Confusing Process vs. Product: Why We Have Problems Producing Quality Software

Dr. David A. Cook
C. S. Draper Labs / USAF Software Technology Support Center
david.cook@hill.af.mil
(801)775-3055
DSN 775-3055

When I say “the program blew up”, I really mean it!!
THIS NEXT AWARD GOES TO KIM FOR HER EXCEPTIONAL WORK.

KIM WORKED EVENINGS AND WEEKENDS TO FIX THE PROBLEMS THAT WERE CAUSED BY HER OWN INCOMPETENCE.

AND IT LOOKS LIKE KIM HAS A FULL PLATE FOR THE COMING YEAR, TOO.

WHICH SIDE FACES THE WALL?
Topics

- Defining Quality
- Role of a Process
- Classic Mistakes
- How To Make Things Better
- Best Practices
- Worst Practices
What Is Quality?

- According to ISO 9001, it means meeting requirements
  - Let’s face it, most requirements are not that good
  - Many requirements are not even correct
  - There are have lots of unstated requirements
  - Stated requirements are frequently incomplete

- A simple definition is “Keeping the customer satisfied”
Problem - Who's the Real Customer?

CUSTOMER

Contract Administrator
Manager
Maintainer
User

Brackett, 1989 SEI Curriculum Module
CE-SPM-01-02-06
My favorite (personal) definition is “Producing a software product that the **customers** find **acceptable**, given their various needs (not their requirements!) and constraints.”
“Good” software is usually an organizational effort. Without a process, you are relying on “heroic” software development.

Software developers require some structure (Have you even looked into a developer’s cubicle? Checked out the tops of their desks? Don’t touch the piles though.)

But we **DO** have a process. We save every sheet of paper we can find!!
**Communication**

- Large-scale software requires many people - and without a process, there is little (if any) effective communication. This results in
  - multiple people developing the same functionality
  - lack of good coupling with other modules
  - lack of understanding of overall requirements (and tendency to assume “that’s not my problem - it’s being handled by somebody else”)
  - Lots of content-free or high “noise to signal” communication

Communicating with a Software Engineers is only slightly less difficult *than communicating with the DEAD.*
The Temple of Software Engineering

If we all work together, look at what we can build!

“Products” of a good process

Faster  Cheaper  Better

GOOD PROCESS

Product, Process, Quality
But just a few stubborn people can ruin the entire process!
ISSUE #1 - We don’t use common sense

- Lack of understanding of the role of the CMM / ISO 9000 in a process
- Failure to focus on the goal (quality software) rather than the process (achieving level X)
- Failure to follow the defined process
- Failure to integrate the entire development team into a cohesive unit
  - All developers must be “on the same team” to achieve quality
  - Management must be enlightened and help (not hinder) the process
  - The process should not degenerate into an “us vs. them” (management vs. development)
**Common Sense**

- Typically, we fail to reward slow progress that follows a process, but are quick to reward coding progress
  - Coding should not normally start until most of the requirements and most of the design is completed
  - Prototypes should usually be discarded after proof of concept
  - Teams that produce code early typically haven’t followed the process
    » They are still guided by the “super-programmer”.
    » They produce code that usually won’t integrate well, and will be un-modifiable and un-maintainable.

116 lines in 4 minutes. GREAT CODING!!

Sure wish I knew what the requirements were!
Common Sense

- Management often exhibits subtle clues that they still believe in “WAISACY”

Why Aren’t I Seeing Any Code Yet??
We don’t use common sense!

What’s the tie-in? Quality is EVERYBODY’S Job!

Winner of the "Not My Job" Award - ADOT
Litchfield Park, AZ 85
Speaking of “Common Sense”

- Management, program managers, and end users have to understand that there are many factors that affect software quality
  - schedule (or time constraints)
  - cost (involves # of people, resources, etc.)
  - functionality of the software (complexity)
- Many factors are beyond management control
- Any change to one variable affects some (if not all) of the others
- Since Quality can’t be “retro-fitted” - you must plan for it
- Quality, once ignored, can never be inserted back into a process
- Want productivity? Focus on quality, and the increase in productivity will follow!

Why is it called COMMON sense?
The Development Triangle

You can control only two sides of a triangle; The third side MUST be a “free variable”.

Faster Schedule

Better Product (Functionality + Quality)

Cheaper Cost
ISSUE #2 - A process must be dynamic, and multiple versions must exist for different development efforts

Processes must adapt to the project phase, as well as to the developers.

- Developers implementing the real-time kernel need a different process from the data collection team
- Review processes differ when in design as opposed to coding and integration
- Nobody starts with a “good process”. They start with a poor process, and improve it!
Common mistake - having an organizational *Process Police* to enforce compliance with a single process

- A good process is like shoes. The size that fits you perfectly doesn’t fit others perfectly
- Face it - your shoe (and process) have to be fitted to your needs.

Every different development task probably needs a separate, customized process. If there is a larger, organizational process framework, it must accommodate these sub-processes
Tailoring a process

- Agreed - there are probably several processes that will work for you. One particular process might work for many different needs within a large development organization.

