Tonight's Topic

Reliability and FMEA Keys to Software Quality

Adam Bahret; Owner & Lead Engineer of Apex Ridge Reliability Consulting Services

Abstract

Software reliability is an often little-understood but key part of software quality. Reliability uses very different tool sets and practices. Understanding how the tools work and how they affect the customer's experience is a valuable lesson for software development, quality, and application/maintenance specialists. Reliability expert Adam Bahret will provide an overview and discussion of the following reliability techniques and tools that software quality professionals need to know about:

- Software Failure Mode Effects Analysis (SFMEA)
- Process based software reliability prediction
- Overview of software reliability standards

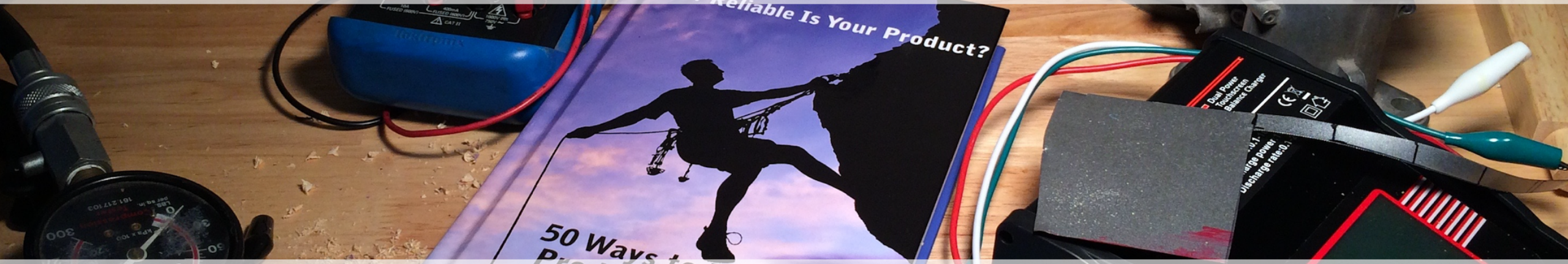
Bio:

Adam Bahret is the founder of Apex Ridge Reliability Consulting Services. He is a mechanical and electrical systems reliability expert with over 20 years of experience in product development across many industries. He has worked extensively with reliability program strategy, accelerated testing methods (HALT, HASS, QALT, and ALT), system reliability measurement and improvement, FMEA, ALT, DOE, RG, predictive analysis, education programs, and organizational culture and practices. He has specialized experience in the medical field, robotics, consumer electronics and appliances, ion implantation, and diesel systems. Adam has an MS in Mechanical Engineering from Northeastern University, is an ASQ nationally certified reliability engineer, and is a senior member of IEEE. More information on Adam and Apex Ridge Reliability can be found at www.apexridge.com.

YOUR LEADER
IN RELIABILITY SOLUTIONS

Adam Bahret

APEX RIDGE RELIABILITY provides tailored reliability solutions
that advance our customer's products in the market.



