08: Managing Quality

Software Project Management
Philippe Kruchten

Module outline

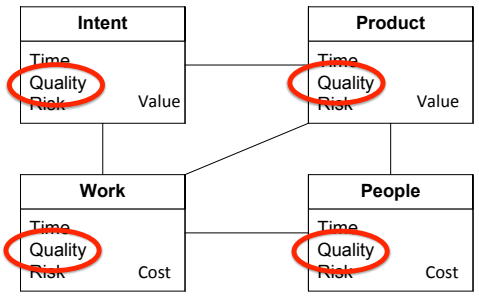
- What is software quality?
- Quality management
  - PDCA
- Defects, the measure on non-quality
- Measurement
  - GQM, ami
- Quality standards
  - IEEE, ISO, others

The Third Question

1. When will we be done? *(schedule)*
2. How much will it cost to get there? *(resource)*

3. Will they like it? *(quality)*
   - Will it be good enough that they will pay good money for it?
   - Will they buy more from us in the future?
   - Will they say good things about us….

Quality



Key issues

- Defining what is quality
- Quality, yes, but at what cost?
  "If quality is good, more quality must be better."
- Who is doing it? When?
- When do we stop? How do I know that we have "enough" quality?

Quality Defined

Traditionally:
- Quality of design, or construction
  - Good Craftmanship, good materials, etc…
- Quality of conformance
  - Norms, standards, codes

- Is this all?

- User satisfaction = compliant product + good quality + delivery within budget and schedule

GEQ: Good Enough Quality

- Not so bad
  "We're not dead yet"
  and as long as customers do not sue us….
- Positive infallibility
  "Anything we do is good"
  We do not even entertain negative thoughts…
- Righteous exhaustion
  "Perfection or bust"
  It is the effort that counts…

Source: James Bach

GEQ: Good Enough Quality (cont.)

- Customer is always right
  "Customers seem to like it"
  So, why bother more….
- Defined process
  "We use a good process"
  If the quality is not there, it cannot be our fault
- Static requirements
  "we satisfy all the written requirements."

Source: James Bach

GEQ: Good Enough Quality (cont.)

- Accountability
  "We fulfill our promises"
  Quality defined by contract.
- Advocacy
  "We made every reasonable effort."
- Dynamic tradeoff
  "We weigh many factors."

Source: James Bach

When failure is not an option

- Certain type of systems: safety-critical systems, which may endanger human life, certain types of failures are not an option.
  - Medical
  - Aviation, space
  - Transportation
  - Nuclear
  - Weapons

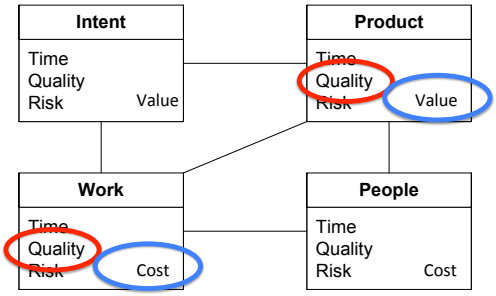
Quotes

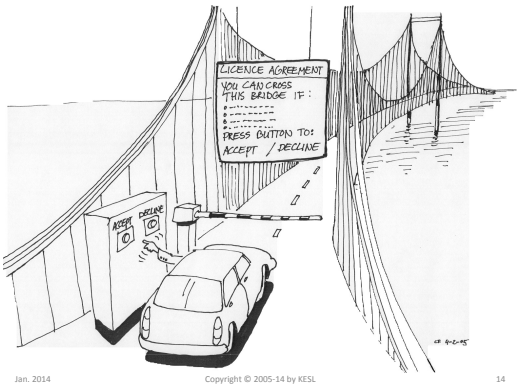
- At the leading edge, there are no subjects, no objects, only the track of Quality ahead, and if you have no formal way of evaluating, no way of acknowledging this Quality, then the train has no way of knowing where to go. Pirsig, 1977
- And what is good, Phaedrus,
  And what is not good—
  Need we ask anyone to tell us these things?
  Plato 370 BC

More quotes

- I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of meager and unsatisfactory kind. (Lord Kelvin, 1900)
- The difficulty in defining quality is to translate future needs into measurable characteristics, so that a product can be designed and turned out to give satisfaction at a price the user will pay. (Shewhart).

