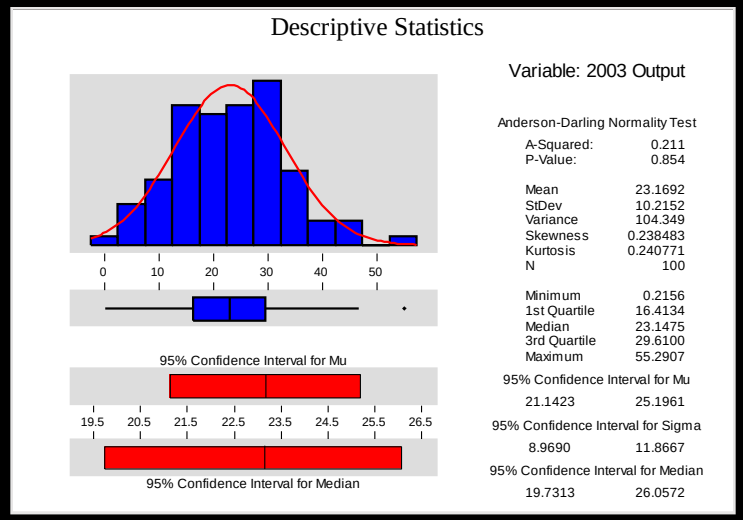


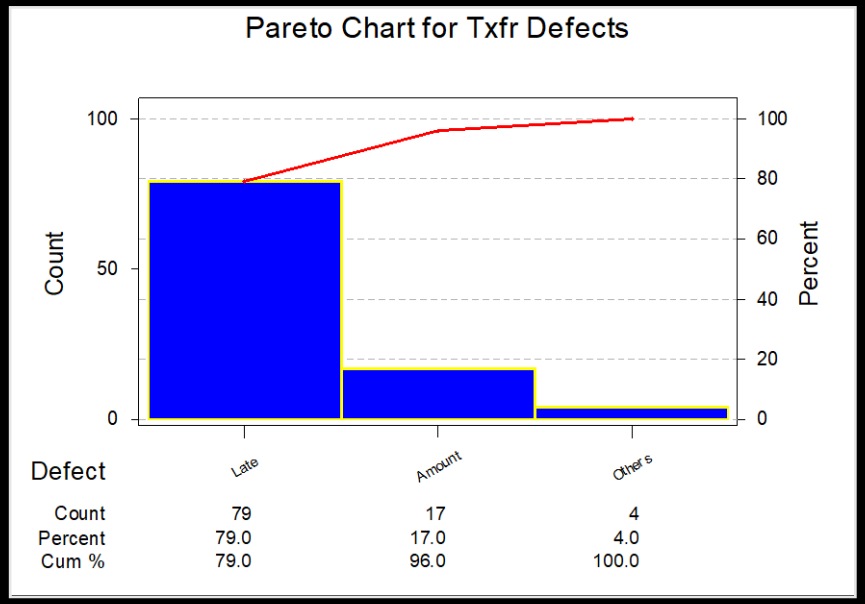
Voice of the Customer	Key Customer Issue	Critical to Quality

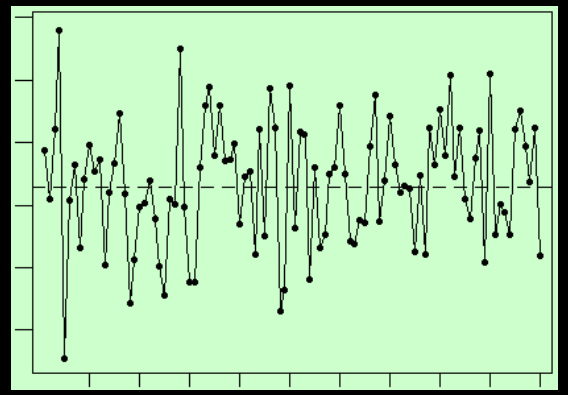
Measure – Baselines and Capability

What is our current level of performance?

- Sample some data / not all data
- Current Process actuals measured against the Customer expectation
- What is the chance that we will succeed at this level every time?







Measure – Failures and Risks

Where does our process fail and why?

Subjective opinion mapped into an "objective" risk profile number

Failure Modes and Effects Analysis (FMEA)

Process/Product

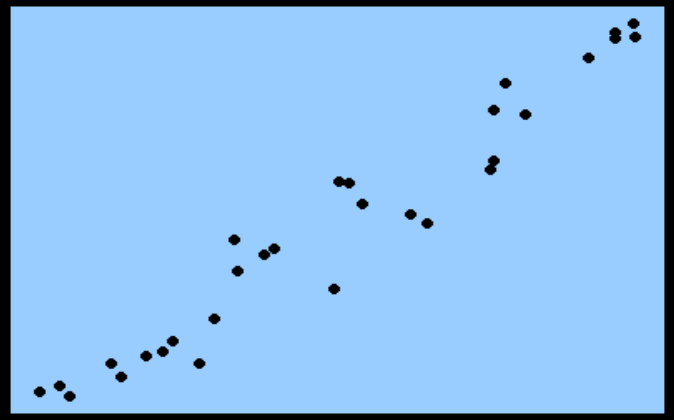
Process or Product Name:		Prepared by:	Page ____ of ____
Responsible:		FMEA Date (Orig) _____ (Rev) _____	

Process Step/Part Number	Potential Failure Mode	Potential Failure Effects	SEV	Potential Causes	OCC	Current Controls	DET	RPN	Actions Recommended	Resp.	Actions Taken	SEV	OCC	DET	RPN
				X1				0							0
								0							0
				X2				0							0
								0							0
				X3				0							0
								0							0
				X4				0							0
								0							0
								0							0
				etc				0							0
								0							0
								0							0
								0							0
								0							0
								0							0
								0							0
								0							0
								0							0

Analyze – Potential Root Causes

What affects our process?

Ishikawa Diagram
(Fishbone)

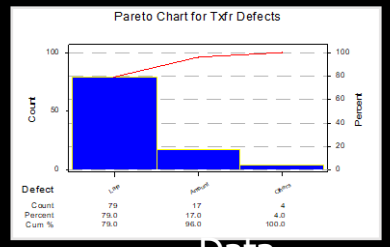


Six Sigma

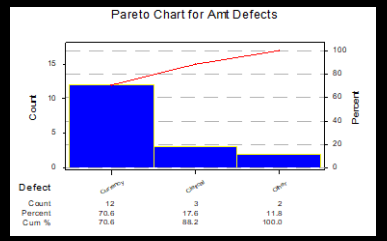
$$y = f(x_1, x_2, x_3 \ldots x_n)$$

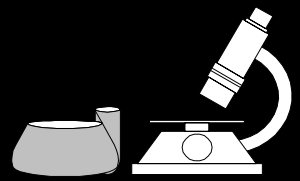
Analyze – Validated Root Causes

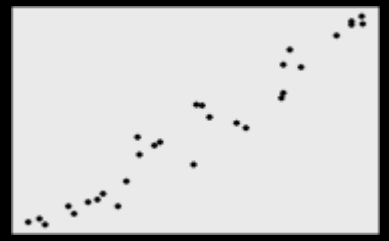
What are the key root causes?



