Application of Quality Management Principles to Software Development

Main Points

Predictability Has Value
What do we mean by "predictability" in Software Development?
How do we measure predictability in Software Development?
What is the value of predictability in Software Development?
How do we achieve predictability in Software Development?
A Simple, but Effective Controllable Software Development Process.

Process Control

Predictability has Value

An Example: The Cost of Computer Virus Protection vs. The Cost of an Attack

What Do We Mean by Predictable?

Our Instinct about Random Variation vs Assignable Cause
How do we know when to start thinking about Assignable Cause?
Average vs. Standard Deviation
Average
Any result could be Random Variation
Problems with basing policy on averages
Standard Deviation
Results outside 3 sd's very unlikely to be Random Variation

What is the Value of Predictability in Software Development?

Comparison of Controlled Processes
Controlled, Well Controlled, No Control, Bimodal (See the graphs on page 8)
Advantages of a Controlled Process
1) Know what to expect (budgeting, planning)
2) Know when things are going right and wrong.
In a non-controlled process, every result is as likely as any other, therefore, especially good or bad results are **not** an indication of having done something right or wrong.

In a non-controlled process, policy decisions are "knee-jerk" almost by definition.

In a non-controlled process, people receive praise/blame for results that are beyond their control.

3) Can tweak the process and have confidence in the results

How do we achieve predictability in Software Development?

How do you get a bell curve?

If you do the same thing the same way, you should get the same (or similar) results.

Defining the Process

The process can be whatever you want—but the process must be manageable, improvable, and auditable—in other words, the process itself must produce meaningful ways of determining:

1) The progress of the process in "real" time (Quality Gates)
2) How well the process performed (on completion) (Metrics)

Note: Metrics (perhaps the best metrics) are not always "planned", but identified—that is, they are a naturally occurring result of the process.

3) That the process is actually being followed. (Auditable Records)

Policy Talk vs. Policy Deployment

Process Improvement

Philosophy / Rationale / Approach (in no particular order)

A bug is both an indication of a failure in the software, and an indication of a failure in the software development process.

The further back in the process that we can catch bugs, the cheaper it is to fix them.

The further back in the process that we can catch bugs, the less likely it is that the bug will negatively impact scheduling and budgeting.

The further back in the process that we can catch bugs, the less tempted we will be to fix them with a kludge or a hack.

Quality Gates

A formal review—that is, the review is itself a well-defined process that can be improved.

Who reviews, how the review proceeds (often, a checklist of things to think about when reviewing).

An excellent Project Management tool—gets you beyond "basically"—everyone agrees on whether or not the milestone has been achieved.

Makes a snapshot of program progress (i.e., multiple related projects) possible.

Consider the following: Quality Gates generally mean that the people who do the actual work don't get the blame when things go wrong—it is the job of the reviewer(s) to make sure that there are no "escapes" (defects in the current process step). Quality Gates push the responsibility for failure "up" (where it belongs.)