Quality = Value, at a Cost





Quality: Definitions from IEEE 610

Narrow view

- "The degree to which a system, component, or process meets specified requirements."
- "The degree to which a system, component, or process meets customer or user needs or expectations."

Broader view

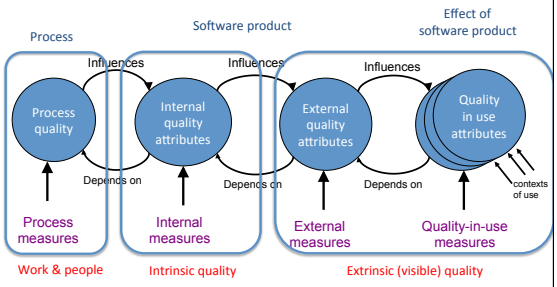
Taxonomy of Quality Attributes

- related to Product operation
  - Correctness Does it do what I want?
  - Reliability Does it do it all of the time? (see Robustness)
  - Efficiency Does it do it fast and cheap? (performance)
  - Usability Can we all run it, operate it?
  - Integrity Is it secure?
- related to Product revision
  - Maintainability can I fix it?
  - Testability can I test it? (=Verifiability)
  - Flexibility can I change it?
- related to Product transition
  - Portability can I use it somewhere else (CPU, OS)?
  - Reusability can I scavenge the software for product Z?
  - Interoperability will it work with X, Y, Z?

Alternate taxonomy: Grady's F URPS

- F Functionality
- U Usability
- R Reliability
- P Performance
- S Supportability
- yet another classification in Std: ISO 9126

The ISO 9126 Quality Model



ISO 9126 Quality model: internal & external

FRUEMP	
Functionality	Does it satisfy the users' needs?
Reliability	Can the software maintain its level of performance ?
Usability	How easy is it to use?
Efficiency	Relates to the physical resources used during execution?
Maintainability	Relates to the effort needed to make changes to the software?
Portability	How easily can it be moved to a new environment?

ISO 9126: Functionality

- Suitability
- Accuracy
- Interoperability
- Security

- *Functionality compliance*

ISO 9126: Reliability

- Maturity
- Fault-tolerance
- Recoverability

- *Reliability compliance*

ISO 9126: Usability

- Understandability
- Learnability
- Operability
- Attractiveness

- *Usability compliance*

ISO 9126: Efficiency

- Time behaviour
- Resource utilisation

- *Efficiency compliance*

ISO 9126: Portability

- Adaptability
- Installability
- Co-existence
- Replaceability

- *Portability compliance*

ISO 9126: Maintainability

- Analysability
- Changeability
- Stability
- Testability

- *Maintainability compliance*

ISO 9126: Quality in use

For specified users in a specific context:

- Effectiveness
- Productivity
- Safety
- Satisfaction

Example 1: recoverability metrics

- Availability: how available is the system for use during a specific period of time
  - X = To / (To + Tr) 0<=X<=1, closest to 1 is best
    - To = operation time, Tr = time to repair
  - Y = A1 / A2
    - A1 = # of cases of successful software use
    - A2 = number of cases of attempted use
- Mean down time:
  - Z = T / N, 0 < Z, the smaller the better
    - T = total down time, N number of observed breakdowns
- *Mean recovery time,*
- *Restorability, restartability,*
- *Restore effectiveness, etc. …*

Example 2: External understandability

- ….
- Demonstration effectiveness: What proportion of functions can the user operate successfully after a demonstration or tutorial?
  - Method: user observation, possibly with video camera
  - X = A / B
    - A number of functions operated successfully
    - B = number of demos/tutorial accessed
- Ease of learning how to perform a task in use: how long does the user take to learn how to operate a function efficiently?

More definitions: QC and QA

- Quality Control (QC)
  - The means by which we are going to evaluate (qualitatively or quantitatively) the Quality
  - reviews, test, surveys, statistical control, etc.
- Quality Assurance (QA)
  - QA provides management with the necessary data to be informed about product quality, to gain insight and confidence that quality goals will be met
  - Including identification of problems affecting quality

Cost of Quality

- Prevention costs
  - Quality planning
  - Formal reviews
  - Test equipments
  - Training
- Appraisal costs
  - Testing
- Internal failure cost
  - Rework
  - Repair

Cost of Quality

- External failure costs
  - Complaint resolution
  - Product return and replacement
  - Help line support
  - Warranty work
  - Loss of future sales ….

