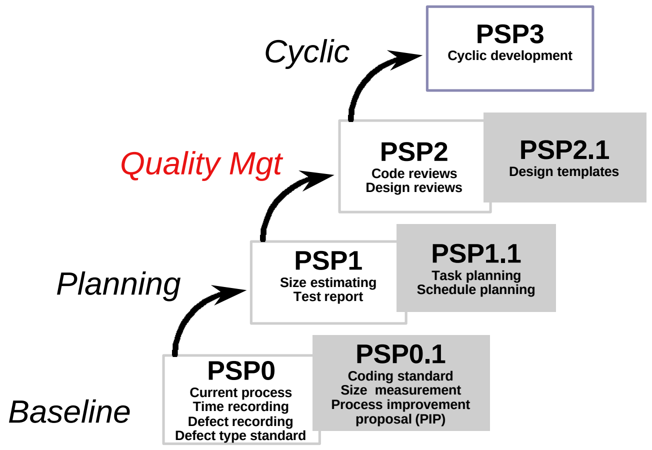
Software Quality Management

Outline

- Review of PSP Levels
- Overview
- SW Quality Economics
- Developing a Quality Strategy
- Process Benchmarking
- Yield Mgt
- Defect Removal & Prevention Strategies

Review of PSP Levels *(Humphrey, 1995, p. 11)*



Overview *(cf. Humphrey, 1995, p. 271, 272)*

- *Quality parts are the basis for quality systems:*
  - *"If you want a high quality software system, you must ensure each of its parts is of high quality."*
- *A quality process is required for a quality product:*
  - *"To improve your product, you must improve your process quality."*
- *Quality and productivity benefit from a quality focus:*
  - *"While the quality benefits of this… are important, the productivity benefits are even more significant."*
- *This chapter:*
  - *Shows how process and product quality are related.*
  - *Shows how to use PSP data to measure and track process quality.*

What is SW Quality?

(cf. Humphrey, 1995, p. 272-273)

- *Quality is defined in terms of the user:*
  - *"Conformance to requirements"*
- *The key questions:*
  - *Who are the users?*
  - *What is important to them?*
  - *How do their priorities relate to the way you build, package, and support your products?*
- *Two aspects of software quality:*
  - *Product*
  - *Process*

Product Quality: The Desire

(cf. Humphrey, 1995, p. 272)

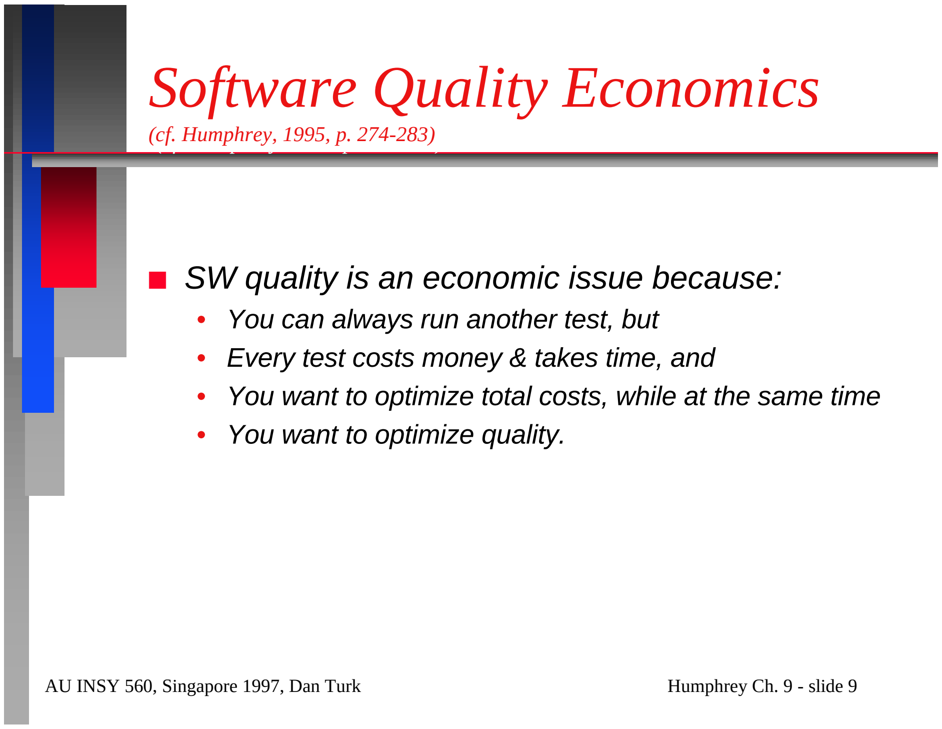
1. *SW must do what user needs, when they need it. "If it does not, nothing else matters."*
2. *SW must work - must not have so many defects that the user cannot use it. "If a minimum defect level has not been achieved, nothing else matters."*
3. *Beyond this threshold, everything else depends on the user, application, and environment.*
   - *Priorities will vary among users, there is no universal definition of "quality".*