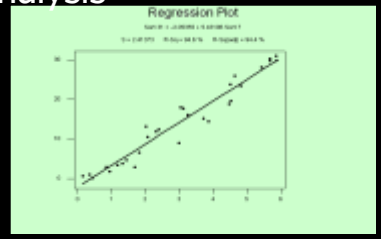
Data Stratification







Regression Analysis



Six Sigma

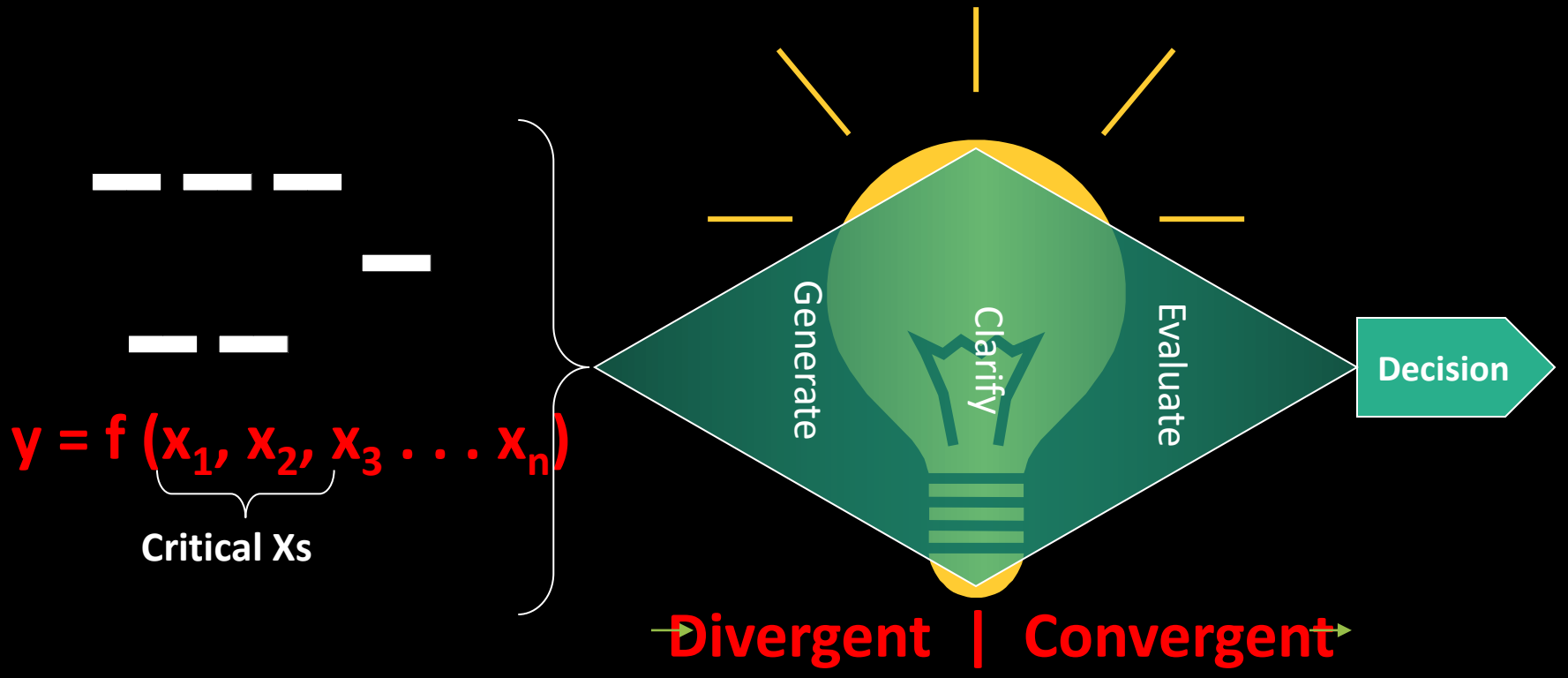
$$y = f(x_1, x_2, x_3 \ldots x_n)$$

Critical Xs

Improve – Potential Solutions

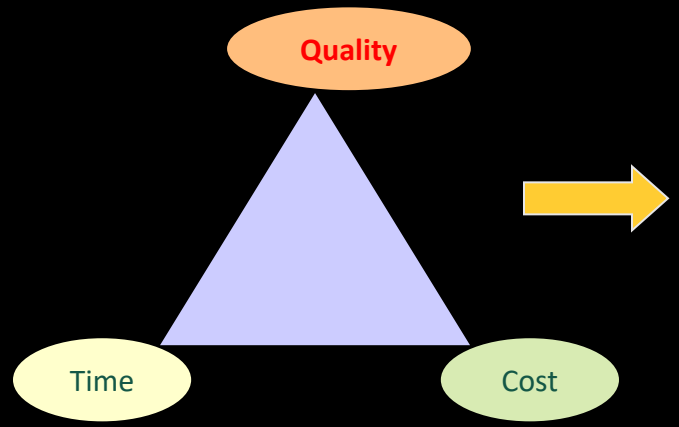
How can we address the root causes we identified?

- Address the causes, not the symptoms.



Improve – Solution Selection

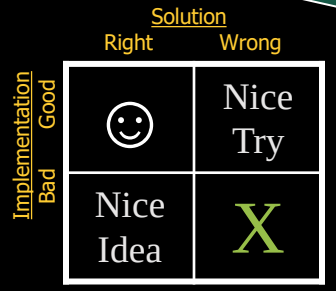
How do we choose the best solution?



Solution Selection Matrix

Solution	Sigma	Time	CBA	Other	Score

Six Sigma

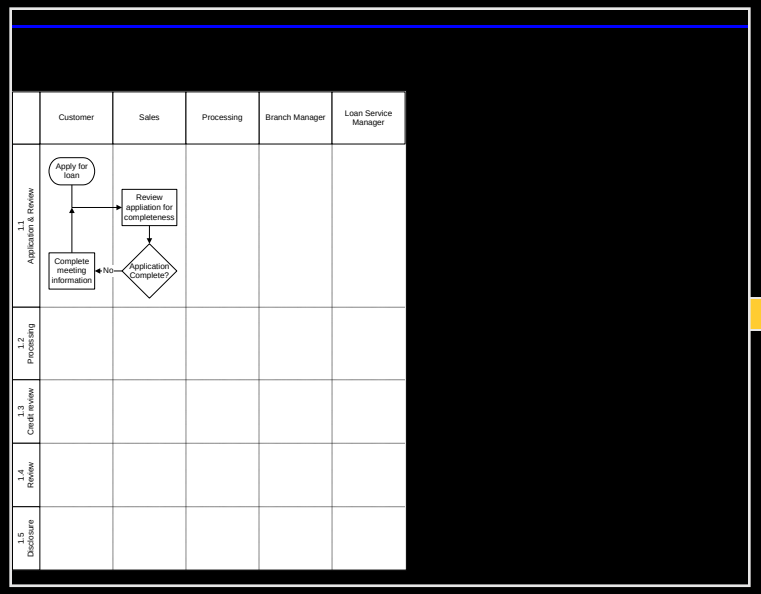


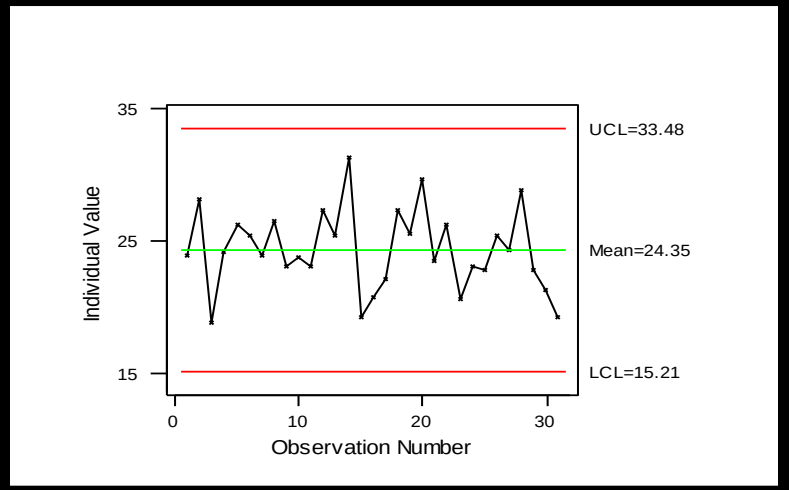
Solution Implementation Plan

Control – Sustainable Benefits

How do we "hold the gains" of our new process?