Measurement

- How do we measure quality?
- Easier to measure non-quality!
  - Problems, defects
  - Non conformance (to specs or standards)
  - Delays, cost overrun
- Return on investment?
- Customer satisfaction?

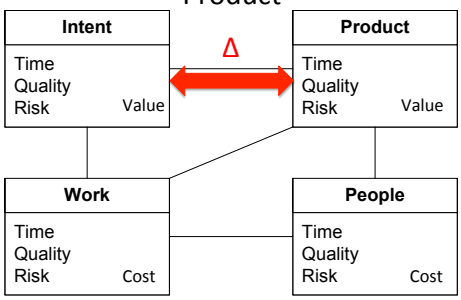
Conformance to Standards (Compliance)

- Not an assurance of success
- But it does help in some context
- Standards have captured some good practice, some common wisdom.
- They provide more objectives ways to assess a situation
  - especially when multiple parties are involved

Key concept: Defect

- A defect is something that causes the software to behave in a manner that is inconsistent with the requirements or the needs of the user (or customer).
- An anomaly, or flaw, in a delivered work product. Examples include : omissions and imperfections found during early lifecycle phases and symptoms of faults contained in software sufficiently mature for test or operation. A defect can be any kind of issue you want tracked and resolved

Defects: the gap between Intent and Product



Defects

Process focus

- We can either:

1. Avoid inserting defects in the system
2. Remove the defects we have (nevertheless) inserted

Product focus

Avoid inserting defects in the system

- Excellent process
- Automation of tedious or error prone tasks
- Highly educated people
- Very sophisticated tools

- Simple systems (?)

- Example: Cleanroom process (Harlan Mills)

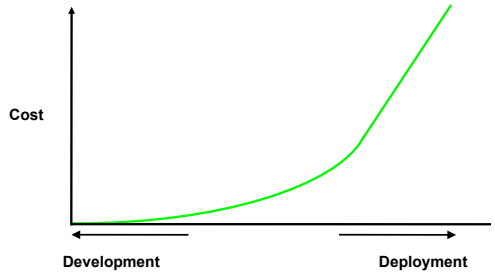
Removing defects from the system

- The earlier we remove them, the better, and the cheaper.

- Reviews, inspections, audits, ….

- Testing, debugging

Cost of fixing defects



Quantitative vs. procedural approach

- We can ask: “Have we done all the right things?”

- Or we can estimate how many defects there are left
  - but can we?

  - There are models for software reliability
  - But they are not very reliable ….!

Defect Removal Efficiency

- Percentage of existing total defects detected by a certain class of quality control activities, in some part of the lifecycle.

Quality Management PDCA

- Plan
- Do
- Check
- Act

- at least once per iteration

Quality Management: Planning (P)

- Setting quality objectives
- On the Product:
  - Selecting key measures for specific quality attributes
    - Performance, etc.
    - Code quality
    - Customer Satisfaction
  - Monitoring defects:
    - Quantity of remaining defects
    - Type of defects
- on the Process:
  - Productivity (velocity)
  - Test coverage
  - Defect removal efficiency
  - Schedule
  - Resource consumption

D = Do

- Do reviews
- Conduct test campaigns, customer surveys, …
- Iterative development helps
  - You can organize multiple waves of testing
  - See trends

C = Check

- Gather information on effectiveness of
  - reviews
  - tests
- Compare quality data collected with objectives
  - Identify discrepancies
  - investigate cause of discrepancy

A = Act

- Improve process
  - Guidelines: programming, design, UI
  - Checklists
- Tooling
- Training
- Replanning or resetting expectations

Managing defects

- Problem reports
- Defects
- Defects attributes:
  - severity
  - cause (and likelihood)
  - consequence
  - *workaround*

Severity

- A simple scale is sufficient

1: major defect (“show stopper”)
  - System crashes
  - Key functions not usable

2: serious user annoyance
  - some functions not usable

3: Minor annoyance

4: Request for enhancement

Severity (cont.)

- Criteria must be refined for each project
- Context is different
- Sensitivity is different

- Clear definitions avoid lots of late arguing when things get tough.