- Most likely, however, several specialized development teams cannot use the same process other groups use.

- If you only have a single organizational process, it’s probably like caffeine-free diet colas. Doesn’t really have much useful value, and gives everyone gas.
process is (usually) better than nothing

- You have to have someplace to start
- Nobody ever starts with a good process - they take a “best guess” and work to improve
- A process must be dynamic - if your process can’t change, then a poor process is NOT better than nothing

*YOU HAVEN’T HEARD WHAT THE PROBLEM IS YET; HOW CAN YOU RECOMMEND BUILDING A DATABASE TO SOLVE IT??*

*WE ALWAYS BUILD A DATABASE.*

*AND WE’LL NEED COFFEE MUGS FOR THE PROJECT TEAM.*

*THE PROBLEM IS THAT WE HAVE POOR PROCESSES.*

*THAT COULD BE THE SLOGAN ON OUR MUGS!*
ISSUE #3 - Poor (or uninformed) management overwhelms good processes

We all laugh at the “pointy-haired boss”, but don’t you have your own favorite story that is based on a Dilbert cartoon?

Isn’t your story WORSE that what happened in the cartoon?

By the way, management is usually clueless because they don’t have access to accurate information.
Why management fails

- Failure to keep up with current trends
  (sure - like you have LOTS of time for reading and study)
- Failure to discern the difference between what has been briefed to you and what is REALLY happening
- Failure to brief those “up the food chain” with the truth
- Failure to recognize good (quality) effort from poor (but glitzy) effort
- BUT MOSTLY - due to unprofessional behavior on the part of a developer
Management Training

Manager Training
You will often be asked to comment on things you don't understand.

These handouts contain nonsense phrases that can be used in any situation.

...So, let's dominate our industry... with quality implementation of methodologies.

I'll get right on it.
work

ISSUE #4 - If you don’t measure the product quality, you can’t have a quality product

Question #1 - What metrics do you use?

Question #2 - What metrics allow you to tailor your process to make it better?

Question #3 - The metrics that were listed in question #1, but not in question #2 — why are you collecting them?

Why do they want developer age per per coding error? And is higher or lower better?
Classic mistakes

- Some mistakes are just endemic to SW development
- To avoid almost all of these mistakes, you must have a strong risk management plan in place and a commitment to quality!
Classic Mistakes - People-Related

- Undermined Motivation
- Weak Personnel
- Uncontrolled Problem Employees
- Heroics
- Adding People to a Late Project
- Developer/Customer Friction
- Noisy, Crowded Offices
- Unrealistic Expectations
- Lack of/Weak Sponsorship
- Lack of/Weak Stakeholder Buy-in
- Lack of/Weak User Input
- Politics Placed Over Substance
- Wishful Thinking

How was I to know about Brooks Law??

McConnel, Rapid Development, 1996
Classic Mistakes - Process

- Overly Optimistic Schedules
- Poor/No Risk Management
- Sub-Contractor Failure
- Poor/No Planning
- Abandon Plan Under Pressure
- Fuzzy Front End
- We’ll Do that as We Go
- Inadequate Design
- Poor/No Q. A.
- Poor/No Management Controls
- Omitting Important Task from Plan/Estimate
- “We Will Catch Up Later”
- Code-like-mad Programming
- Uncontrolled Feature Creep
- “We Must Use The Latest and Greatest Technology” Syndrome

McConnel, Rapid Development, 1996
Classic Mistakes - Technology

- Silver Bullet
- Over Estimate Saving for Tool Introduction
- Musical Tools in the Middle
- Poor/No Configuration Control

McConnel, Rapid Development, 1996
OK - enough bad news!
How can I make things better??
process(es)!

- Train your people
- Encourage your people
- Encourage adherence, but allow change
- Provide negative encouragement to those who just won’t follow any process
- Management must
  - buy-in
  - show support
  - show enthusiasm
  - follow the process, too
Implement required training

- ..or maybe education

- Benefits of process improvement must be explained

- Process improvement must be viewed as a “team effort” - the benefits may not be apparent, but occur “down the line”

How am I going to tell them that they shouldn’t use COBOL for real-time systems?

Cost of software
Isn’t this the same guy who *made* us attend the “Process Improvement Seminar” last week?

I’ll go gather the requirements and come up with a design. The rest of you START CODING!!
Second of all (?) - get a life(cycle)

If you don’t have a life cycle, you don’t know
- Where you are
- Where you are going

A Need → Requirements → High-Level Design → Low-level Design → Code

Code → Unit Test → Integration Test → System Test → Finished System

Success/Failures

Requirements, Process, Quality
Speaking of lifecycles... (pause for humor)

Don’t use the Waterfall model OR its’ variations

- Waterfall Model - Basic requirements, then design, then code, and then test.

- Pond Model - Code and ideas stagnate and grow other life forms.

- Water Fountain Model - Same as pond model, though looks prettier. Underneath the pretty “wrapper”, it still stagnates.

- Firehose Model - Well focused effort on putting out fires. Unfortunately, no time or effort is spent on PREVENTING fires!

