Foundations of Software Engineering

Lecture 16: Process: Linear to Iterative

Michael Hilton

institute for SOFTWARE RESEARCH

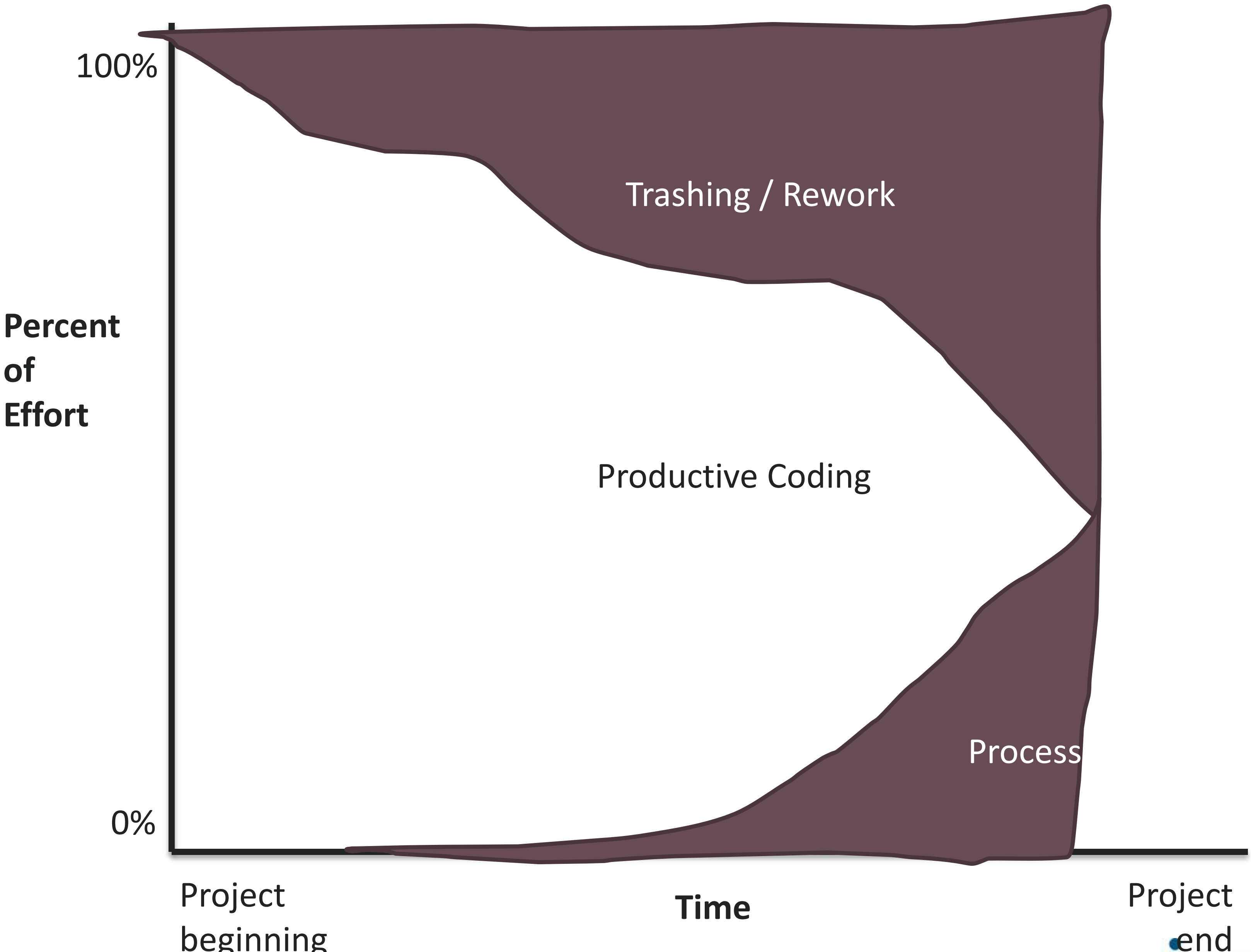
Learning goals

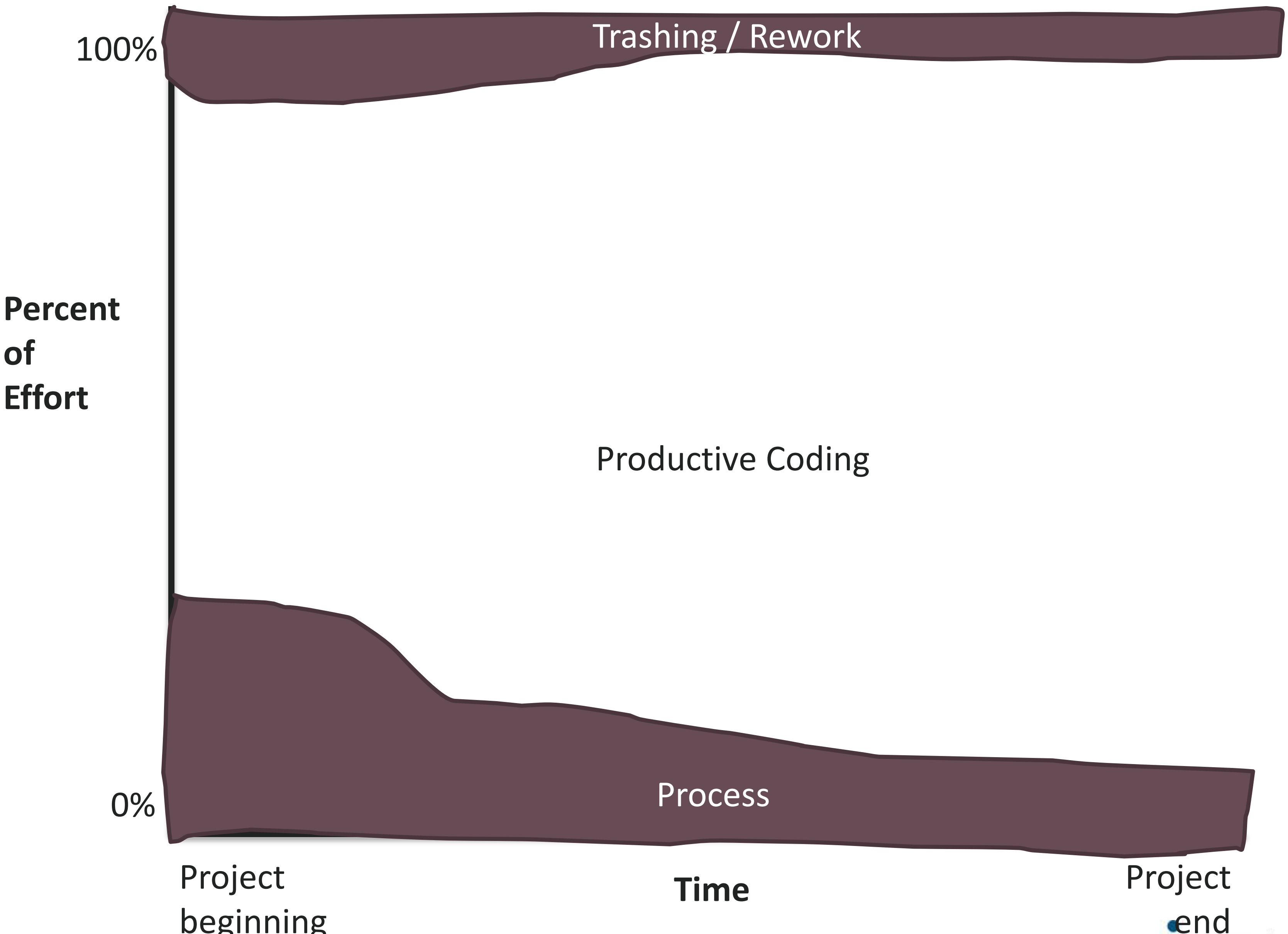
- Understand the need for process considerations
- Select a process suitable for a given project
- Address project and engineering risks through iteration
- Ensure process quality.

(Circular dependency between QA planning and process…)