- IEEE 1044 (?) “anomalies”

Defects Must Be Actively Managed!

- Use a simple database
- Spreadsheet for small projects

- Bugzilla
- Rational ClearQuest
- Jira
- Trak
- Microsoft DDTS
- etc….

Monitoring defects

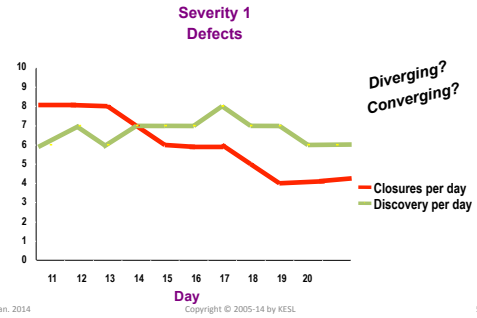
- Track weekly at least:

1. Defects found (by severity)
2. Defects closed (by severity)

- and trend (rate over time)

- Additionally
  - Defect density for major subsystems
  - Instability of some subsystems

Tracking Defects, day after day



Orthogonal Defect Classification (ODC)

- IBM, circa 1992, Ram Chillarege

- ODC captures and turns semantic information in the software defect stream into a measurement on the process.

ODC

Opener Section (These attributes are usually available when the defect is opened.)			Closer Section (These attributes are usually available when the defect is fixed.)				
Defect Removal Activities	Triggers	Impact	Target	Defect Type	Qualifer	Age	Source
Design Rev, Code Inspection, Unit Test, Function Test, System Test.	1. Design Conformance 2. Logic/ Flow 3. Backward Compatibility 4. Lateral Compatibility 5. Concurrency 6. Internal Document 7. Language Dependency 8. Side Effect 9. Rare Situations 10. Simple Path 11. Complex Path 12. Coverage 13. Variation 14. Sequencing 15. Interaction 16. Workload/Stress 17. Recovery/Exception 18. Startup/Restart 19. Hardware Configuration 20. Software Configuration 21. Blocked Test (previously Normal Mode)	1. Installability 2. Serviceability 3. Standards 4. Integrity/Security 5. Migration 6. Reliability 7. Performance 8. Documentation 9. Requirements 10. Maintenance 11. Usability 12. Accessibility 13. Capability	Design/Code	1. Assign/Init 2. Checking 3. Alg/Method 4. Func/Class/Object 5. Timing/Serial 6. Interface/O-O Messages 7. Relationship	1. Missing 2. Incorrect 3. Extraneous	1. Base 2. New 3. Rewritten 4. ReFixed	1. Developed In-House 2. Reused From Library 3. Outsourced 4. Ported

Defect Removal Activities	Triggers	Impact
Design Rev, Code Inspection, Unit test, Function Test, System Test.	1. Design Conformance 2. Logic/ Flow 3. Backward Compatibility 4. Lateral Compatibility 5. Concurrency 6. Internal Document 7. Language Dependency 8. Side Effect 9. Rare Situations 10. Simple Path 11. Complex Path 12. Coverage 13. Variation 14. Sequencing 15. Interaction 16. Workload/Stress 17. Recovery/Exception 18. Startup/Restart 19. Hardware Configuration 20. Software Configuration 21. Blocked Test (previously Normal Mode)	1. Installability 2. Serviceability 3. Standards 4. Integrity/Security 5. Migration 6. Reliability 7. Performance 8. Documentation 9. Requirements 10. Maintenance 11. Usability 12. Accessibility 13. Capability

Jan. 2014 56

Target	Defect Type	Qualifer	Age	Source
Design/Code	1. Assign/Init 2. Checking 3. Alg/Method 4. Func/Class/Object 5. Timing/Serial 6. Interface/O-O Messages 7. Relationship	1. Missing 2. Incorrect 3. Extraneous	1. Base 2. New 3. Rewritten 4. ReFixed	1. Developed In-House 2. Reused From Library 3. Outsourced 4. Ported

Jan. 2014 Copyright © 2005-14 by KESL 57

Defects Database: What to track? Why?

- Exercise:
  - what attributes should we track for a defect and why?

Jan. 2014 Copyright © 2005-14 by KESL 58

Am I ready to "ship"?

- When?

  Absolute level of quality?
  Efficiency of defect finding/fixing is getting low?