Google

rethink robotics

FLIR SYSTEMS

Medtronic

KEURIG GREEN MOUNTAIN

amazon robotics

Taco Comfort Solutions

PRATT & WHITNEY DEPENDABLE ENGINES

HYUNDAI

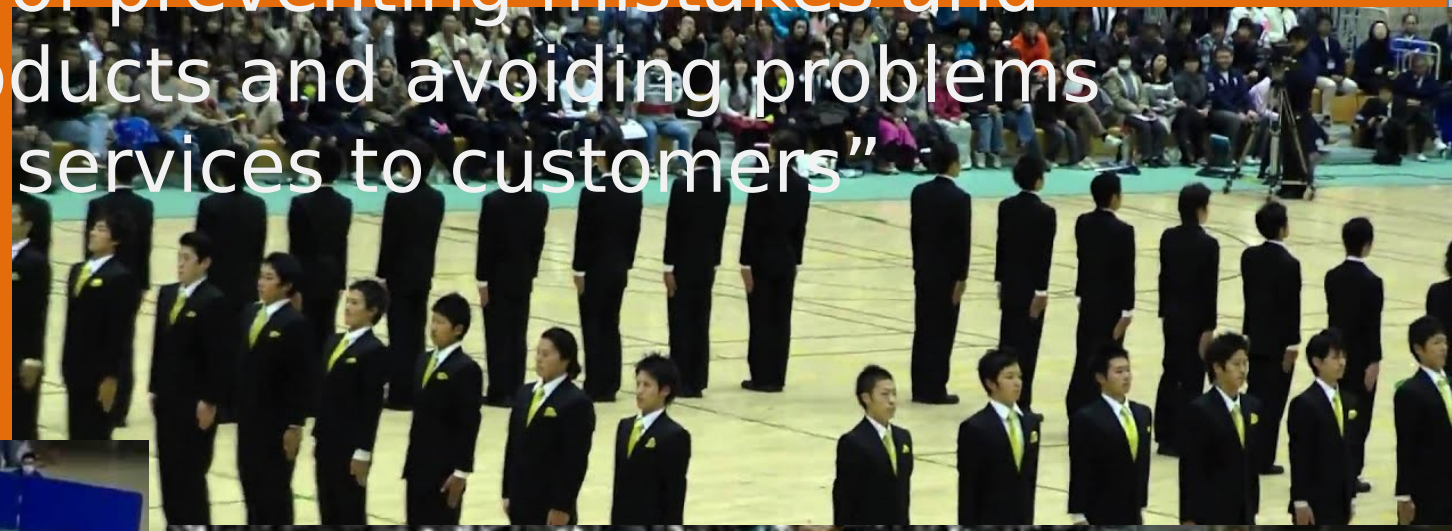
AMERICA'S NAVY

VARIAN medical systems

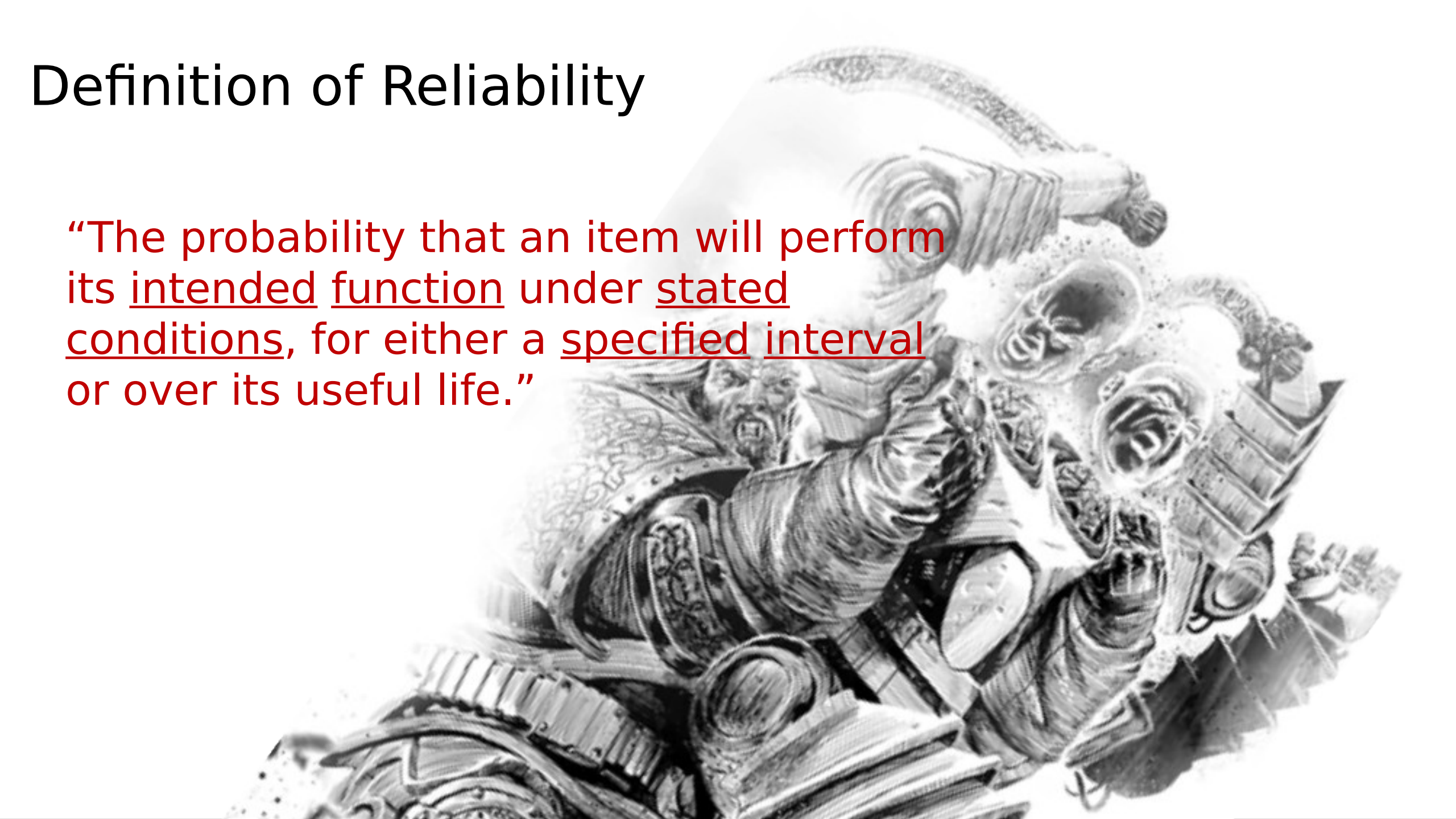
BostonDynamics

Definition of Quality

“Quality assurance is a way of preventing mistakes and defects in manufactured products and avoiding problems when delivering products or services to customers”



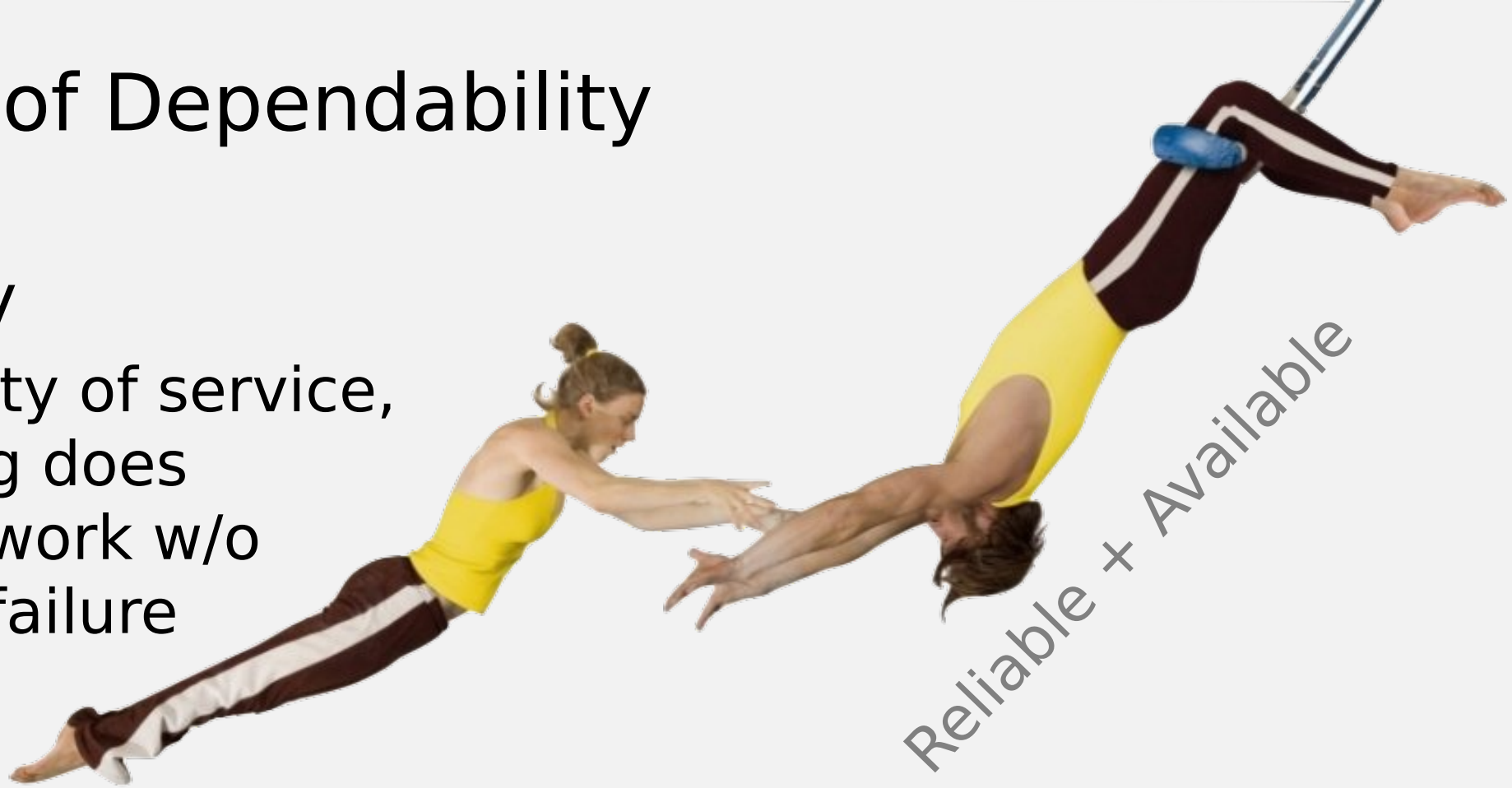
Definition of Reliability



"The probability that an item will perform its intended function under stated conditions, for either a specified interval or over its useful life."

Definition of Dependability

- Reliability
  - Continuity of service, how long does system work w/o system failure

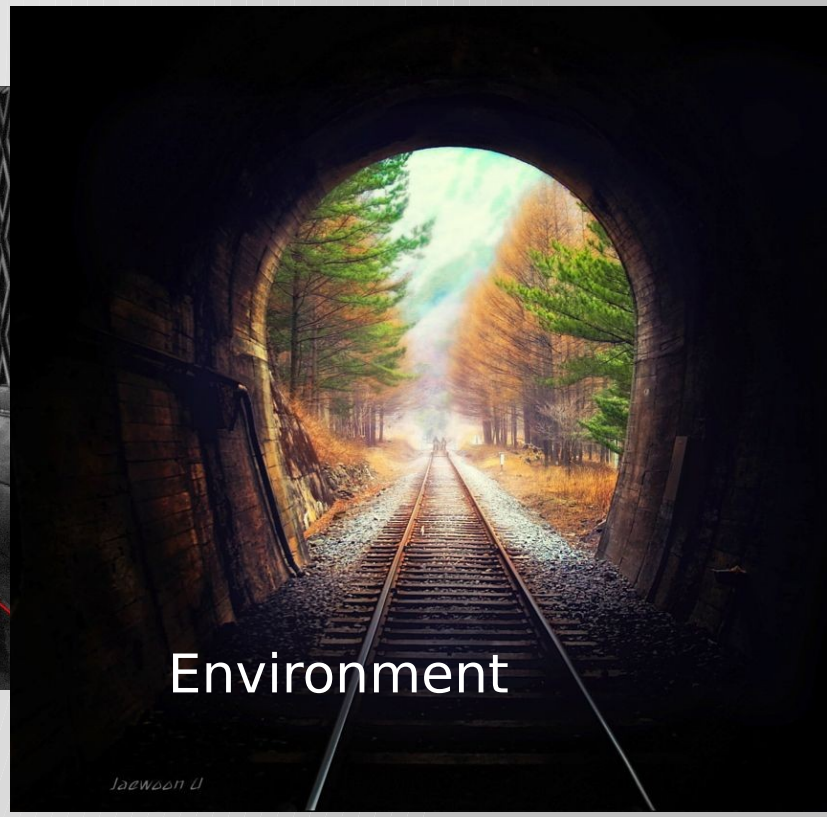


- Availability
  - Readiness of Service, how frequently it fails and how quickly it can be repaired/restored/recovered