- Some variation is normal and OK
- How High and Low can an "X" go yet not materially impact the "Y"
- Pre-plan approach for control exceptions





DFSS – The Design Methodology

Design for Six Sigma

- Uses
  - Design new processes, products, and/or services from scratch
  - Replace old processes where improvement will not suffice
- Differences between DFSS and DMAIC
  - Projects typically longer than 4-6 months
  - Extensive definition of Customer Requirements (CTQs)
  - Heavy emphasis on benchmarking and simulation; less emphasis on baselining
- Key Tools
  - Multi-Generational Planning (MGP)
  - Quality Function Deployment (QFD)

Roles & Responsibilities

Champions of Six-Sigma Initiatives

- Promote awareness and execution of Six Sigma within lines of business and/or functions
- Identify potential Six Sigma projects to be executed by **Black Belts** and **Green Belts**
- Identify, select, and support Black Belt and Green Belt candidates
- Participate in 2-3 days of workshop training

Black Belts

- Use Six Sigma methodologies and advanced tools (to execute business improvement projects
- Are dedicated full-time (100%) to Six Sigma
- Serve as Six Sigma knowledge leaders within Business Unit(s)
- Undergo 5 weeks of training over 5-10 months

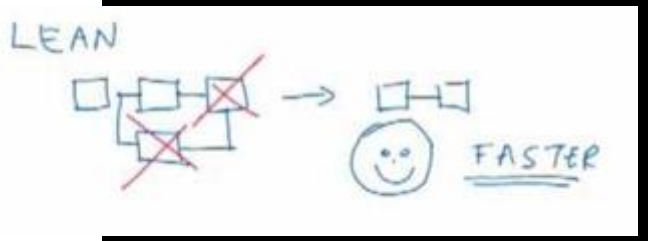
Green Belts

- Use Six Sigma DMAIC methodology and basic tools to execute improvements within their existing job function(s)
- May lead smaller improvement projects within Business Unit(s)
- Bring knowledge of Six Sigma concepts & tools to their respective job function(s)
- Undergo 8-11 days of training over 3-6 months

Other Roles

- **Subject Matter Experts**
  - Provide specific process knowledge to Six Sigma teams
  - Ad hoc members of Six Sigma project teams
- **Financial Controllers**
  - Ensure validity and reliability of financial figures used by Six Sigma project teams
  - Assist in development of financial components of initial business case and final cost-benefit analysis

Six-Sigma vs. Lean Methods



Six-Sigma Consultant?

Want to Initiative Six-Sigma Culture in your Organization?
Start with Identifying the Process you want to improve…

Thank You

Software Quality Management

- K G Krishna

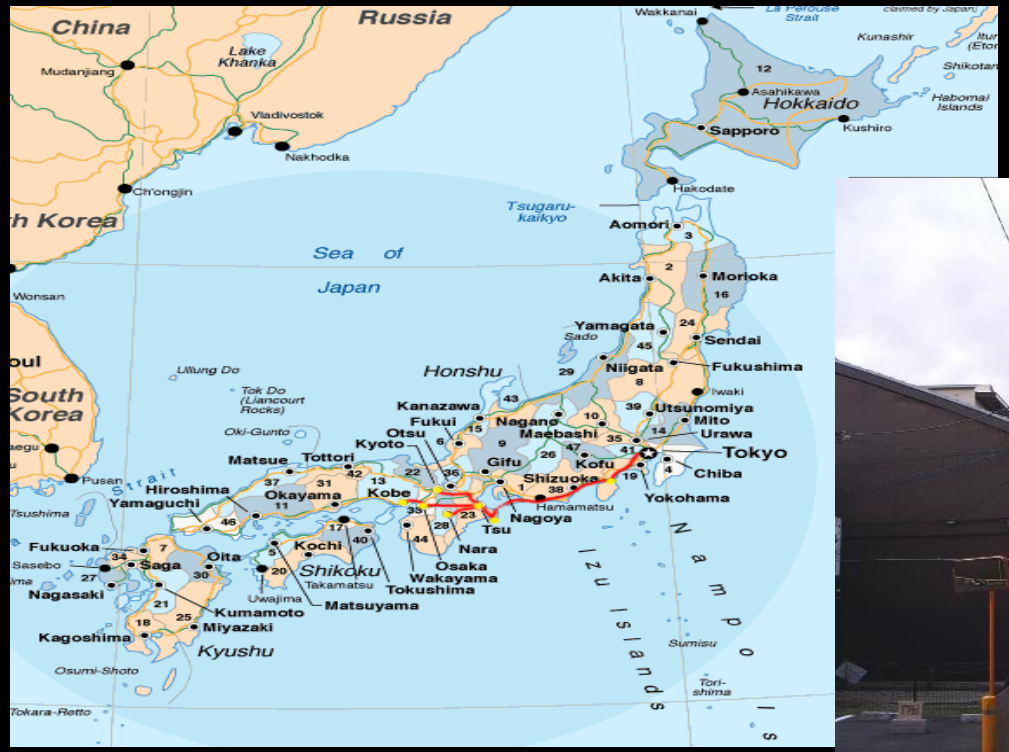
Indo/US-Japanese Cross-Cultural Differences ➔

Chammach
VS
Chopsticks

Cross-Cultural Lessons from
Indian and Japanese Software Project Mgmt.

KG KRISHNA

JAPAN: “Small is Beautiful” – Efficient and Organized



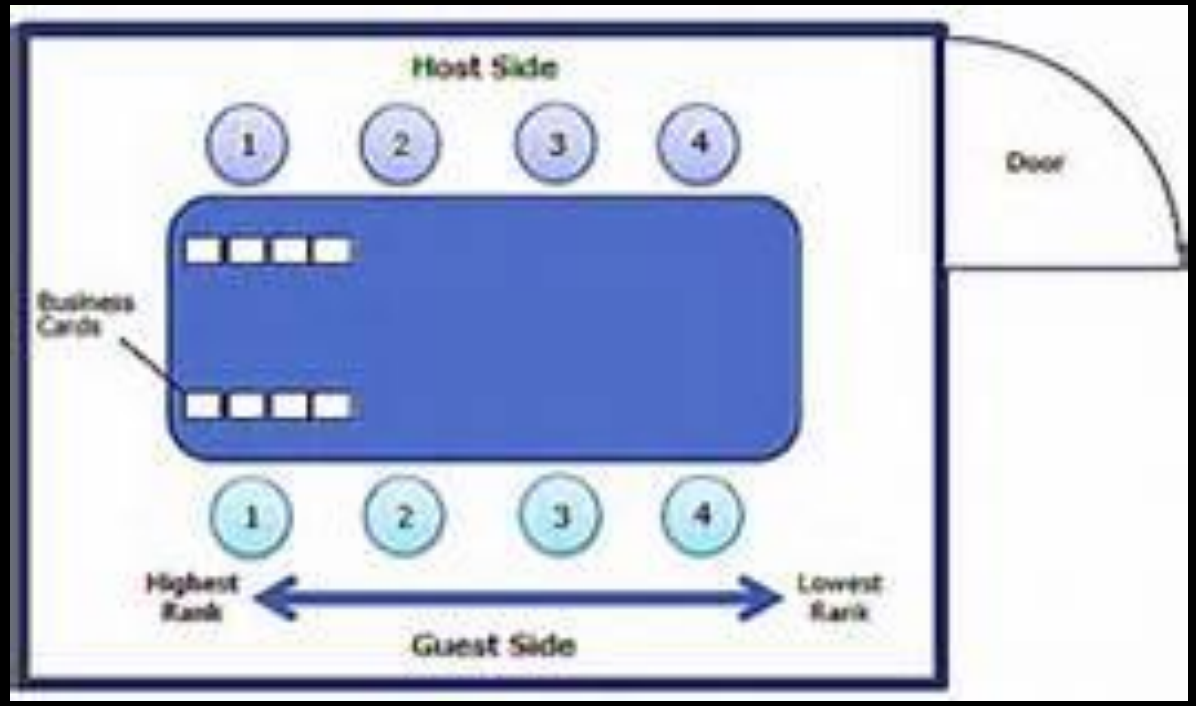
Kanzi Script: ‘Icon’ic and Rich in Context and Expression