- Estimating residual defects:
  - Trend (rate of discovery and correction)
  - Defect density: defects / KSLOC
  - Defect pooling
  - Defect seeding
- Warning: only works with large systems (statistical effects)

Jan. 2014 Copyright © 2005-14 by KESL 59

Technique: Defect Density

- OrcaSys V1: 100 KSLOC
  - 650 defect before release
  - 50 after release
- OrcaSys V2: added 50K SLOC
  - 400 defect before release
  - 75 after release
- OrcaSys V3: added another 100KSLOC
  - detected 600 defect so far
  - Quality objective is 95%
  - are you ready to ship?

Jan. 2014 Copyright © 2005-14 by KESL Source: S.McConnell 60

Technique: Defect Density (cont)

- OrcaSys V1: 100 KSLOC
  - 700 defect s=> density = 7 defect / KSLOC
- OrcaSys V2: added 50K SLOC
  - 475 defects => density >9.5 defect/KSLOC
- OrcaSys V3:
  - with a defect density of 7 to 10 you are looking at a total of 700 to 1000 defects in version 3.
  - To find 95% you are aiming at 650 to 950 defects before release
  - you have found only 600 … still some ways to go.

Jan. 2014 Copyright © 2005-14 by KESL Source: S.McConnell 61

Technique: Defect Pooling

- Create 2 pools of defects (A and B)

$Defects_{Unique} = Defects_A + Defects_B - Defects_{A\&B}$

$Defects_{Total} = (Defects_A \times Defects_B) / Defects_{A\&B}$

Technique: Defect Pooling (cont.)

- OrcaSys V3:
  - Pool A: entered by people whose initial is A-K
  - = 400 defects
  - Pool B: entered by people whose initial is L-Z
  - = 350 defects
  - Common defects (exist in A and B): 150
- Unique defects: 400 + 350 – 150 = 600
- Estimate of total defects:
  - (400 x 350) / 150 = 933

Technique: Defect Seeding

- Infect the software with a sample of representative defects
- Check the defects found and the proportion of seeded versus indigenous

$IndigenousDefects_{Total} = (SeededDefects_{Planted} / SeededDefects_{Found}) * IndigenousDefects_{Found}$

Technique: Defect Seeding (cont.)

- OrcaSys V3:
  - You plant 50
  - 31 of the 50 have been found on top of 600 indigenous defect

You are probably looking at
50 / 31 * 600 = 967 defect total

Combine techniques

- OrcaSys V3:
  - seeding: 967
  - Pooling: 933
  - defect density: 650 - 950
  - trend: 800
- Precision is not important: with 600 so far you are not ready to deliver.

Technique: Review

- Requirements review
- Design reviews
- Code reviews

- How to organize a review?
- How effective are reviews?
- Alternatives?

- Reviews, inspections, walkthroughs, audits

Technique: Review (cont.)

- Fagan inspections
  - Moderator
  - Two inspectors or readers
  - Authors
- Identify potential faults
- Use some checklist

Reviews?

- Integral part of CMM level 3 and up
- Effectiveness not clear
- Drawback:
  - Time-consuming: meeting
  - Shallow
  - Morale
- Advantages:
  - Morale
  - Mentoring effect

Technique: Pair Programming

- How to organize it?
- Advantages
- Drawbacks
- Effectiveness and efficiency still being debated

Other things to know about SW Quality

- Surveys
- Measurement
- GQM = Goal Question Metric
- TQM = Total Quality Management
- Kaizen, etc.
- ISO 9000 Quality Standards
- Six Sigma movement

More quotes

- I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of meager and unsatisfactory kind. (Lord Kelvin, 1883)
- The difficulty in defining quality is to translate future needs into measurable characteristics, so that a product can be designed and turned out to give satisfaction at a price the user will pay. (Shewhart).

Measurement

- Halstead, and the "software science" (1970's)
  - Code complexity
- McCabe and the Cyclomatic number:
  - C = e – n + p
- Later: OO measures
- Many things to measure, but little net results on quality
  - Automate is not the answer

Goal Question Metric approach

- Vic Basili, U. of Maryland, 1985-1992
- A fresh wind of sanity
- Asking "why?"
  - Why do we measure?
  - How do we relate measures to quality or other useful goals?