Hardware Reliability has a simple mission

APEX RIDGE RELIABILITY

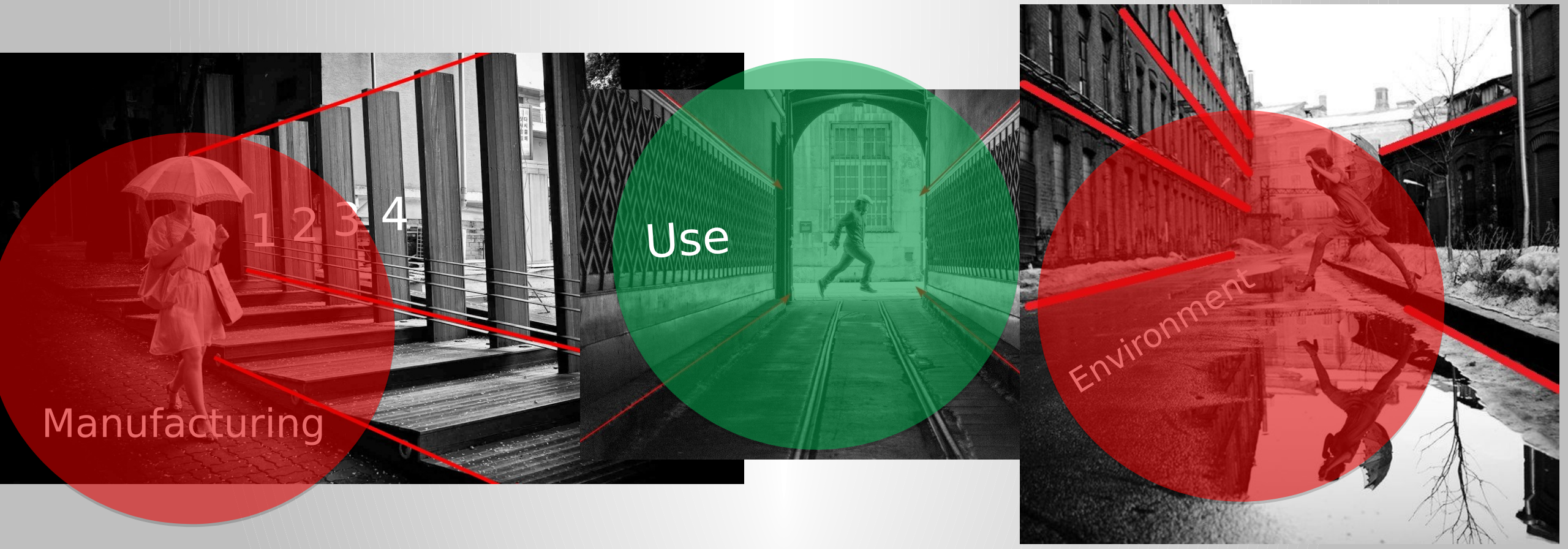
Mitigate the effects on performance from the variability of these three things



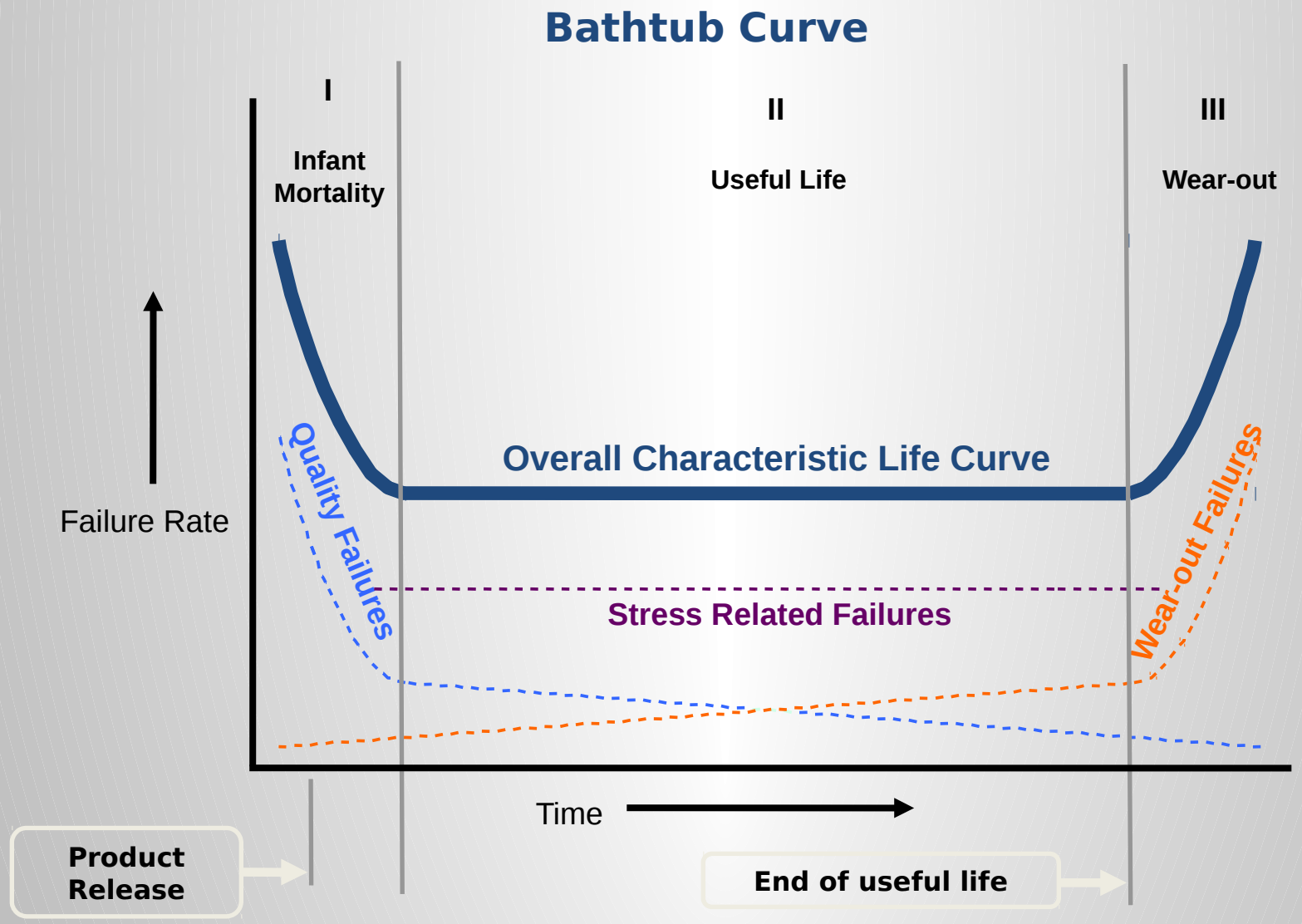
What is software reliability trying to control the variability of?

Variabilities that drive Software Reliability?