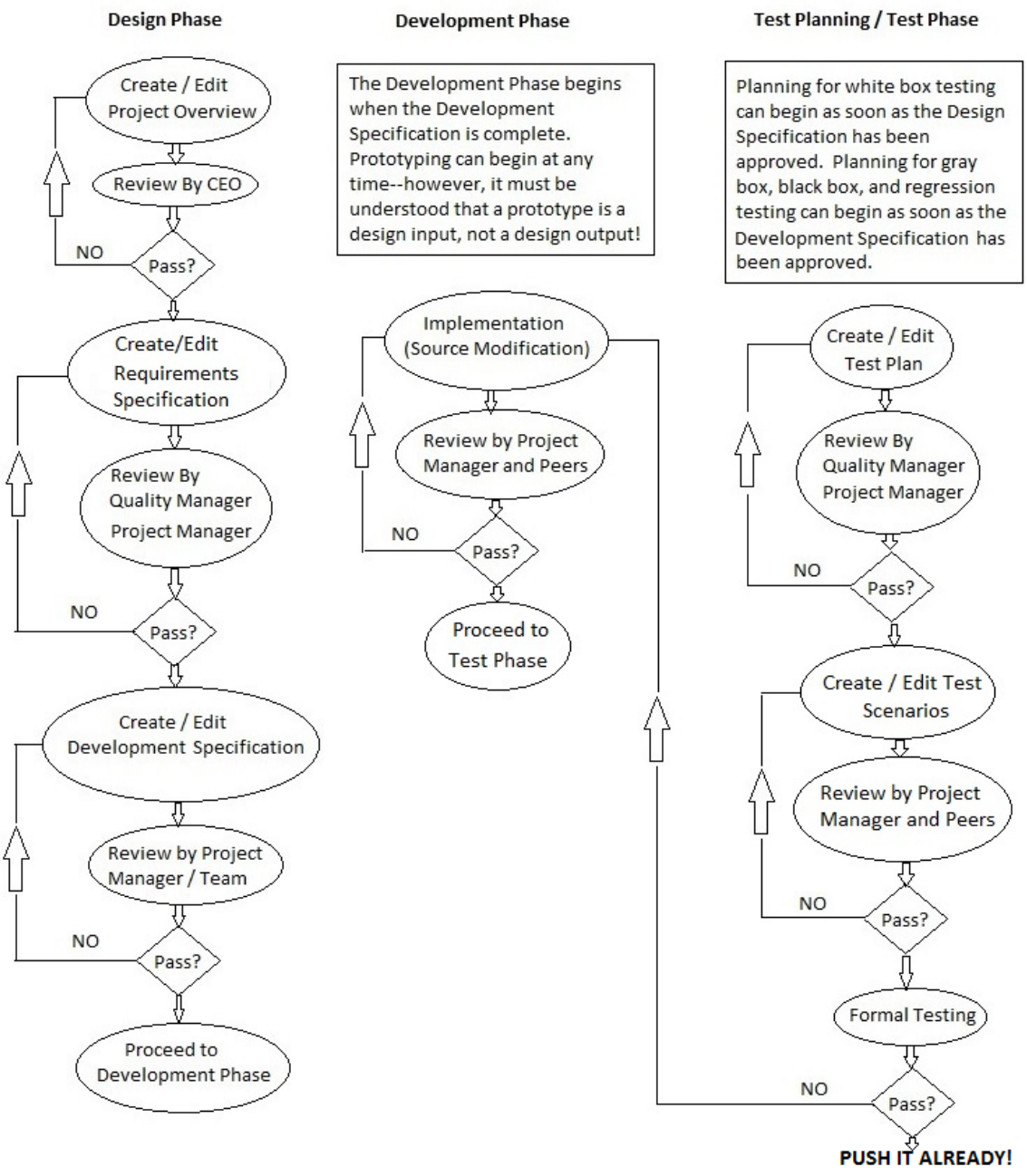
Allows you to identify the source (location in the process) of bugs (AKA "escapes").

Traceability

The ability to trace a feature from design input through testing (in both directions): A _real_ pain in the rump, (especially specifications to test) but absolutely critical for a well-controlled process.

1) Insures that everything gets tested
2) In combination with Quality Gates, makes it possible to determine where in a process failures are occurring.

A Simple but Effective Controlled Software Development Process



What is the Process Definition?

See above (a pretty standard "waterfall" model). What differentiates this model from most software development processes in use are the Quality Gates—that is, a formal review of each step in the process. The review is itself must be a well-defined process—that is, a process that can be repeated, monitored, audited and improved.

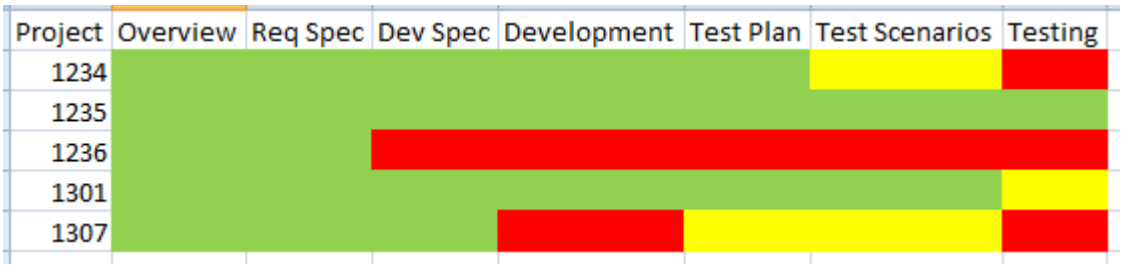
A couple of definitions:

Requirements Specification: (What) Business Requirements, Performance Requirements, Security Requirements, etc.