Effective measurement

- Focused on specific goals
- Applied to all life-cycle product, processes and resources
- Interpreted based on characterization and understanding of the organizational context, environment and goals

GQM

- Goal: Conceptual level
  - Object of measurement (product, process, resources) point of view, quality attribute
- Question: Operational Level
  - Characterize the object relative to a quality attribute
  - relate it to the goal
- Metric: Quantitative level
  - set of data associated with the question

GQM : Example

- Goal:
  Improve the timeliness of change request processing from the Project manager viewpoint
  - Purpose: improve
  - Issue: timeliness
  - Object: change request (process)
  - Viewpoint: project manager

GQM Example (cont.)

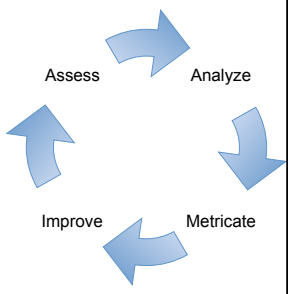
- Question 1: What is the current change request speed
- Metrics 1:
  - average cycle time
  - standard deviation
  - % above upper limit
- Question 2: Is the performance improving?
- Metrics 2:
  - current average cycle time/ baseline average cycle time
  - Subjective rating of manager's satisfaction

GQM: Example

- More questions:
  - Is the change management process documented?
  - Is the actual change process the one that is documented?

The ami approach

- Kevin Pulford, Annie Kuntzmann-Combelles, et al.
- European Esprit project AMI
- Expand on the original GQM
- Detailed step-by-step process



Sounds like Deming again....

The ami approach

- Assess
  1. assess your environment
  2. define primary management goals
     - knowledge goals
     - change or achievement goals
  3. check the goals against the assessment

  – Inputs: plans, business goals, audits, previous measurements
  – Support: assessment framework (CMM, SPICE, ISO 9000)

The ami approach (cont.)

- Analyze
  4. Break down management goals into subgoals
     - identify entities and attributes
  5. check consistency of resulting goal-tree
  6. Identify metrics from subgoals
     - qualitative, quantitative?
     - Something that can be measured: # of defect, # lines, of code, etc…

  – Input: goals
  – Support: templates, procedures, tools

The ami approach (cont.)

- Metricate
  7. Write and validate a measurement plan
     Objectives
     Metrics definition, collection procedure
     Primitive metric, metric
     Responsibilities, time scales
  8. Collect primitive data
  9. Verify primary data

  – Input: goals, questions, metrics,
  – Support: templates, procedures, tools

The ami approach (cont.)

- Improve
  10. Distribute, analyze and review measurement data
  11. Validate the metrics
  12. Relate data to the goals and implementation actions

  – Inputs: Measurement plan and collected data
  – Support: guidelines for presenting data

Metric definition template

- Name
- Definition
  – If not primitive, how calculated
- Goals
  – Goals and questions that are related to this metric
- Analysis procedure
  – Range, threshold, calibration, standards
  – Models and tools
- Responsibility
  – Who collects, aggregate, analyze, report

Example:

- OrcaSys V1 is not very good…large number of problems reported by the registrar's office.
- Goal 1: gain better understanding of product quality
- Goal 2: improve quality as perceived by user
- Subgoal1: understand how the different part of the system affect quality
- SG2. Gain a measure of the quality of delivered software
- SG3: understand how the different part of the development process affect quality

Example (cont.)

- Metric 3.1: Customer problems traceable to a problem in the requirements specification
- Metric 3.2: same to Design
- metric 3.3: …
- ….
- Metric 3.6 number of defects detected by reviews

Example (cont.)

- M6: Error distribution
- Definition: Shows how the defects reported by the customer are attributed to various parts of the process
- Goal: associated to subgoal 2: understand where errors are introduced
- Analysis: metric extracted from Defect database, using attribute: root-cause {req, analysis, design, code, test, doc, other}

Example (cont.)

- OrcaSys V2, week 23

	Severity 1	Severity 2	Severity 3
Req	2	18	34
Analysis	0	2	12
design	6	23	56
Code	4	97	156
Test	0	2	4

Example (cont.)