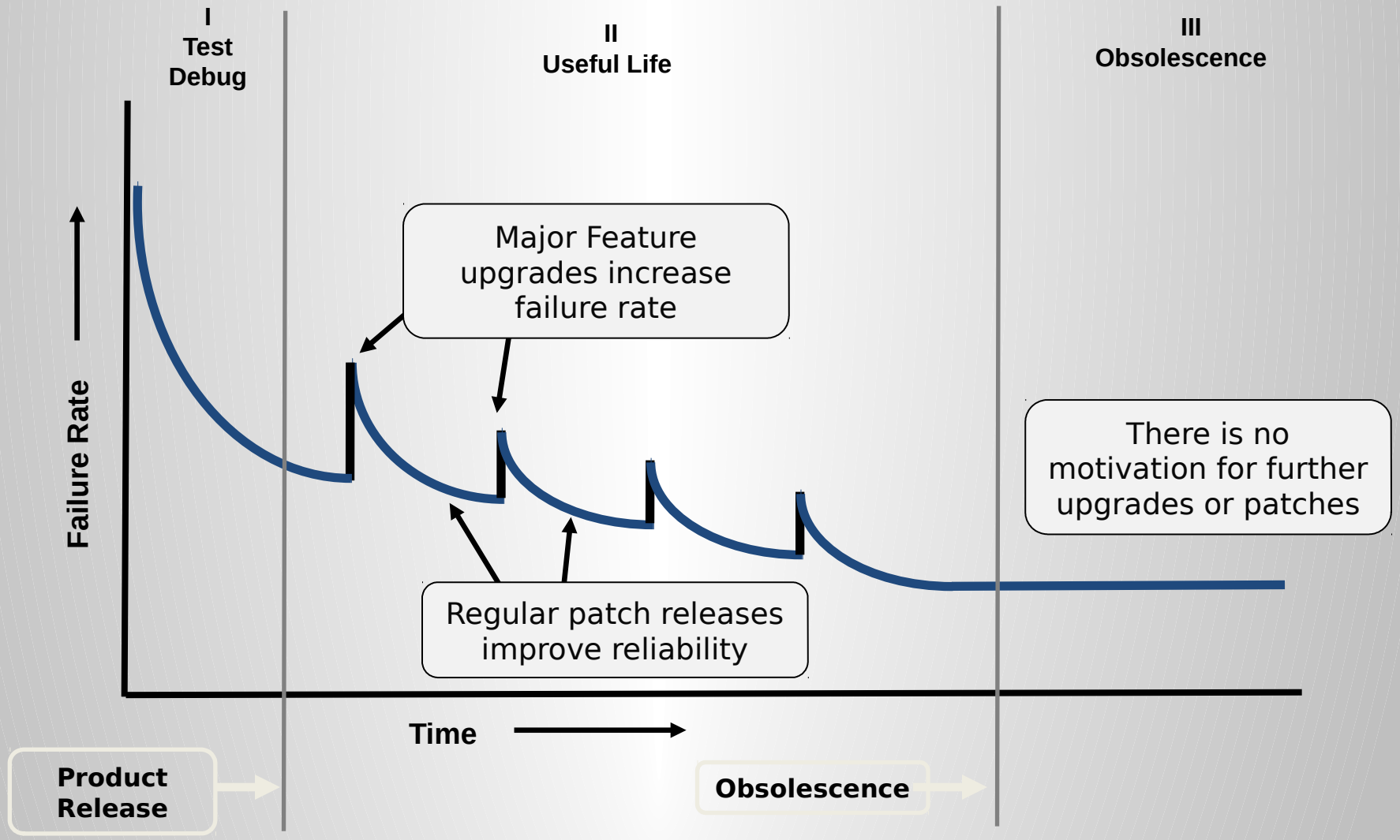
APEX RIDGE RELIABILITY



Mechanical & Electrical Reliability



Software Reliability



A few Strategies

- Instead of manufacturing can we predict reliability for software with how it is developed?
- Processes
- Debugging
- Sourced code

Continuous Integration

APEX RIDGE RELIABILITY

Determine your criteria for undo check-in.
Keeping metrics will reward those with clean check-ins

Code Check-in
- Style Checks
- Lint – Static Code Checks
- Compile – Report Warnings / Errors

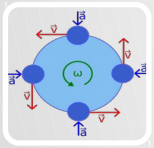
Unit Tests
- Module Level Test – Pass/Fail
- Coverage
- Profiling – Report

Determine your criteria for Module Passing
Metrics for Coverage are critical
Comparing Profiles (actual time per test) can find code that passes but execution time or memory footprint changed too much

If All Unit Test (Modules) pass...

1. Increment Build #
2. Annotate Source Control
3. Pull from SCM and Build
4. Includes Installer & Documentation
5. Backup Everything if Successful

Build
- Establish Version, Build
- Build Installer
- Backup Build Sources, Bins, Installer



Simulate
- Test Installer
- Startup & Initialization
- Run Battery of Full Integration Tests

Simulation may be best performed on a Virtual Machine
- Virtual Box is free
- Command the VM to restart from Last version & Clean Install

Sourced Code

- Reliability When you have to rely on other people's code
- Software Package Management Systems
- How do you know if a library has security vulnerabilities?
  - The problems: Historically you search for libraries to build upon. How is it vetted. They need to be vetted via open source.

- The Paradigm
  - Npm pkgSafe (javaScript), NuGet, Open pkg, Pip (python)
  - Every language platform has a community library management organization
    - Self Policing, maintaining integrity,
    - Manage Complex deep dependencies, secure dependency hierarchy all the way to the drivers.
    - Prevents you from shipping
    - It checks dependancies when you set your parameter

It will tell you how many vulnerabilities you have and then go get the patches.

Using the Binaries

- Black Duck (When you don’t have access to source code)

- Building on open source: It will look for vulnerabilities in the binaries (compiled output)..

So even though the compiled libraries can’t be easily viewed or might even be encrypted the binaries will have identity data that can be evaluated

- Static Analsyis : Not running the code. Just analyzing the source code.
  - Static code analysis tool for Java, C, C++, and C#. It analyzes every line of code and potential execution path and produces a list of potential code defects

- Dynamic Analysis: Runs code and can find things like memory run away. Does a memory fingerprint and then a second memory fingerprint. (Profile in visual studio)
  Is Memory runaway a type of wear-out?