Development Specification: (How) Critical algorithms, new or modified classes, database schema modifications, database tables updated, SOUP or OTS software to be used, etc.

How can projects be tracked in "real time"?

It is the reviews that make project tracking both possible and meaningful, because a signed-off review indicates that the step is actually complete, and not just "basically" complete.

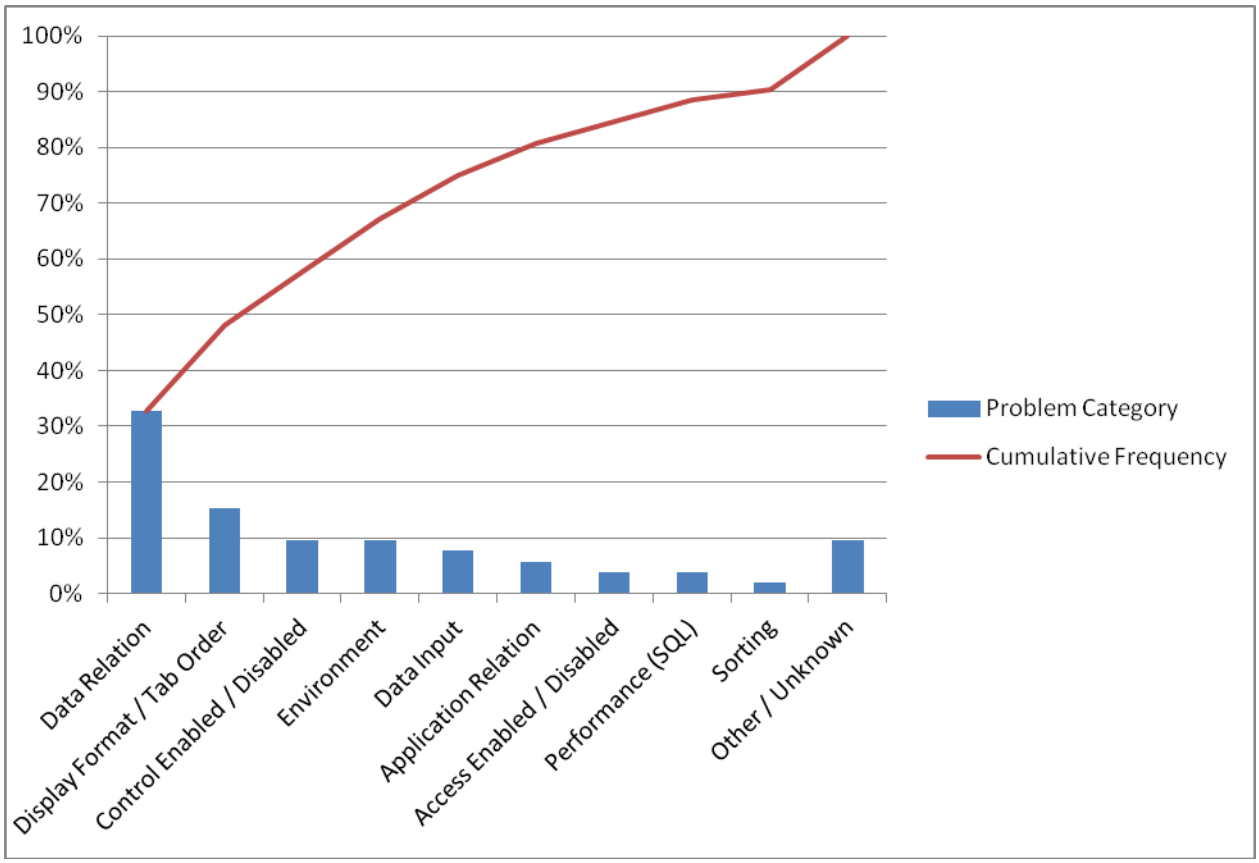


What is the Auditable Record of the Project?

Overview, Requirements/Development Specifications, Test Plan/Scenarios/Results and, very importantly, **documentation of reviews**.

Review documentation is important because an auditor needs some kind of evidence that the right people are thoroughly reviewing and approving the process at each step.

What is an Important Metric?



Quantity of bugs by type (collected over one year, in this case)
Note: only possible if bugs are analyzed and documented *before* they are fixed.

What is a Potential Process Improvement Initiative?

About 1/3 of this company's bug reports were related to "Data Relation" problems—that is, the database schema or the data has been modified by or for one functional element of the application in a way that causes another functional element of the application to fail.