Product Quality: The Reality
(cf. Humphrey, 1995, p. 273)

- The reality:
  - SW developers, while they do not debate the previous points, do not act as though they are what is important.
  - Developers focus on installability, usability, operational efficiency, testing…
  - This testing is the single-most costly element of SW development in most org's.
  - Because the components are not of high quality, system testing focuses on removing defects. When these defects are removed the system then reaches a "bare minimum quality threshold."
- What could be:
  - By creating quality components, and reducing their defect content, developers have time to address the more important issues…
- The conclusion:
  - Defect mgt is the foundation upon which to build SW quality.

Process Quality (cf. Humphrey, 1995, p. 273-274)

- Def:
  - "A quality PSP is [a process] that meets your need to efficiently produce quality products."
- Characteristics:
  - The process consistently produces quality software
  - The process is usable, efficient, and can be readily learned and adapted / improved.
- You are in control. Make the process what you need it to be.

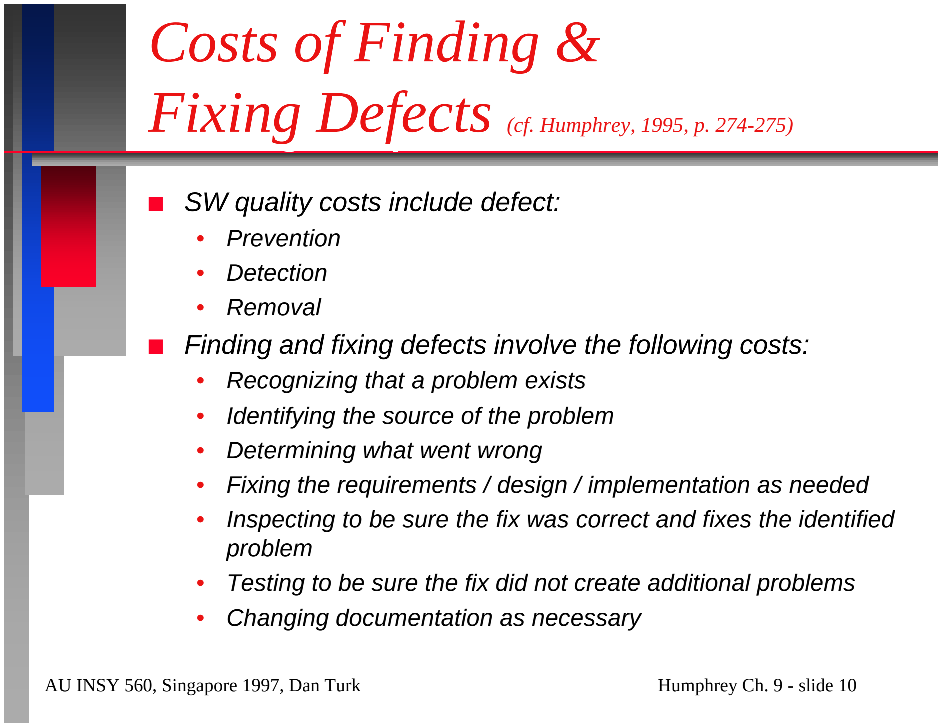


Software Quality Economics

(cf. Humphrey, 1995, p. 274-283)

- *SW quality is an economic issue because:*
  - *You can always run another test, but*
  - *Every test costs money & takes time, and*
  - *You want to optimize total costs, while at the same time*
  - *You want to optimize quality.*

AU INSY 560, Singapore 1997, Dan Turk　　Humphrey Ch. 9 - slide 9



Costs of Finding & Fixing Defects *(cf. Humphrey, 1995, p. 274-275)*

- *SW quality costs include defect:*
  - *Prevention*
  - *Detection*
  - *Removal*
- *Finding and fixing defects involve the following costs:*
  - *Recognizing that a problem exists*
  - *Identifying the source of the problem*
  - *Determining what went wrong*
  - *Fixing the requirements / design / implementation as needed*
  - *Inspecting to be sure the fix was correct and fixes the identified problem*
  - *Testing to be sure the fix did not create additional problems*
  - *Changing documentation as necessary*

AU INSY 560, Singapore 1997, Dan Turk　　Humphrey Ch. 9 - slide 10

Relative Fix Times by Phase

(cf. Humphrey, 1995, p. 275)