- OrcaSys V2, week 23

	total	percentage	Year ago
Req	54	9.0%	12.6 %
Analysis	14	2.3 %	8.2 %
Design	85	14.2 %	10.5 %
Code	257	42.8 %	34.1 %
Test	4	<1 %	3.2 %
…	…	…	

Presenting data

- Tables
- Histograms
- Pie charts
- Scatter plots

- Trends (over time) are often more important than absolute values

What to measure? (in general)

- Management indicators
  - Work and progress (over time)
  - Budget cost and expenditures (over time)
  - Staffing and turnover (over time)
- Quality
  - Change traffic and stability (over time)
  - Breakage and modularity (av. breakage per change)
  - Rework and adaptability (av. rework per change)
  - MTBF and maturity (defect rate over time)

Walker Royce, 1999

RUP: try to focus on:

- Modularity:
  - Scrap and rework are localized
- Adaptability:
  - Rework (effort for resolution)
- Maturity
  - Defect frequency, over time
- Maintainability:
  - productivity in maintenance relative to development

Source: IBM 2003 & Walker Royce 1999

Collected data (primitive metrics)

- Total SLOC
- Total effort
- Configured SLOC
- Software change orders (SCO) of different level
- Open rework: B=SLOC broken by SCO
- Closed rework: F=cumulative fixed SLOC
- Rework effort: E
- Usage time: UT

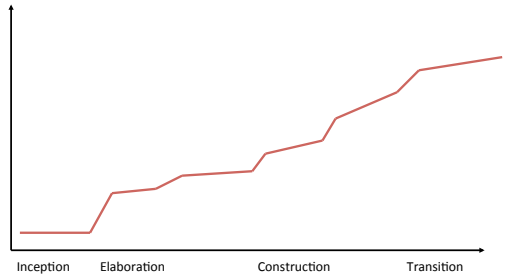
Software Change Order vs. Defect

- Software change order
  0. Critical
  1. Important, non critical
  2. Minor and improvements
  3. Change in requirements, new features

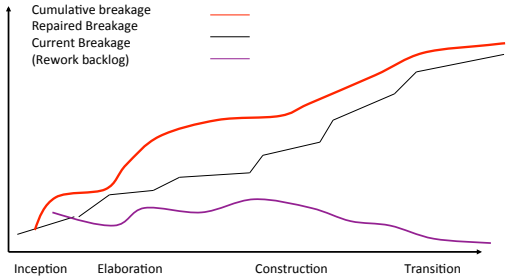
Metrics

- scrap ratio: B/ SLOCt
- Rework ratio: B/Effort
- Modularity: B/N (average breakage per SCO)
- Adaptability: E/N (average effort per SCO)
- Maturity: UT/(SCO0+SCO1) = meantime between defect
- Maintainability: (scrap ratio)/(rework ratio)

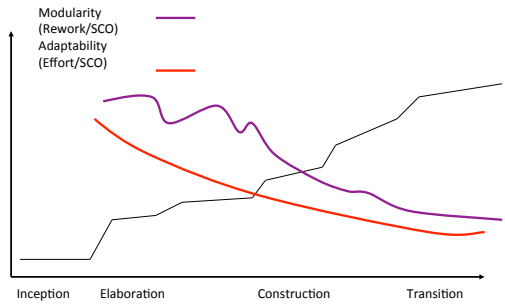
Total SLOC



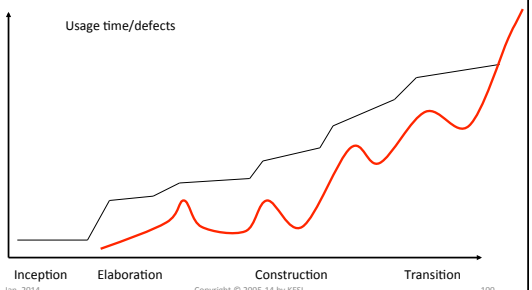
Breakage



Modularity and Adaptability



Maturity



(Too?) Many Standards on SW Quality

- IEEE 730: Software Quality Assurance Plan
  - IEEE 730.1 Guide for SQA Planning
- IEEE 982: Measures for reliable software
- IEEE 1044: Classification of anomalies
- IEEE 1061: Software Quality Metrics Methodology
- ISO 9000 family
- And now : ISO25000 series

IEEE 730: Software Quality Assurance Plan

1. Purpose
2. References
3. Management
4. Documentation
5. Standards, practices, conventions, metrics
6. Reviews and audits
7. Test
8. Problem reporting and corrective actions
9. Tools, techniques, methodologies
10. Code control
11. Media control
12. Supplier control
13. Record collection
14. Training
15. Risk management