How might we reduce Data Relation bugs that get into production?

> Think about traceability—if we follow the bug back through the development process, where does it first appear—or in other words, what is the Quality Gate that should have stopped the bug, but didn't?

What is a possible positive (quality) benefit of a pre-test solution to this problem? In other words, if you determined that both improved design procedures and improved testing procedures solved the problem equally well, and you could only afford to do one of those things, what are some rationales for choosing the design approach?

Some End Notes

Basically, we're talking about ISO 9001 as applied to software development, or, in other words, going beyond testing as the be-all and end-all of software quality management.

If you intend to implement an ISO 9001 compatible software development process, and you've never done it before:

1) If the boss is not fully onboard, the process is doomed to fail.

2) Get help.

3) It's hard. Expect blowback.

4) It works.

Questions?

Something to think about (or discuss, time permitting):

How could you apply the concept of a controlled process using quality gates to other aspects of software development such as:

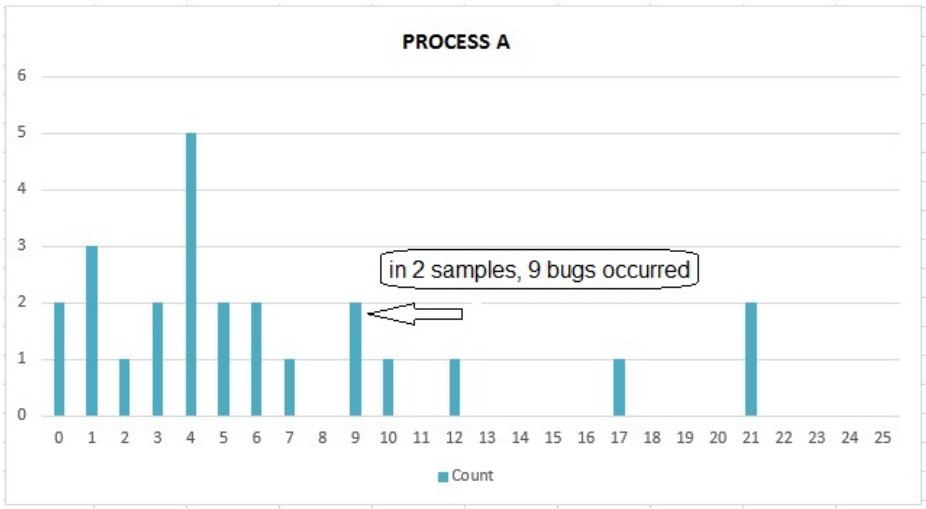
- Budgeting
- Scheduling
- Developer Hiring
- Developer Training
- Client Services (Problem Resolution)

Bugs per 100 hours development time: the bar shows number of occurrences in a sample size of 25.

Mean	6.36
Std Dev	5.92

Chance > 10 bugs : 27%

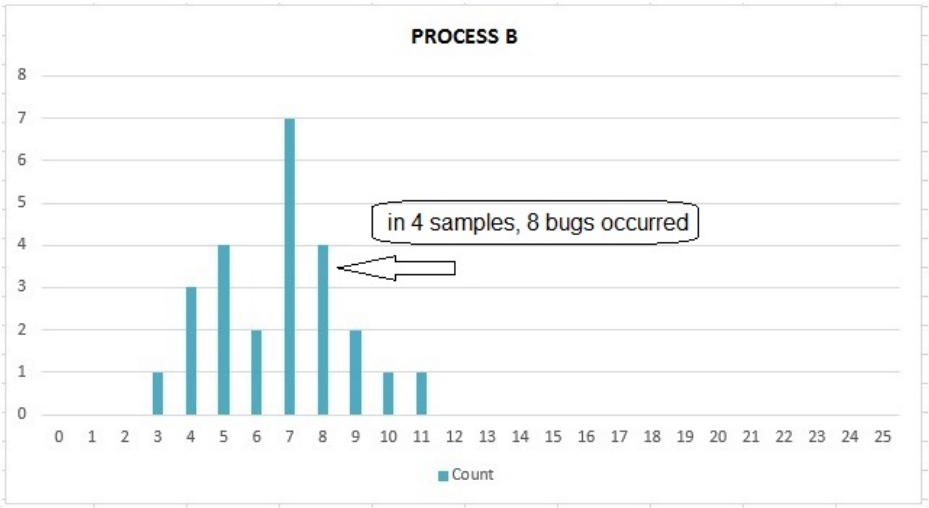
Chance > 15 bugs : 7%



Mean	6.68
Std Dev	1.99

Chance > 10 bugs : 5%

Chance > 15 bugs : 0% (.001%)