Cultural Backdrop

- Dual Personalities (“Outer” / “Inner”)
- Integrated Work-Life
- Expect Conformity / Harmony in Group
- Mutual Trust/Respect
- Supplier/Vendor Relationship
- Honouring Commitments
- Consensus Driven
- Highly Observant
- Detail Oriented (Metrics)
- Extensive Note-Takers
- Visual/Graphics

Initiating Business & Personal Relationship

- Long-Drawn / Long-Term
- Based On Mutual Trust
- Require Patience (Testing the Waters)
- Pilot Project(s)
- Business Intermediary
- Scope for Negotiations
- Positive Attitude



Project Initiation

- Aptitutde & Attitude is the Key
- Do a LOT of Homework
- Honesty and Consistency
- Love Your Company/Country/Competition
- Build Personal Relationship

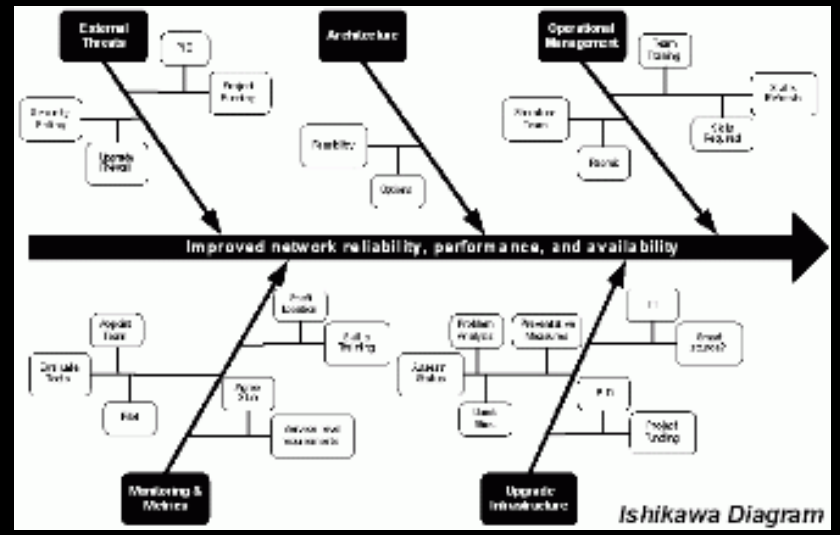
- “Requirements To Capture?” – Not Really!
- Ask Questions & Appreciate Answers
- You Don’t Get All Requirements in One-Go
- Onsite KT (Knowledge Transfer) is Just the Begining

Change Management!, Really?

- Requirements ALWAYS Change
- So is, Effort
- Deadlines (*Cut-off* Dates) are DEADLY.
- Cost/Budgets are FIXED
- Quality should be the HIGHEST

Project Communications

- No “English”
- Extensive Visuals / Graphics / Prototypes
- Adopt “Language of Metrics”
- Email / Fax / Personal Meetings
- Do NOT Force Decision Making
- Proactive Communication of Issues
- Leverage “Informal/After-Office” Channels
- “Issue – Backup” Plan
- Apologize. Apologize. Apologize



Project Delivery

- No Last-Minute Surprises
- Help “Save Face”
- Use “Informal” Channels
- Have a Trusted Negotiator
- Emphasize Relationship
- Focus on Analysis/Lessons-Learnt
- No Blame-Game
- Always Suggest ‘Way Forward’

Testing & QC

- Developers design, write test cases, and run test case.
- Statistical bug data is collected throughout product life cycle.
- Project Manager analyzes the data to control the project.
- Independent QA engineers Test Tools.

Only 2 Bugs / 10,000 Surface at Customer's Site !

% Bug Detection in Software Development Phases:

Code-after-release:	21.5
Unit-testing:	35.1
Subsystem-testing:	28.0
System Testing:	6.1
Inspection (QC Dept.):	9.3
At Customer Site:	0.02

Source: QA Statistics at Hitachi Software

Test-Case Design Guidelines

a. Approx. **1 test case per 5 - 15 LOC**

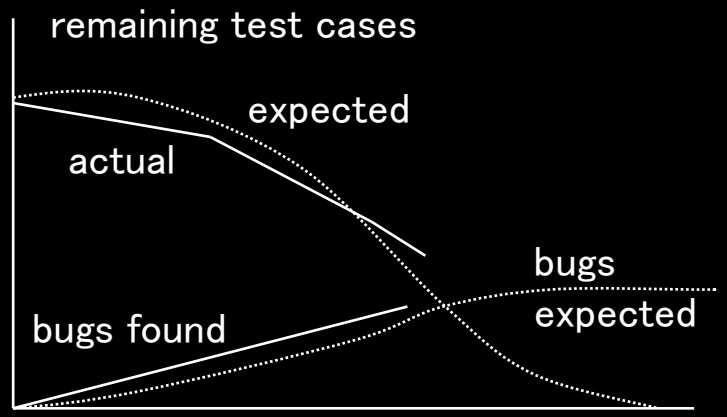
Language processor	1 test case per 8-12LOC
Online system	1 test case per 5-10 LOC
Batch system	1 test case per 10-15 LOC

b. Normal cases 60% or less,

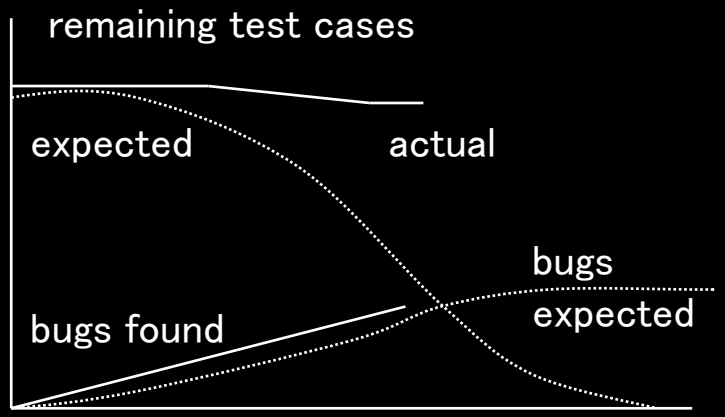
Abnormal cases	15% or more,
Boundary cases	10% or more,
Enviromental cases	15% or more