A simple process

1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1

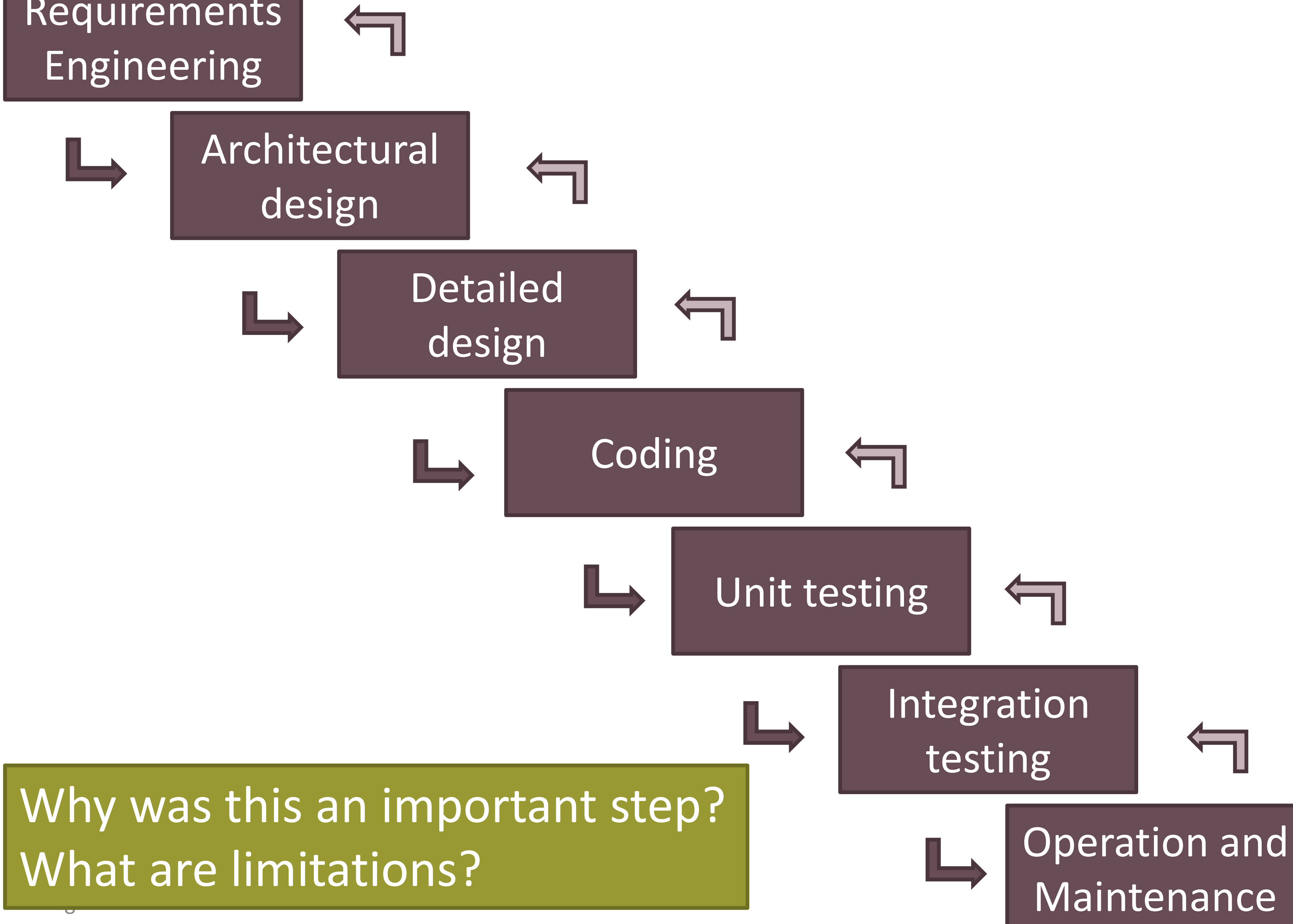

100%
Trashing / Rework
Percent
of
Effort
Productive Coding
Process
0%
Project
beginning
Time
Project
end