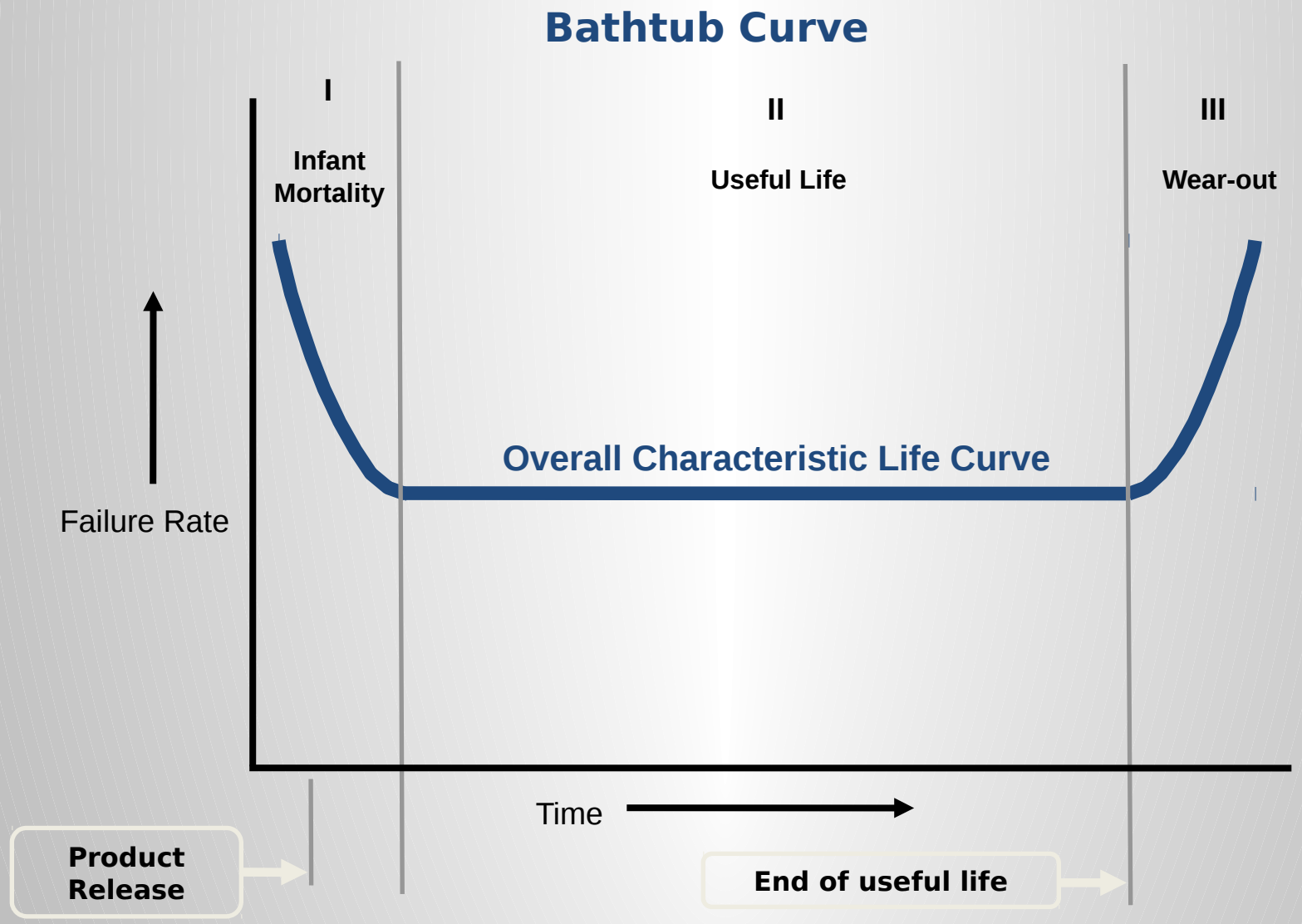
Wear-out

- We belive it doesn’t happen in software.
- There is a version that does.

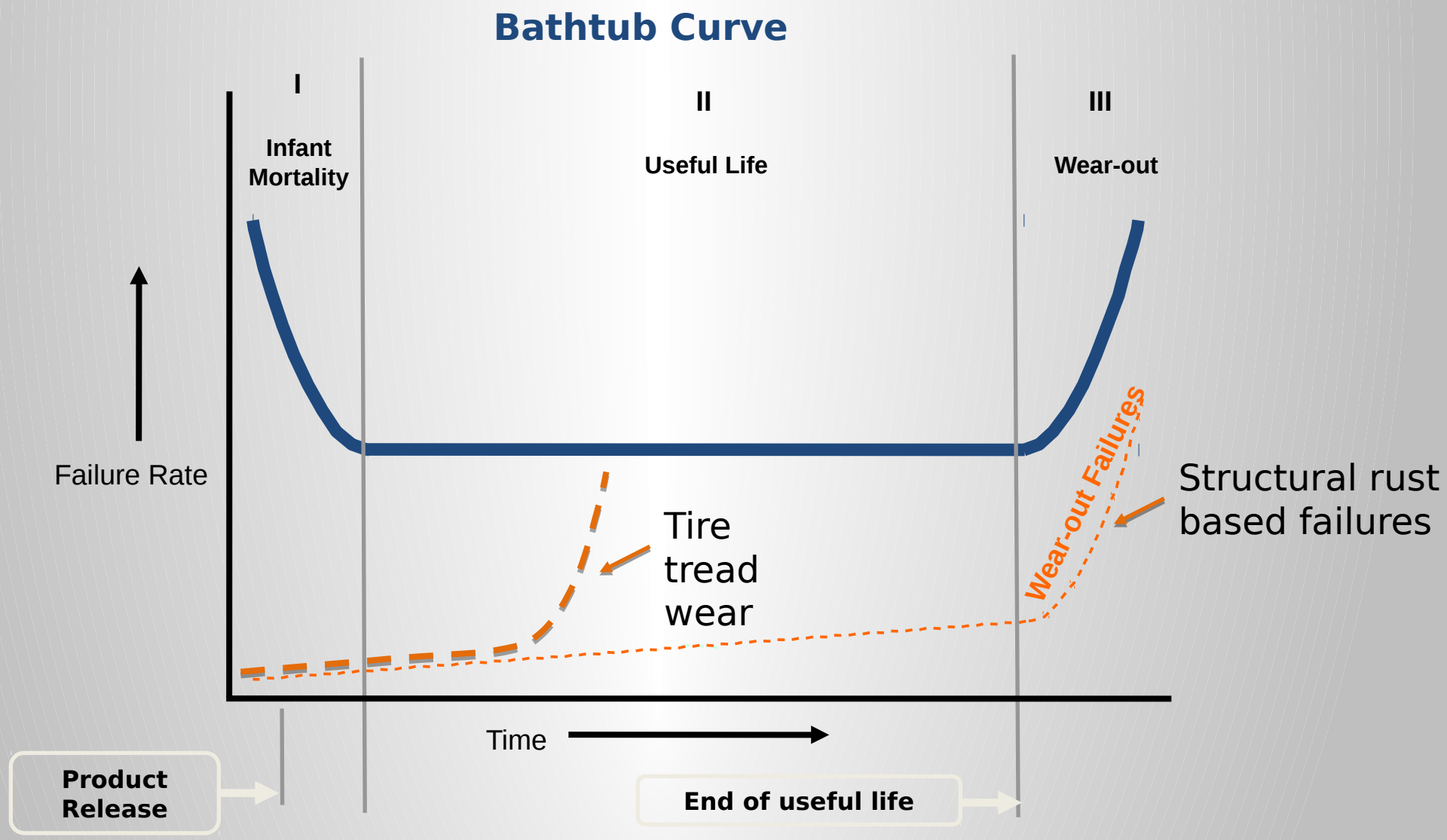
“Hi IT, I need help, my system is frozen.”

“Did you reboot it?”