Source: QA Statistics at Hitachi Software

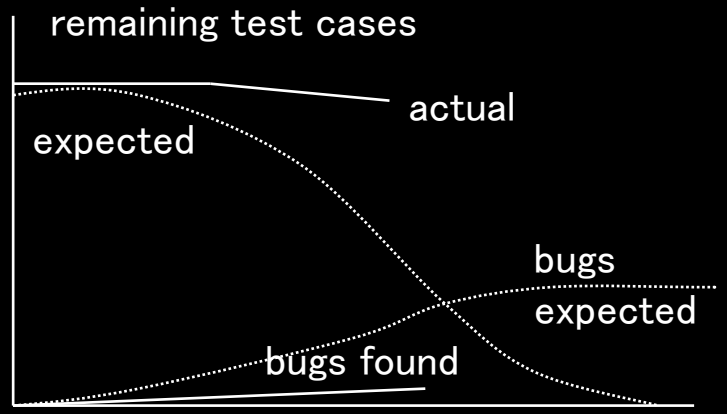
Good and Bad Testing



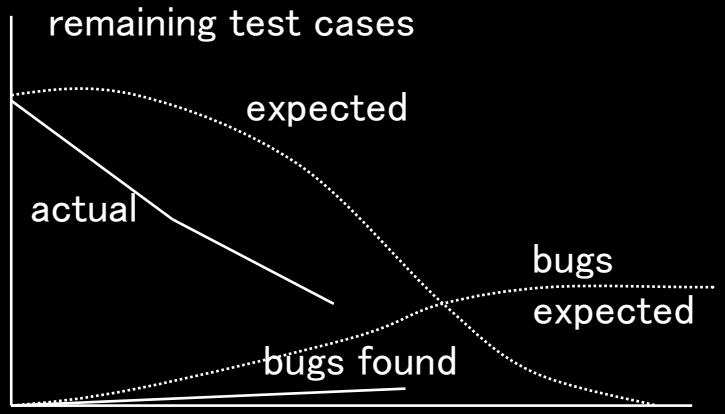
best case



low quality in previous process



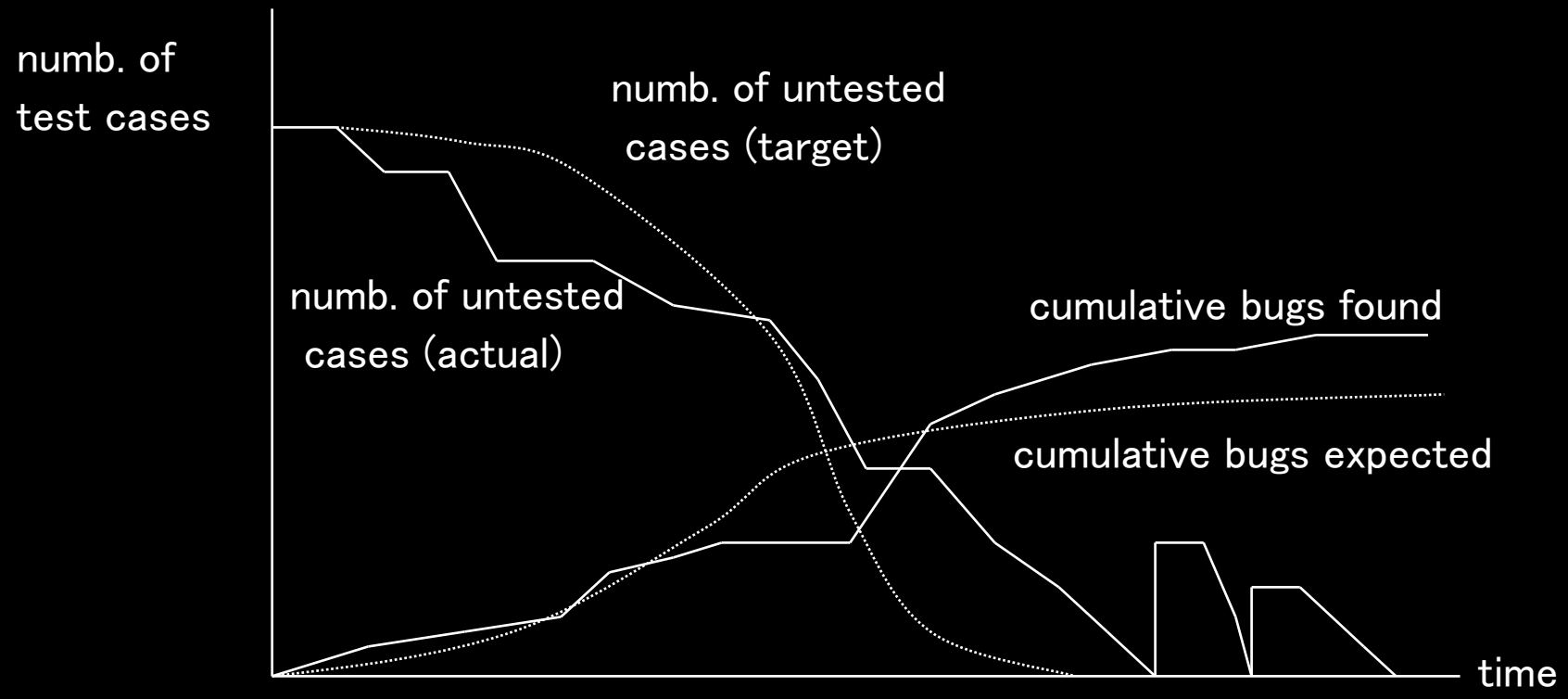
inexperienced testers



low test case quality

Source: QA Statistics at Hitachi Software

When To Release Software ?