But the Waterfall model worked for me back in the 60s.
And yet more...

- Toilet Bowl Model - Combination of Spiral and Waterfall models. Usually have problem with things that don’t flush.

- Thunderstorm Model - Loud, noisy and dangerous. Usually results in flooding with developers moving to higher ground.

- Tornado Model - Faster implementation of Spiral Model, usually wipes out development staff.

- Hurricane Model - Close attention paid to tracking its course, though no one can predict when it will arrive.

Thanks to Hans-Ludwig Hausen, GMD (German National Research Centre for Information Technology)
What type of lifecycle to use

- Pick one that’s iterative (you might as well plan to throw the first system away - you’re probably going to anyway!)

- Pick one that allows prototyping and/or exploratory programming

- Consider spiral or another newer iterative model - pick one that allows tailoring

- Follow the lifecycle!

I didn’t use no stinkin’ lifecycles in school, so I ain’t gonna use them now!
Lifecycle Limitations

- Compare developing software to aging fine wine
- You can plan all you want, but it’s still going to take some minimal amount of time
- Too quick - and the quality suffers
- No amount of planning can shorten the time below some constant - but improper planning and preparation lengthen the time considerably!
evaluate the process and the lifecycle

- Collect honest metrics
- Only collect useful metrics
- Use them to modify the process
- Make sure that the metrics focus on quality and error prevention.
- Most likely, you will have to combine metric collection with testing and reviews

Why do I need coding errors per fortnight?
testing

- Myths
  - Testing shows that my program works!
  - Finding a lot of errors is good!
  - Finding few errors is good!
  - Testing can be performed by the team that wrote the software!
  - I’m CMM Level X (or ISO 9001) - therefore my products must be high-quality, and don’t need testing!!

**Testing typically removes only 50% of the errors present. You must have quality code going into testing to have quality code coming out!**
check)

- Don’t reply upon only upon a testing team
- Understand the importance of
  - Peer reviews
  - Self-inspections

<table>
<thead>
<tr>
<th>Testing Type</th>
<th>Defects Found Per Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Use</td>
<td>0.210</td>
</tr>
<tr>
<td>Black Box</td>
<td>0.282</td>
</tr>
<tr>
<td>White Box</td>
<td>0.322</td>
</tr>
<tr>
<td>Reading / Inspections</td>
<td>1.057</td>
</tr>
</tbody>
</table>

AFIT, 1994
Implement best practices

- Understand that what works for one organization might not work for another

- Best practices can be gathered from various sources, most particularly the Software Program Manager’s Network (www.spmn.com)

- Back in 1995, the DOD instituted a Best Practices Initiative.

Use the Best Practices that make sense for your organization.
Airlie Software Council List: Best Practices

- Use formal risk management
- Have the user manual as the specification
- Implement inspections, reviews, and walkthroughs
- Use metrics-based scheduling and tracking
- Have binary quality gates at an inch-pebble vs. milestone level

But I have a compiler. Why should I review my code??

You’ll LOVE desk-checking!
Airlie Software Council List

- Have program-wide visibility of the project plan and progress of the plan
- Implement defect tracking against quality targets
- Separate specification of hardware and software functionality
- People-aware management accountability

If we didn’t have a plan, how would we know how we’re doing?
practices..

- The Airlie council also came up with a list of worst practices.

- As with the best practices, not all will apply to you, and some might be irrelevant.

- Key advice - the people “in the trenches” are your best source for worst practices. Don’t ignore them!
Worst Practices

- Don’t use schedule compression to justify usage of new technology on any time critical project

- Don’t specify implementation technology in the RFP

- Don’t put items out of project control on the critical path
Worst Practices

- Don’t expect to recover from any substantial schedule slips (>10%) unless you plan to reduce functionality
- Don’t expect to achieve large, positive improvements (>10%) over past performance
- Don’t bury all project complexity in the software (as opposed to hardware)
- Don’t advocate use of unproved “silver bullet” approaches

Edward Yourdon, “Rise & Resurrection of the American Programmer”
Worst Practices

- Don’t conduct critical system engineering tasks (like software/hardware partitioning) without sufficient SOFTWARE ENGINEERING expertise.

- Don’t believe that formal reviews provide an accurate picture of the project. Expect usefulness of formal reviews to be inversely proportional to the number of people attending beyond five.

I said **ONLY** invited reviewers!
Personal Best / Worst Practices

- Don’t assume legacy code works as specified or documented
- Don’t assume that interfaces are correctly documented or specified, either
- If legacy code or interfaces are out of your control, demand early access to test the code or interfaces
- Provide proper motivation for the right behavior

But they’re supposed to fit together!
Encourage the “right” behavior

➢ Try not to reward negative behavior, or even worse, provide positive motivation for negative behavior
Final Thought

- The purpose of a process is to guide you, and to prevent you from making the same errors the next time.
- Track inadequacies - and feed them back into process improvement.

Those who don’t learn from their mistakes (and improve their process) are condemned to make the same mistakes over and over again!