Most relevant is ISO 9000-3

- Guidelines for Applying ISO 9001 1994 to Computer Software
  - Use ISO 9000-3 if you develop, supply, install, or maintain computer software
  - Paraphrases 9001 (but hard to read ☹)
- Main messages of 9000-3:
  - Manage quality proactively
  - Develop a quality system and document it
    - Plans, procedures, reviews, metrics, ….
  - Collect and manage data and information relative to the application of the quality system

Old and current fads and buzzwords

- TQM: Total Quality Management
  - 1950's, Deming (again)
- QFD : Quality Function Deployment
  - House of Quality
  - Originated in Japan: Yoji Akao, circa 1972
  - not software specific
- Six Sigma
  - Manufacturing
  - Applicability to software highly dubious

TQM Philosophy, in 14 steps

1. Create constancy of purpose for improvement of product and service. Constancy of purpose requires innovation, investment in research and education, continuous improvement of product and service, maintenance of equipment, furniture and fixtures, and new aids to production.
2. Adopt the new philosophy. Management must undergo a transformation and begin to believe in quality products and services.
3. Cease dependence on mass inspection. Inspect products and services only enough to be able to identify ways to improve the process.
4. End the practice of awarding business on price tag alone. The lowest priced goods are not always the highest quality; choose a supplier based on its record of improvement and then make a long-term commitment to it.

TQM

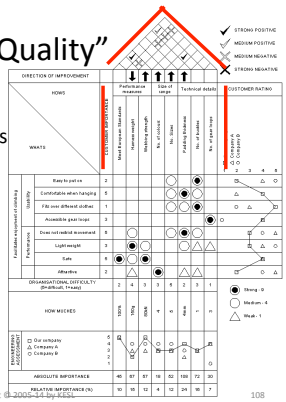
5. Improve constantly and forever the system of product and service. Improvement is not a one-time effort; management is responsible for leading the organization into the practice of continual improvement in quality and productivity.
6. Institute training and retraining. Workers need to know how to do their jobs correctly even if they need to learn new skills.
7. Institute leadership. Leadership is the job of management. Managers have the responsibility to discover the barriers that prevent staff from taking pride in what they do. The staff will know what those barriers are.
8. Drive out fear. People often fear reprisal if they "make waves" at work. Managers need to create an environment where workers can express concerns with confidence.
9. Break down barriers between staff areas. Managers should promote teamwork by helping staff in different areas/departments work together. Fostering interrelationships among departments encourages higher quality decision-making.

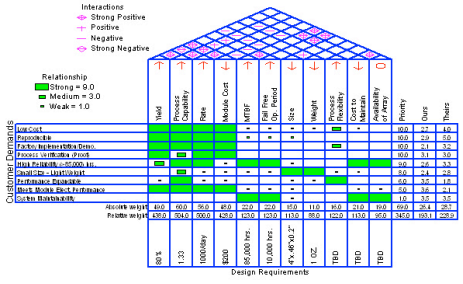
TQM

10. Eliminate slogans, exhortations, and targets for the workforce. Using slogans alone, without an investigation into the processes of the workplace, can be offensive to workers because they imply that a better job could be done. Managers need to learn real ways of motivating people in their organizations.
11. Eliminate numerical quotas. Quotas impede quality more than any other working condition; they leave no room for improvement. Workers need the flexibility to give customers the level of service they need.
12. Remove barriers to pride of workmanship. Give workers respect and feedback about how they are doing their jobs.
13. Institute a vigorous program of education and retraining. With continuous improvement, job descriptions will change. As a result, employees need to be educated and retrained so they will be successful at new job responsibilities.
14. Take action to accomplish the transformation. Management must work as a team to carry out the previous 13 steps.

"House of Quality"

- Elaborate technique to analyze customer needs and perceptions
- A final diagram to combine results, document information, perceptions and decisions





Summary

- Software quality
  - Many dimensions & attributes
  - What is "good enough"?
- Measurements
  - many proposed metrics, few are easy to correlate to quality
  - "software science" not here yet
- Goal Question Metric:
  - a top down approach
- Many standards:
  - IEEE 730
  - ISO 9000-3

Quality