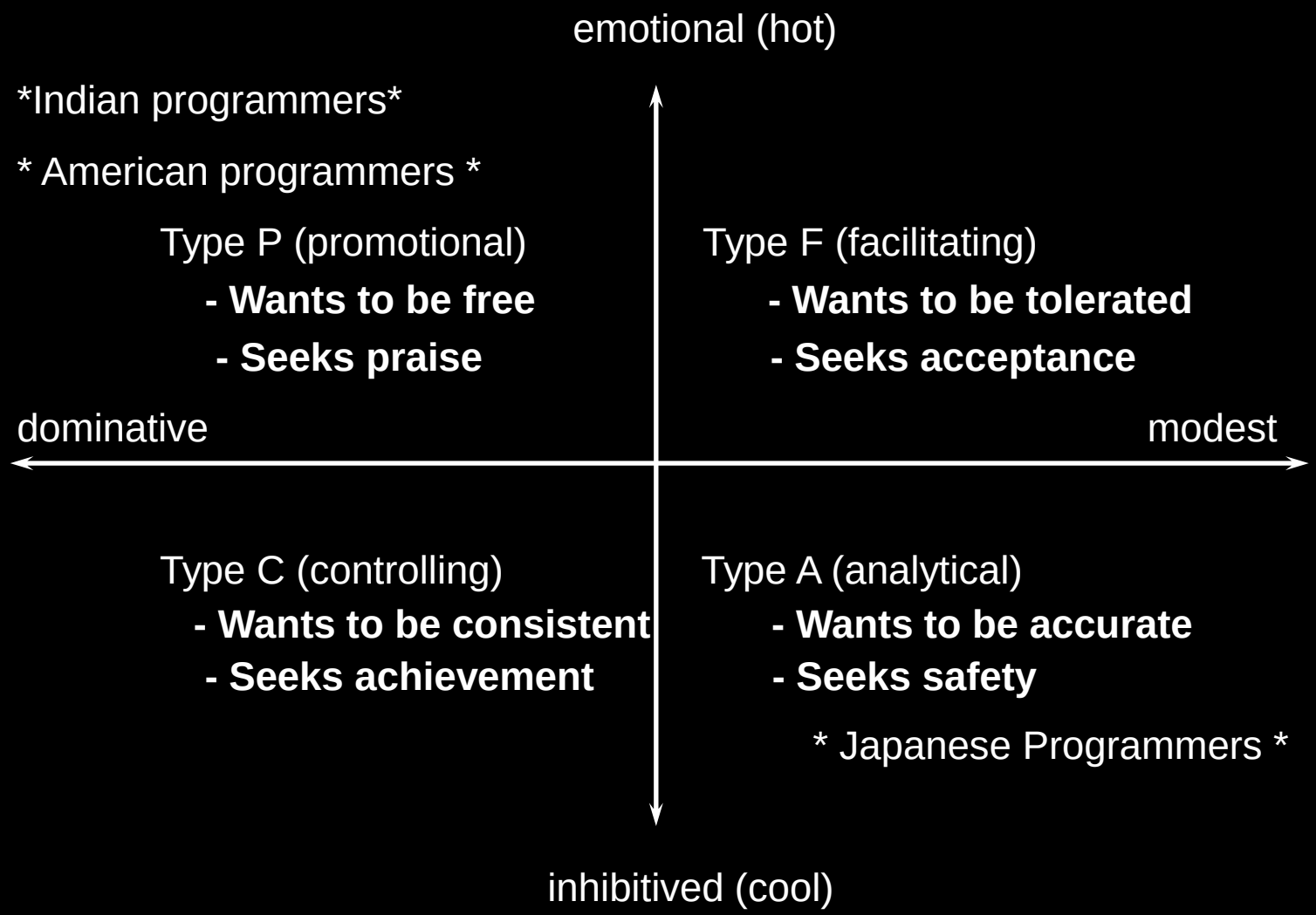
Source: QA Statistics at Hitachi Software

Developers Write Test-cases

- Density of Test-Cases Measure Programmer Competency
- Statistical Bug Data collection throughout the life cycle
- Bug Data Analysis by Project Manager
  - o Determine the best strategies to improve the quality
  - o Predict the release date
- Independent QA engineers Test Tools
  - o Approximately 8% of the entire engineers (330 out of 4,000).

Source: QA Statistics at Hitachi Software

Classification of Behaviours (Shepard's)



Source: QA Statistics at Hitachi Software

The Cost of Quality

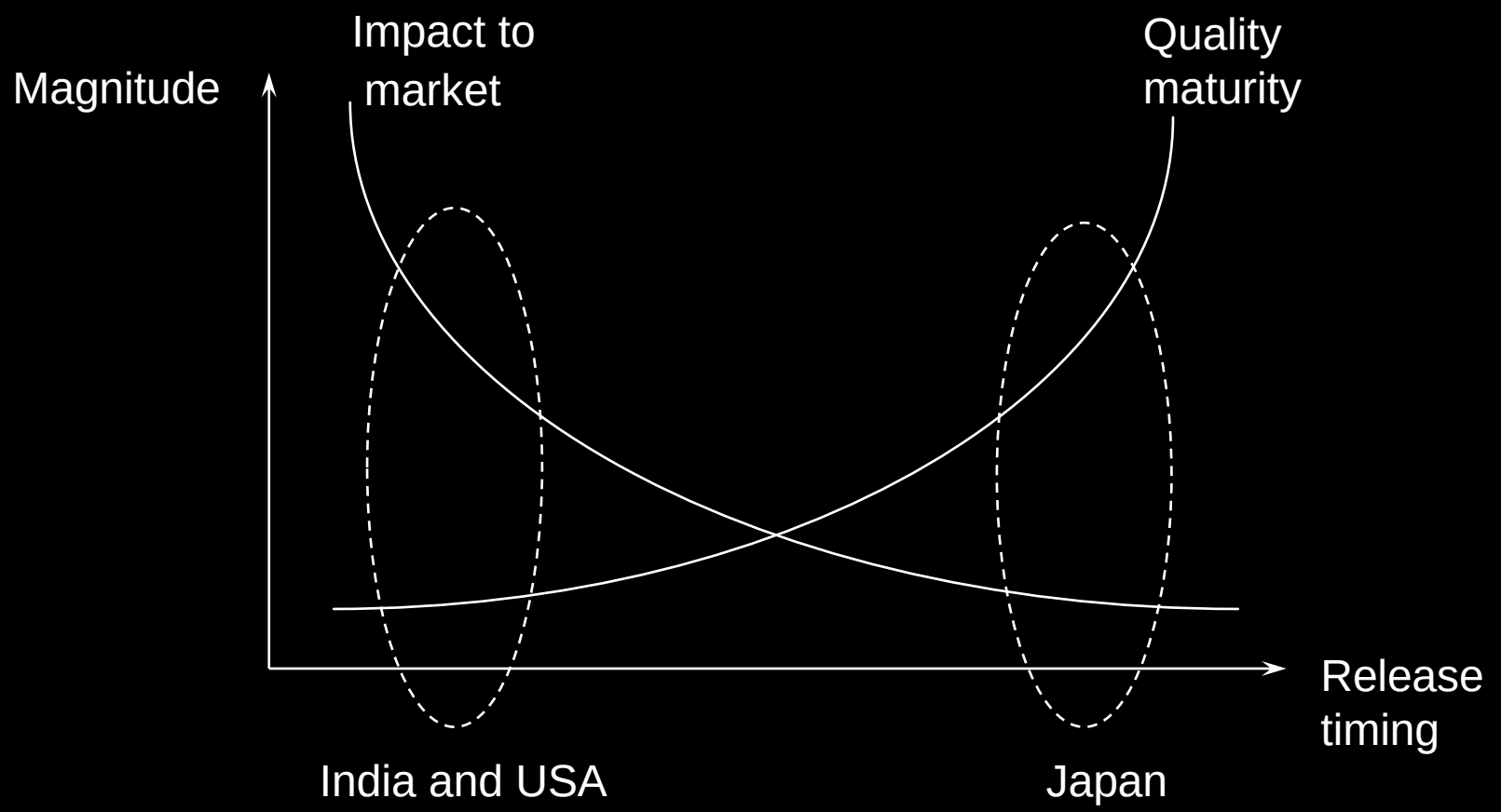
- **A 12-month project with 10 developers** (working on C-based software with 100,000 LOC using Water-Fall Life Cycle Model)

Requirement specs.	2 months
Design	3 months
Coding	2 months
Debugging	3 months
Testing (by QA engrs.)	2 months

- **Details of the 3-month debugging phase**
  - 100,000 LOC software needs 10,000 test cases (1,000 per developer).
  - 10 days to design 10,000 test cases (100 a day per developer)
  - 10 days to check 1,000 test cases in code inspection (10 a day per developer)
  - 40 days to check 10,000 test cases in machine debugging (25 a day per dvlp)

Source: QA Statistics at Hitachi Software

Differing Priorities/Strategies (Cultural)



Doing Business with Japanese: Summary

- ☑ Relationship First, Business Next
- ☑ Patience. Aptitude. Respect
- ☑ Continuous Improvement
- ☑ Honouring Commitments
- ☑ Constant Communications / Negotiations
- ☑ Speak Language of Metrics
- ☑ Analysis. Learnings. Way-Forward
- ☑ Apologize. Apologize. Apologize

- ☑ Relationship First, Business Next
- ☑ Patience. Aptitude. Respect
- ☑ Continuous Improvement
- ☑ Honouring Commitments
- ☑ Constant Communications / Negotiations
- ☑ Speak Language of Metrics
- ☑ Analysis. Learnings. Way-Forward
- ☑ Apologize. Apologize. Apologize

Arigatou gozaimasu.