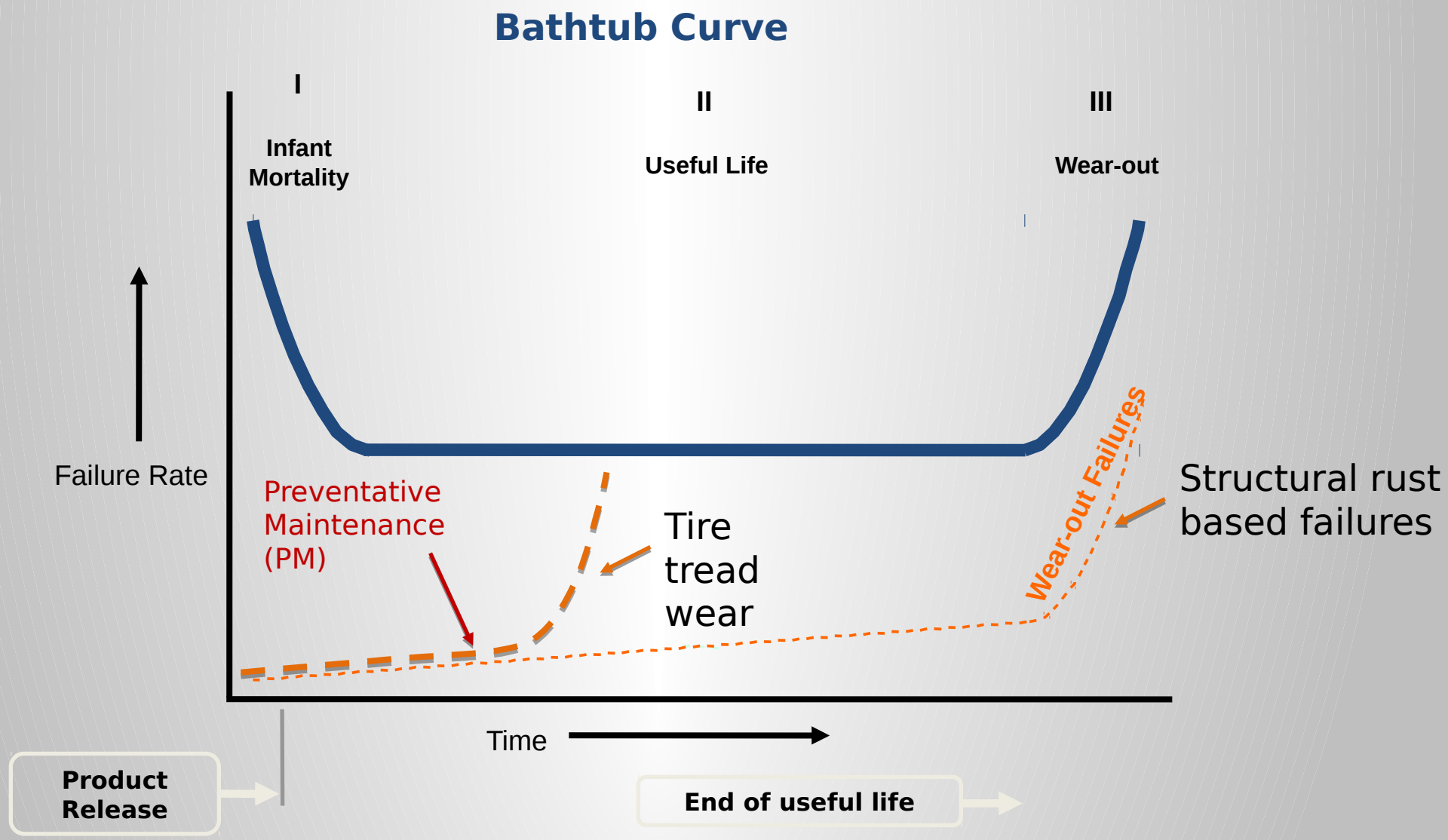
Electro mechanical wearout



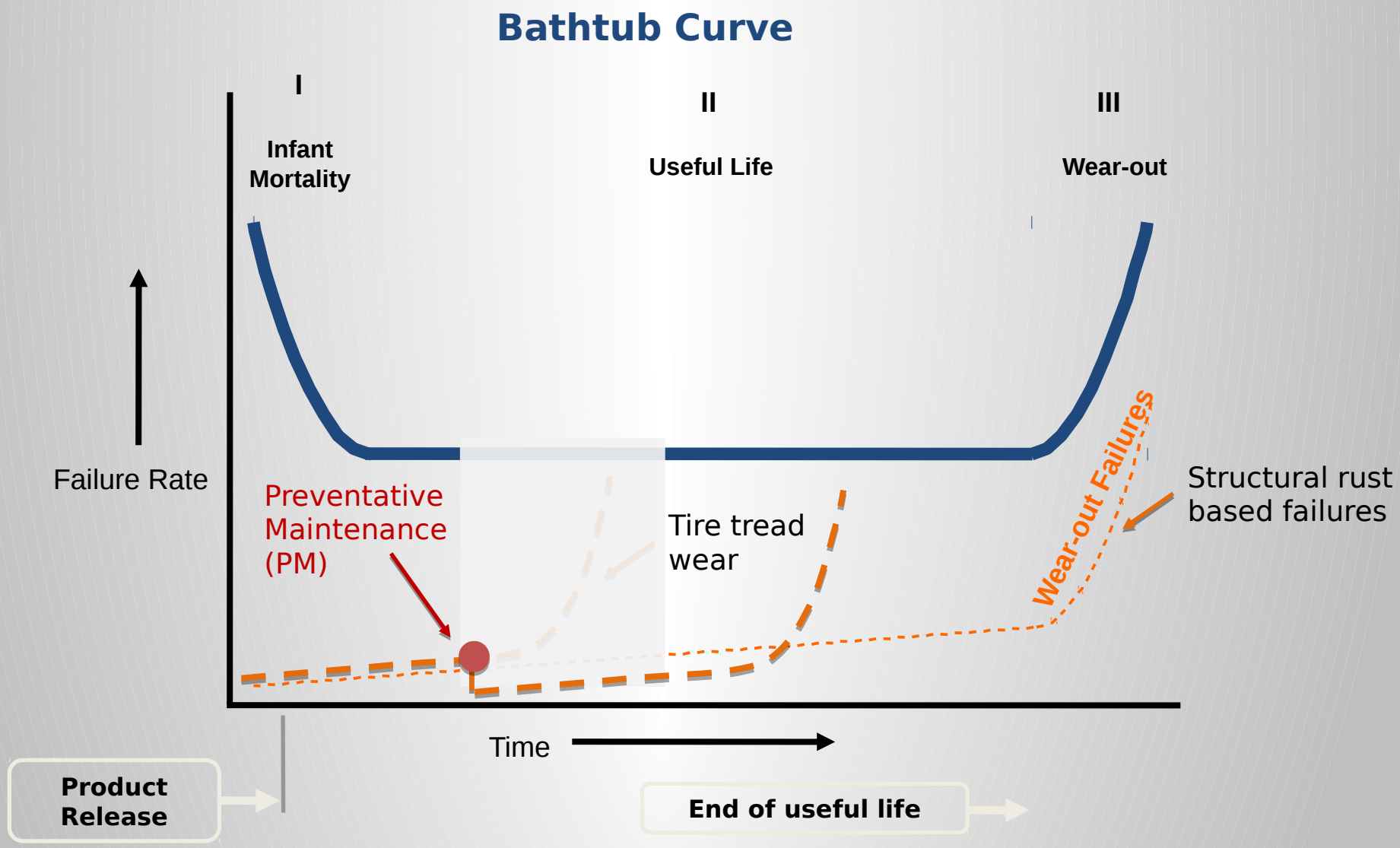
Electro mechanical wearout



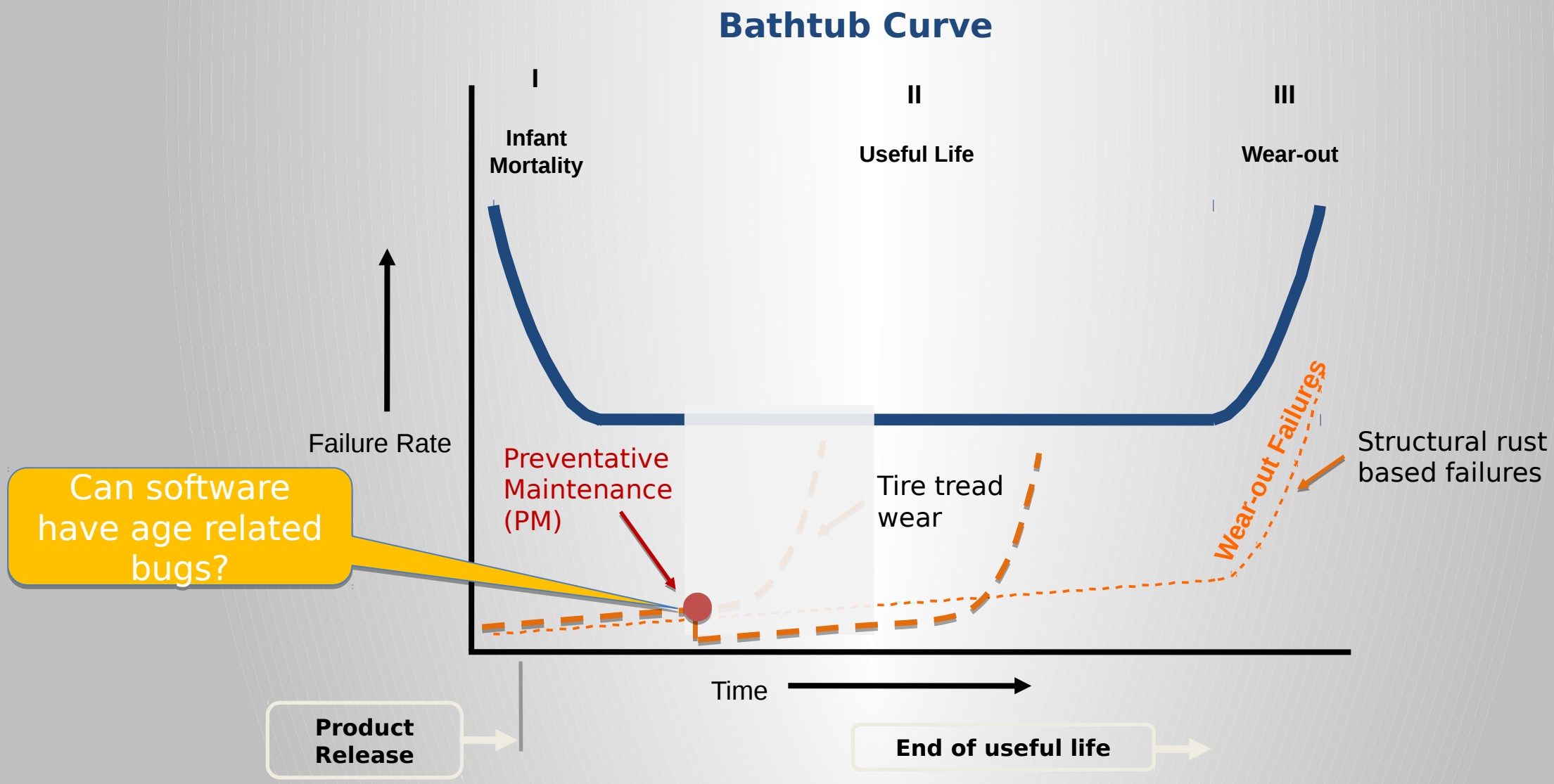
Electro mechanical wearout



Electro mechanical wearout



Electro mechanical wearout



What does it cost our customers?

APEX RIDGE RELIABILITY

amazon web services

Six hours per year

Google Docs

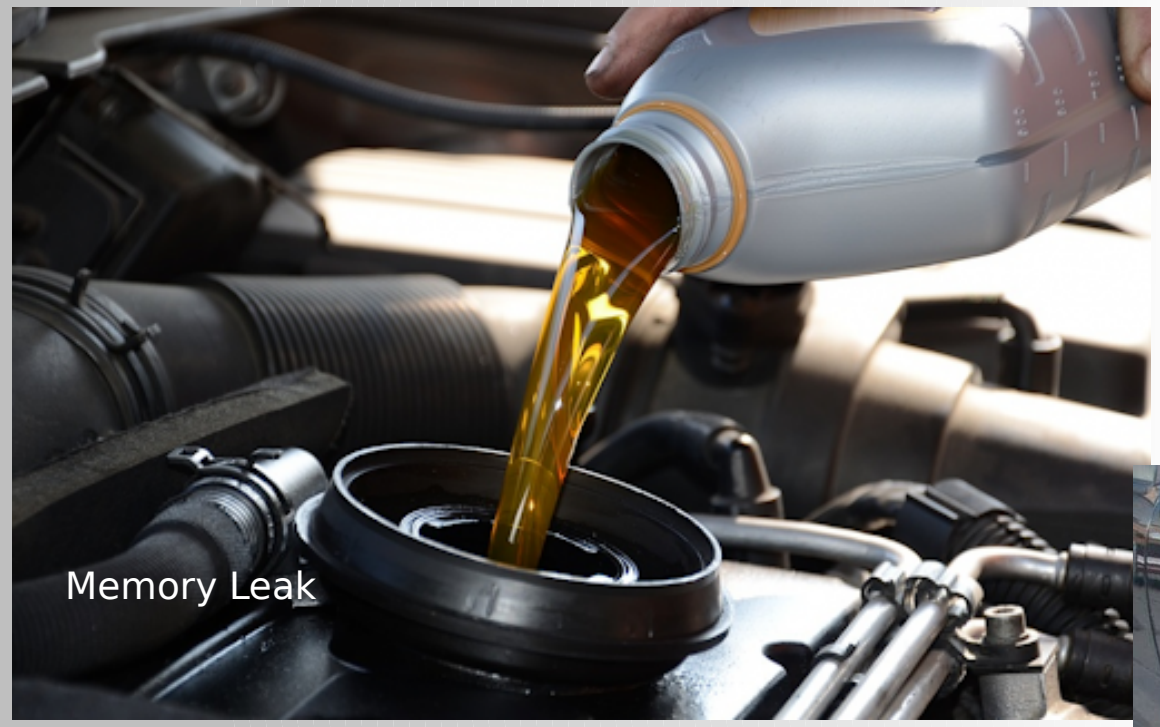
One hour per year

Aging-related Bug – Definition

Aging-related bug := A fault that leads to the accumulation of errors either inside the running application or in its system-context environment, resulting in an increased failure rate and/or degraded performance.

Example:

- A bug causing memory leaks in the application
- Note that the aging phenomenon requires a delay between (first) fault activation and failure occurrence.
- Note also that the software *appears to age* due to such a bug; there is no physical deterioration



Short term Wearout

Long term Wearout

Mandelbug

Mandelbug:= A fault whose activation and/or error propagation are complex. Typically, a Mandelbug is difficult to isolate, and/or the failures caused by a it are not systematically reproducible.

Example: A bug whose activation is scheduling-dependent:

- The residual faults in a thoroughly-tested piece of software are mainly Mandelbugs.

Mandelbug Complexity Factors

- Long time lag between fault activation and failure appearance
- Operating environment (OS resources, other applications running concurrently, hardware, network…)
- Timing among submitted operations
- Sequencing or ordering of operations

That sounds a lot like variability of environment

A failure due to a Mandelbug may not show up upon the resubmission of a workload if the operating *environment* has changed enough.

On environment: Code in protection for the things you can't control in the external environment. This could be hardweare, usb drive pulled
File gets locked as you are saving it. Maybe an antivirus grabs it and then it's locked.

- **Traditional testing** tends to be ineffective for Mandelbugs; more suitable verification strategies are
  - Code reviews
  - Model checking
  - Combinatorial testing

Bohrbug

Bohrbug: = A fault that is easily isolated and that manifests *consistently* under a well-defined set of conditions, because its activation and error propagation lack complexity.

Example: A bug causing a failure whenever the user enters a negative date of birth

- Since they are easily found, Bohrbugs may be detected and fixed during the software testing phase.