	IBM	TRW	IBM	JPL	F&W
Requirements		1			
Design	1.5	3-6			
Design Reviews			1		
Before Coding	1				
Coding	1.5	10			
Code Inspection			20		
Before Test	10				
Reviews & Inspections				90-120	2-5
Test	60	15-70	82	10,000	10
Field Use	100	40-100			

Questions About Relative Fix Times *(cf. Humphrey, 1995, p. 276-277)*

- *Question:*
  - *How do we know that the inspections are not finding the "easy" defects, but that the "difficult" ones are being left for test?*
- *Answer:*
  - *There is at least some evidence that inspections are at least as good as test at finding difficult defects:*
  - *PSP data shows that the relative fix time is the same between reviews and test - irregardless of defect type (cf. Fig 9.1 & 9.2, p. 277).*
  - *PSP data shows that reviews are 2+ times as efficient as testing at finding/fixing defects.*
  - *Organizations that do inspections report significant improvement in productivity and schedule performance.*
  - *The phase when the defect is injected does not seem to change this pattern.*

An Example (cf. Humphrey, 1995, p. 278)

- The situation:
  - 50,000 LOC estimated, 5-person team
  - 10 months spent defining req's, prototyping
  - 3 months spent in high-level design
  - Ready to start detailed design & implementation
  - Integration & system test to start in 5 months
  - Testing expected to take 3 months
  - Spec's inspection would require an additional 3+ months

An Example (cont.)

- Q:
  - Should the specifications be inspected?
  - If so, it looks like delivery will be delayed
- A:
  - Where did the 3-month test estimate come from?
  - Large projects are approximately 1/2 done when integration test begins (Brooks)
  - This project (18 months to start test) which was originally scheduled for 21 months, should have been scheduled for 36 months. If we continue without inspection, we'll appear to be on schedule until the 19th or 20th month, and then will experience 12-16 months of slippage...

Economics of Defect Removal

(cf. Humphrey, 1995, p. 278, 279)

- *Yield = % of defects found in review or inspection (out of the total # defects found over the life of the system)*
- *PSP students average about 70%*
- *For a 50,000 LOC project with 50 defects / KLOC not found by the compiler, 2500 defects are left to find in reviews or test. If these are found in test (at 5-10 hours per defect) this would take 20,000+ hours of time, or approximately 18 months for 5 people.*
- *If inspections were used and a 70% yield were obtained, 1750 defects would be found in review (at 1/2 hour per defect) and 750 would be left for test. This would take a total of about 6000 hours, or 6-8 months for our team of 5. This is a savings of <u>one whole year</u>.*

Why Don't More Organizations Perform Reviews? (cf. Humphrey, 1995, p. 280)

- *Lack necessary data to make good plans. Thus schedules are based on guesses and are unrealistic.*
- *Yield is not managed. Data is not available for how effective each phase is, or what the relative costs of defect removal in each phase are.*
- *Thus it is not apparent the great cost savings that could be achieved.*

Cost of Quality *(cf. Humphrey, 1995, p. 280-282)*

- *Cost of Quality (COQ) = Cost of POOR Quality*
- *COQ = A way to "quantify the size of the quality problem in language that will have impact on upper management." (Juran)*
- *Three components of COQ:*
  - *Failure costs (compile & test)*
  - *Appraisal costs (design & code reviews + inspections)*
  - *Prevention costs (Prototyping, causal analysis, process improvement)*

PSP COQ Measures

(cf. Humphrey, 1995, p. 281-283)

- *Failure COQ*
  - *(compile + test time) / total time * 100*
- *Appraisal COQ*
  - *(design rev + code rev time) / total time * 100*
- *Total COQ*
  - *Appraisal COQ + Failure COQ*
- *Appraisal % of Total COQ*
  - *Appraisal COQ / Total COQ * 100*
- *A/FR (Appraisal to Failure) Cost Ratio*
  - *Appraisal COQ / Failure COQ*

Developing a Quality Strategy

(cf. Humphrey, 1995, p. 283-286)

- Decide how to measure your process.
  - Distinguish between measures of product and process quality.
- Determine what QA methods work for you.
  - Product focus - won't work well because most errors are direct or indirect consequences of the process you use.
  - Process - try to improve the way you find and fix defects. This helps but does not address the sources of the problems.
  - Prevention (continuous improvement) - focus on the causes of the errors that produced the defects.
  - Combined - most optimal.
- Periodically reevaluate and set new goals.
  - The process needs to be continual and dynamic, and
  - Must be maintained over the long term

Process Benchmarking

(cf. Humphrey, 1995, p. 286-292)

- Def:
  - A benchmark is a measure against which to compare yourself, your process, etc.
- Process benchmarks should:
  - Measure the ability of the process to produce high-quality products
  - Provide a clear ordering of process performance from best to worst
  - Indicate the ability of the process to withstand disruptions
  - Provide required data that is objectively measurable
  - Provide data in a timely manner
  - Use standardized measures
- No SW benchmark meets all these criteria

Process Benchmarking (cont.)