ありがとうございます

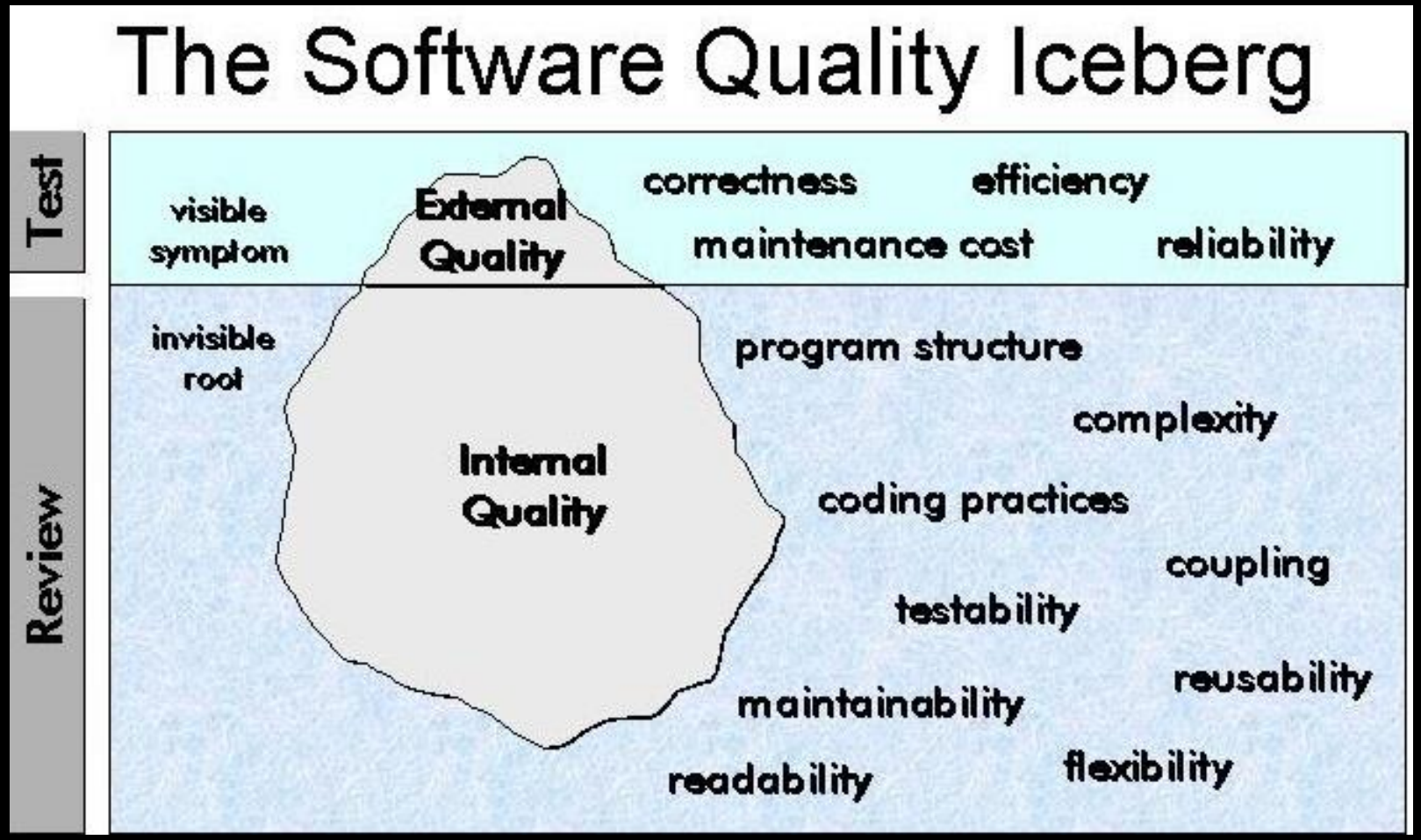
[thank you very much]

Software Quality Management

- K G Krishna

Summary of Lessons Learnt ➔

Software Quality – Several Perspectives!



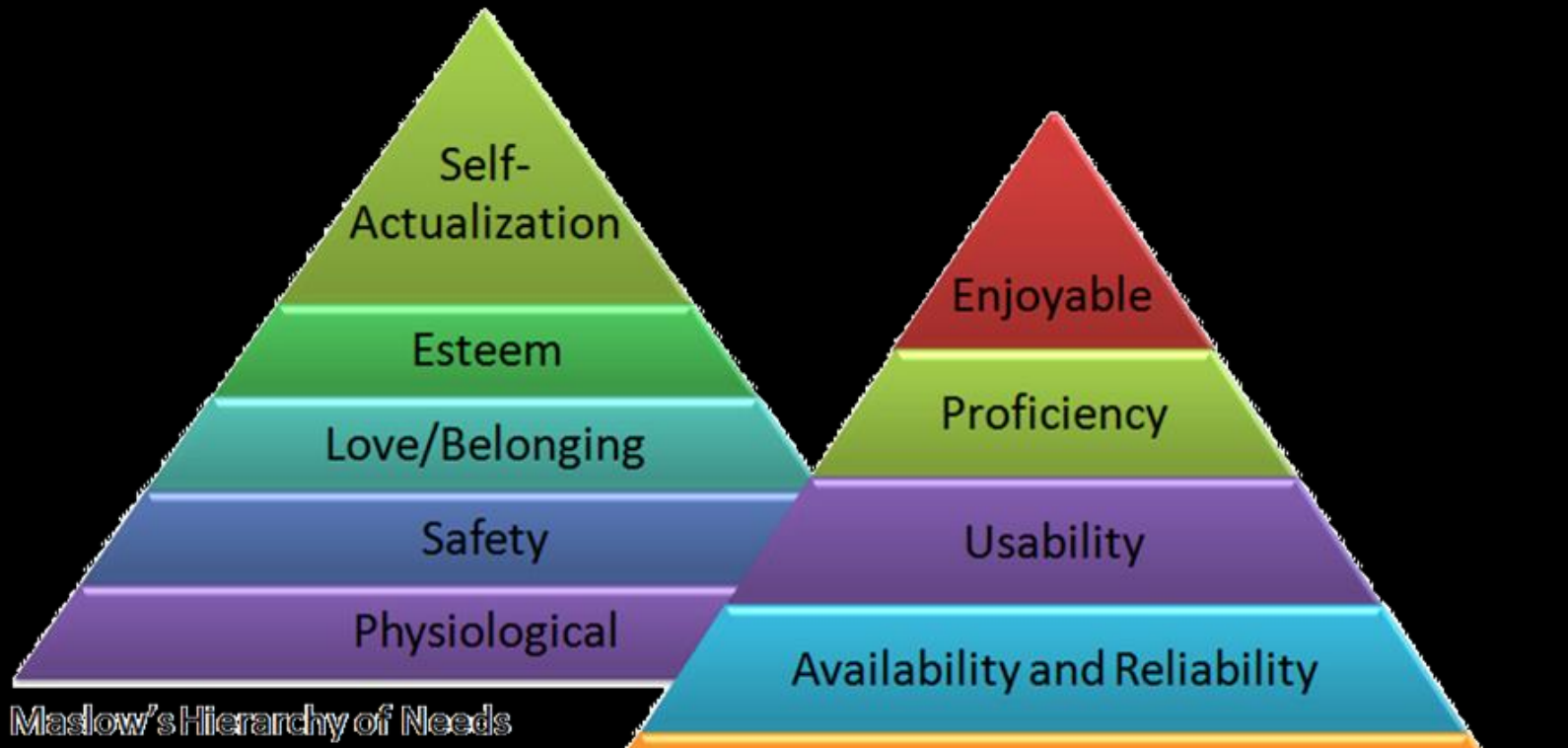
Source courtesy: http://software-quality.blogspot.in/