Dealing with Bohrbugs-Mandelbugs

APEX RIDGE RELIABILITY

- Depending on the type, appropriate strategies are needed
- **Traditional testing** tends to be ineffective for Mandelbugs; more suitable verification strategies are
  - Code reviews
  - Model checking
  - Combinatorial testing
  - …
- Failures caused by Mandelbugs can be **tolerated** by
  - Retrying a failed operation
  - Restarting a process
  - Failover to an identical replica
  - Rejuvenation ”Did you restart your computer first?”
  - …

Design for Reliability (Mandelbug strategy)

APEX RIDGE RELIABILITY

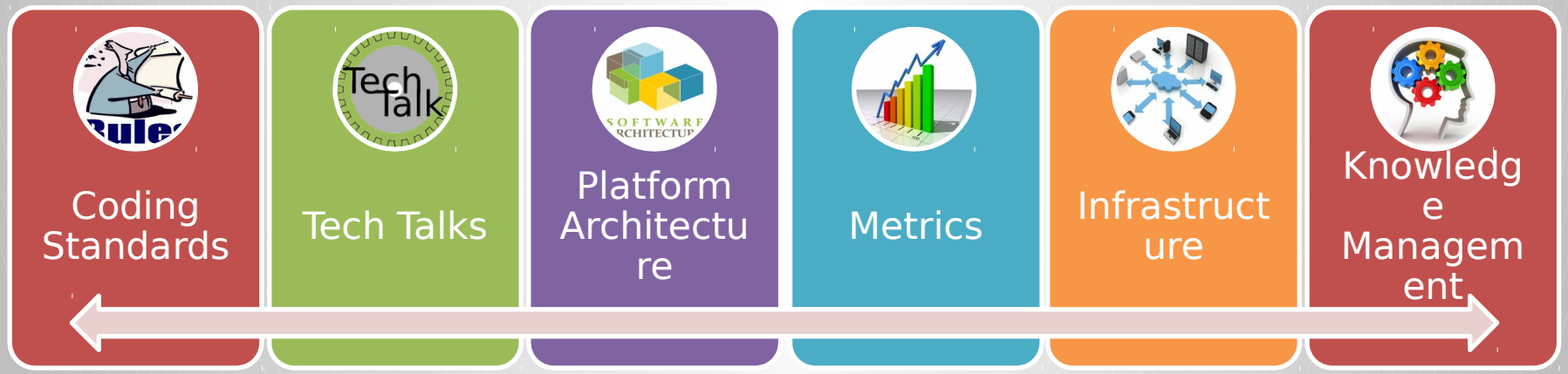
- Single Thread vs multi thread is a design pattern strategy (assist with Mandelbug)
- One mandelbug is deadlock from multithread

- Software Design Patterns
  - Design patterns are used to represent some of the best practices adapted by experienced object-oriented software developers. A design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems. It describes the problem, the solution, when to apply the solution, and its consequences.

Made by the team

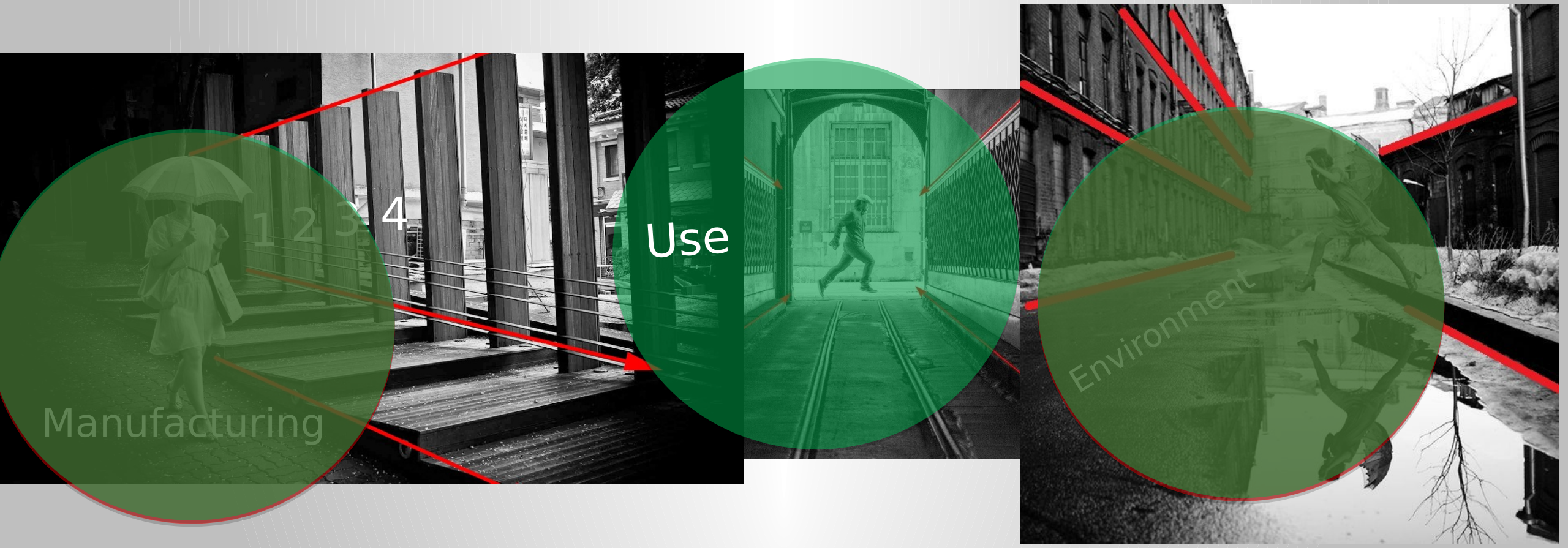
High Performance Development Teams

- Identify the developer types
  - Architects, Early Adopters, Teachers, Testers
  - Pragmatists, Sufficient Coders, Algorithm Creators
  - Strict Coders, Troubleshooters, Process Followers
- What roles do you put them in?
- What teams / functions do you need?
- How do you get the right person to the right team



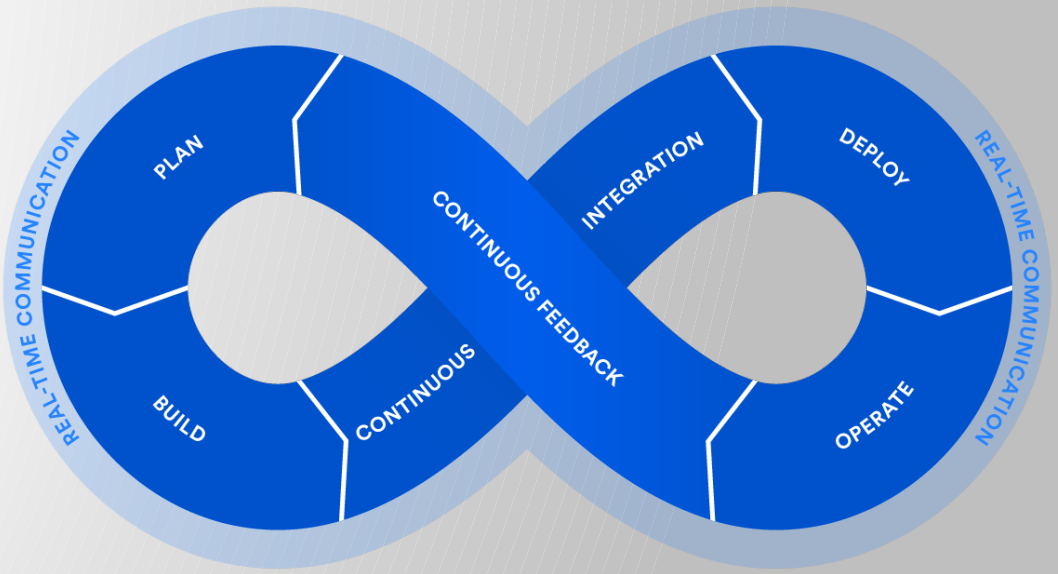

Can you teach AI to look at code and know bad code from good code?
How do we know bad code when we see it?

All three variables are there



Dev ops

- DevOps is a set of practices that combines software Development (Dev) and information-technology Operations (Ops) which aims to shorten the systems development life cycle and provide Continuous delivery with high software quality



UILD

TINUOUS
DBACK

Lifecycle

git

GitHub

GitLab

docker

SUBVERSION

plasticscm

mercurial

PERFORCE
Version everything.

eclipse

kubernetes

Cobertura

JACOCO
Java Code Coverage

Jenkins

APACHE
JMeter™

JUnit 5

Team

Quality