- Useful benchmarks
  - A/FR - yield (cf. Fig 9.10, p. 189)
  - A/FR - defects (cf. Fig 9.11, p. 289)
- While these appear to be useful benchmarks (moderate to good correlation), they are not easily standardized, and thus it is hard to compare between people, projects, and organizations.
  - Defects defined / counted differently, and thus yield changes.
  - Phases defined differently, so A/FR ratio not standardized.
- However, they still can help you assess changes in your process.
- Be careful when comparing productivity vs. yield. There is not a clear relationship…
- Plot your project against the benchmarks.

Yield Management (cf. Humphrey, 1995, p. 292-296)

- Two competing processes:
  - Defect injection
  - Defect removal
- Relatively small changes in your process yield can result in large changes in the number of defects in the final product.
- Example, p. 293:
  - 1/3 drop in yield (75 to 50%) causes a doubling of the defect content.
- Example, p. 293, 294:
  - Semiconductor defect injection (small change in one step greatly changes overall yield) vs. software defect removal (small change in one step has insignificant affect on overall yield)
  - 1% probability of injecting a defect in each of 10 phases, compared with a 50% chance of injecting a defect in just one of these phases: $(1-0.01)^{10}=0.904$, $0.5(1-0.01)^{9}=0.457$ ----> Overall yield reduced by 50%
  - 80 and 40% phase yields, 100000 initial defects, 5 phases: $100000(1-0.8)^{5}=32$, $(100000-32)/100000=0.99968$ $100000(1-0.4)(1-0.8)^{4}=96$, $(100000-96)/100000=0.999004$ ----> Overall yield reduced by only about 1%

Five Categories of Defect Causes *(Humphrey, 1995, p. 295)*

- *Education*
  - *You did not understand how to do something*
- *Communication*
  - *You were not properly informed about something*
- *Oversight*
  - *You omitted doing something*
- *Transcription*
  - *You knew what to do but made a mistake in doing it*
- *Process*
  - *Your process somehow misdirected your actions*

- *PIP's could specifically address these areas...*

Defect Removal Strategies

(cf. Humphrey, 1995, p. 296-298)

- *Possible strategies to follow:*
  - *Compound multiple defect removal phases so the combined effect produces acceptable quality (cf. previous examples)*
  - *Evaluate cost & removal efficiency of these phases and use the most economical combination*
  - *Track, analyze, and manage these phases so that yield does not drop, and react to and correct the situation if it does*
  - *Begin defect prevention activities*
- *ID highest yield methods and combine into most cost-effective approach*
  - *A/FR ratio will help distinguish where the most effective activities are, when you may not be able to distinguish between 80/40 or 40/80 inspection / test using yield numbers.*
- *Put maximum effort early in process*
- *Specialize defect removal focus by phase: (see next slide)*

Defect Removal Strategies Focused by Phase *(cf. Humphrey, 1995, p. 298-300)*

- *High-level Design Review*
  - *Completeness of requirements*
  - *System constraints*
  - *System state machine*
  - *Class and object structure*
  - *...*
- *Detailed Design Review*
  - *Completeness of high-level design*
  - *Object state machines*
  - *Logical correctness of object methods & procedures*
  - *...*
- *Code Review*
  - *Complete / proper implementation of detailed design*
  - *Variable & parameter declarations*
  - *Punctuation*
  - *...*
- *Unit Testing*
  - *Check all paths…*
  - *Verify normal, limit, and outside-limit parameter value operations*
  - *Loop & recursion termination…*
  - *...*

Defect Prevention Strategies

(cf. Humphrey, 1995, p. 301-304)

- *Possible strategies to follow: Focus on defects*
  - *Found in final program test or use*
  - *That occur most frequently*
  - *That are most difficult to find / fix*
  - *Those for which you can easily ID preventative actions*
  - *Those that most annoy you*
- *General PSP defect prevention process:*
  - *Using collected defect data, choose a specific type of defect for analysis*
  - *Make specific changes in your process to address the causes of these defects (Educ, Comm, Oversight, Transcription, Process)*
  - *Walk-through the new process to assess its helpfulness*
  - *Test new process on a PSP-size program*
  - *Make process changes "permanent" (PIP)*