Software Quality: A Technical Definition

(**IEEE** 610.12-1990) Standard Glossary of Software Engineering Terminology:
"the degree to which a system. component, or process meets (1) specified requirements, and (2) customer or user needs or expectations“

(**ISO** 9003-3-1991) Guidelines for the application of ISO 9001 to the Development, Supply and Maintenance of Software:
"the totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs"

Software Quality *Maturity*: Correctness → Enjoyable

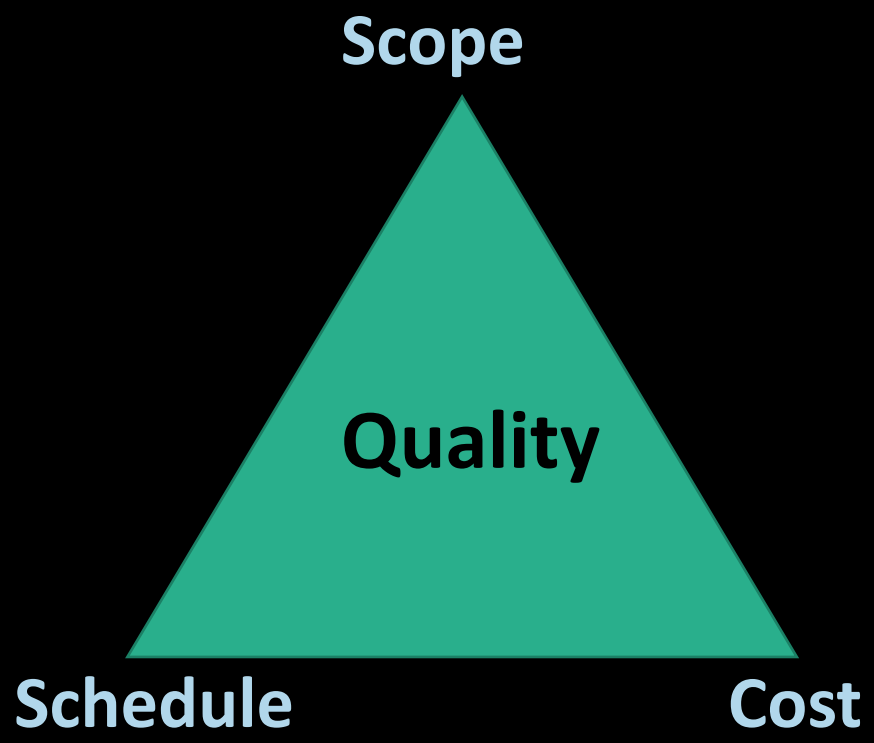


Source courtesy: http://sce2.umkc.edu/

QA vs. QC vs. Testing

http://www.highlandsinfotech.com/

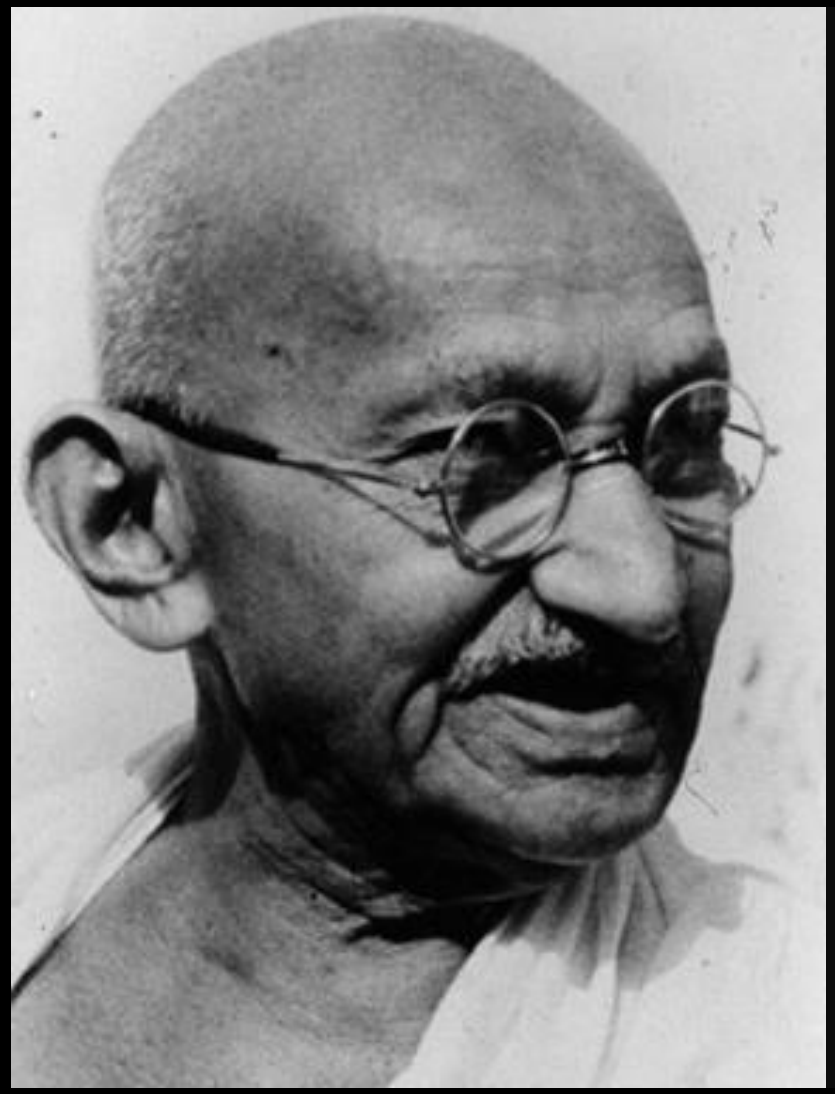
Quality: A Balancing Act by Project Manager



Quality Management Systems

Quality : A Culture!

- ✓ Doing Right The First Time
- ✓ Ensuring Simplicity in Everything We Think, Do and Communicate
- ✓ Putting Oneself in Customers' Shoes
- ✓ Quality is a Journey, Not a Destination
- ✓ Quality is Proactiveness
- ✓ Quality is paying Attention-To-Detail
- ✓ There is Never a 100% Defect-free Delivery
- ✓ "Quality means Doing it Right when no one is looking " – *Henry Ford*
- ✓ Quality is Self-Discipline



A customer is the most important visitor on our premises. He is not dependent on us. We are dependent on him. He is not an interruption in our work. He is the purpose of it. He is not an outsider in our business. He is part of it. We are not doing him a favor by serving him. He is doing us a favor by giving us an opportunity to do so.

— Mahatma Gandhi —

It is the quality of our work which will please God and not the quantity.

— Mahatma Gandhi —

Thank You.