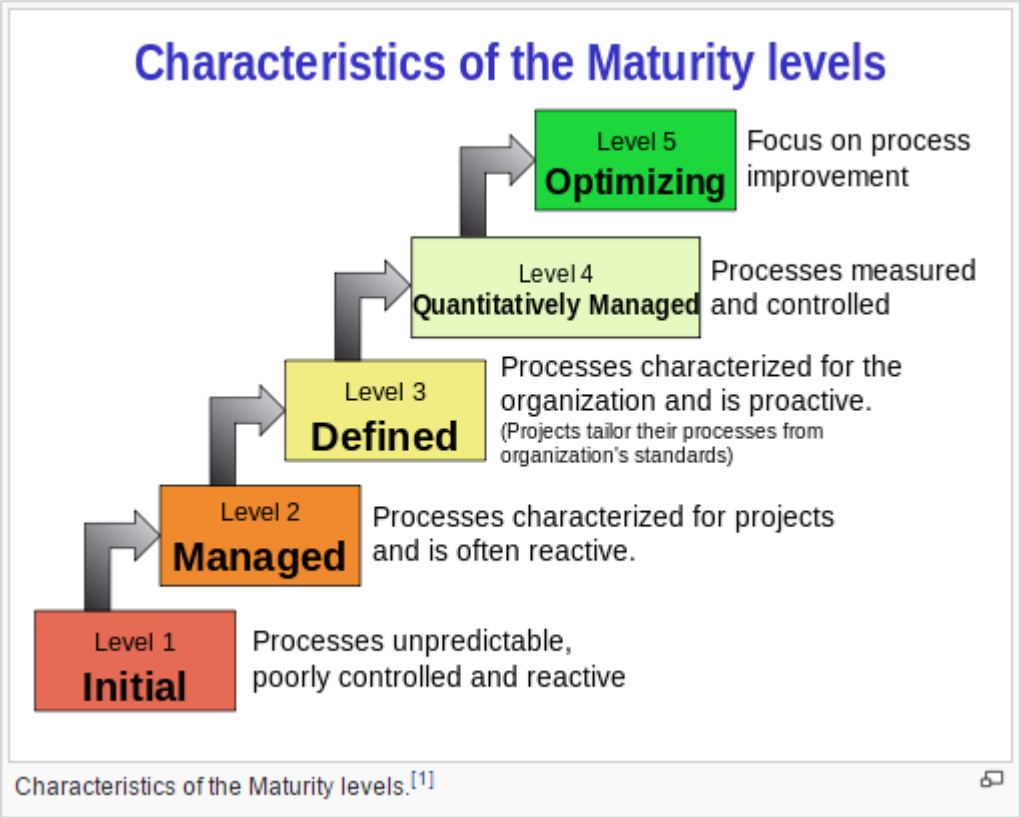
An example of a common misconception about what a Software Quality Engineer does:

Job description

Software Quality Engineer – Provider Technology
Portland, OR or Tacoma WA

You are part of a team responsible for assuring the delivery of high quality products and software services. In this role, you will be developing test strategy, test scenarios and effectively translating them into 'Automation ready' test cases. You will set test execution schedules, execute tests and report out result metrics to all appropriate project stakeholders. You will serve as primary interface to analyze requirements and verify against it delivery by identifying defects and seek resolution as appropriate. You also will be serving customers at the highest level with high quality product/services and ensure all escalations are resolved in timely manner.

A useful way of visualizing the goals of QM implementation (Capability Maturity Model Index).



Characteristics of the Maturity levels.[1]

References:

If you are interested in learning more, here are some standard reference books:

Quality by Donna C. S. Summers

The Certified Software Quality Engineer Handbook by Linda Westfall

A Practical Field Guide for ISO 9001:2008 by Erik Valdemar Myhrberg, Ph.D. (there may be a more recent version)

IEC 62304 Medical Device Software – Software life cycle processes

Programs:

If you are super-duper interested, University of Wisconsin Stout has an excellent online QM program (but it's expensive).