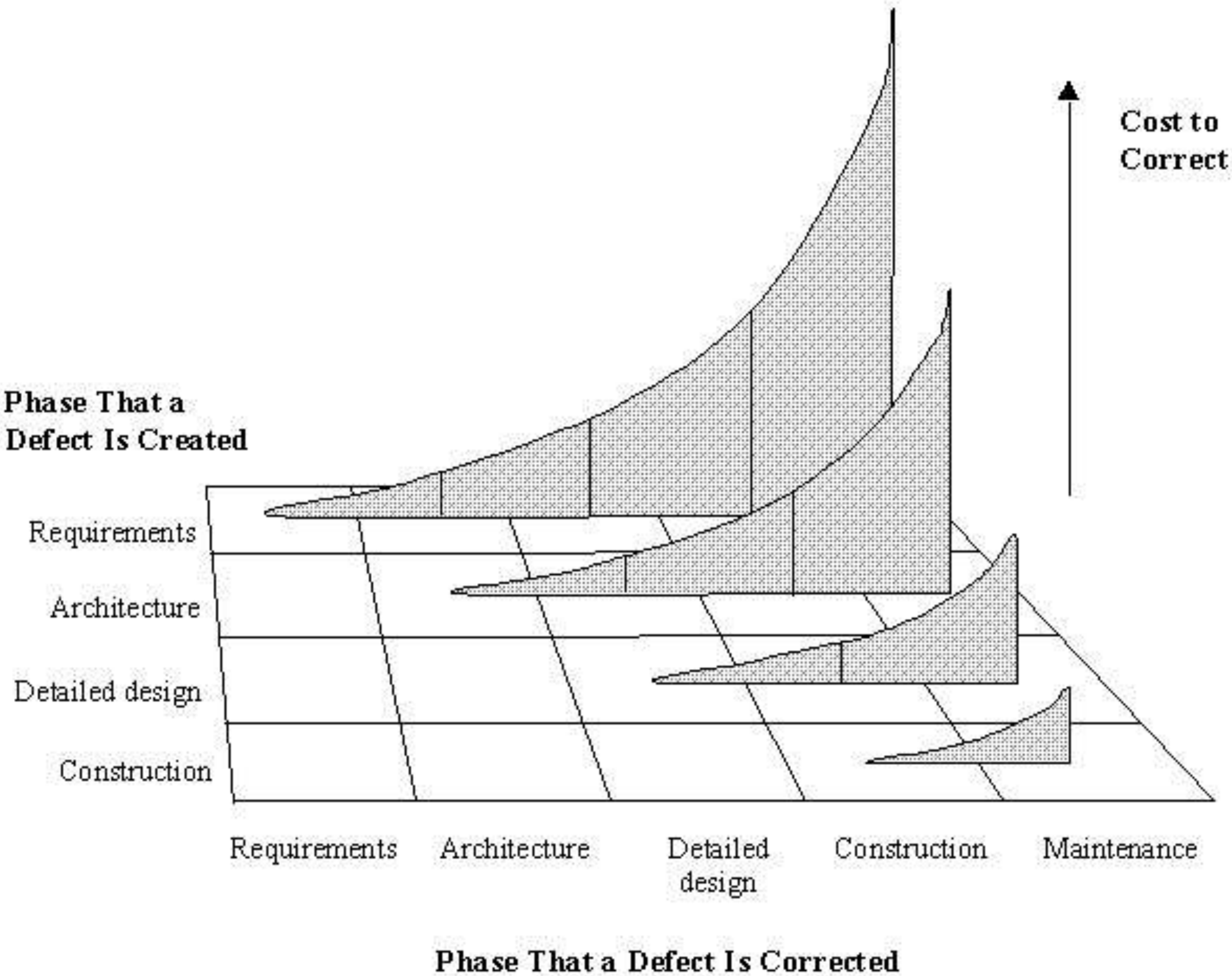
institute for SOFTWARE RESEARCH

The Waterfall Model

Win Royce and Barry Boehm, 1970



Why was this an important step?
What are limitations?


Cost to Correct
Phase That a Defect Is Created
Requirements
Architecture
Detailed design
Construction
Requirements
Architecture
Detailed design
Construction
Maintenance
Phase That a Defect Is Corrected

History lesson: 1968 NATO Conference on Software *Engineering*

- Envy of engineers: Within time, predictable, reliable.
- Provocative Title, Call for Action

1968

Envy of Engineers

- Producing a car/bridge
  - Estimable costs and risks
  - Expected results
  - High quality
- Separation between plan and production
- Simulation before construction
- Quality assurance through measurement
- Potential for automation

Software *Engineering*?

„The Establishment and use of sound ***engineering principles*** *in order to obtain* ***economically*** *software that is* ***reliable*** *and works* ***efficiently*** *on* ***real*** *machines.”*

[Bauer 1975, S. 524]

isr institute for SOFTWARE RESEARCH

waterfallconf.com

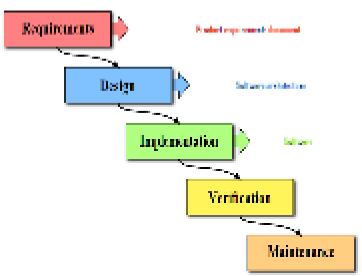
nato software engineering

Waterfall Conference 2015

Coming in Late Winter of 2015. Dedicated to all aspects of the Waterfall Model of software development.

At a glance

3 days, 150+ speakers, hundreds of Waterfall enthusiasts. Join the world's process pioneers, builders, and innovators for three intense days. Learn about the Waterfall Model, challenge your assumptions, and fire up your brain.



A conference dedicated to all aspects of the Waterfall Model of software development. Many companies are **dropping Agile, Kanban and Lean** to move back to the safe and sequential development process. As you know it is much easier to fix a requirements bug in the requirements phase than to fix that same bug in the implementation phase, as to fix a requirements bug in the implementation phase requires scrapping at least some implementation and design work which has already been completed. As you know the waterfall model provides a structured approach; the model itself progresses linearly through discrete, easily understandable and explainable phases and thus is easy to understand; it also provides easily identifiable milestones in the development process. It is for this reason that the Waterfall Conference is so popular in many software engineering companies.

Keynotes by industry leaders, sessions by real live developers and process enthusiasts. Sponsorship opportunities available.

After years of being disparaged by some in the software development

Registration Includes

- Access to all keynotes and breakout sessions
- World-Class learning experience
- Breakfast, lunch and receiptions
- Special events, including famous Waterfall Bash

Social

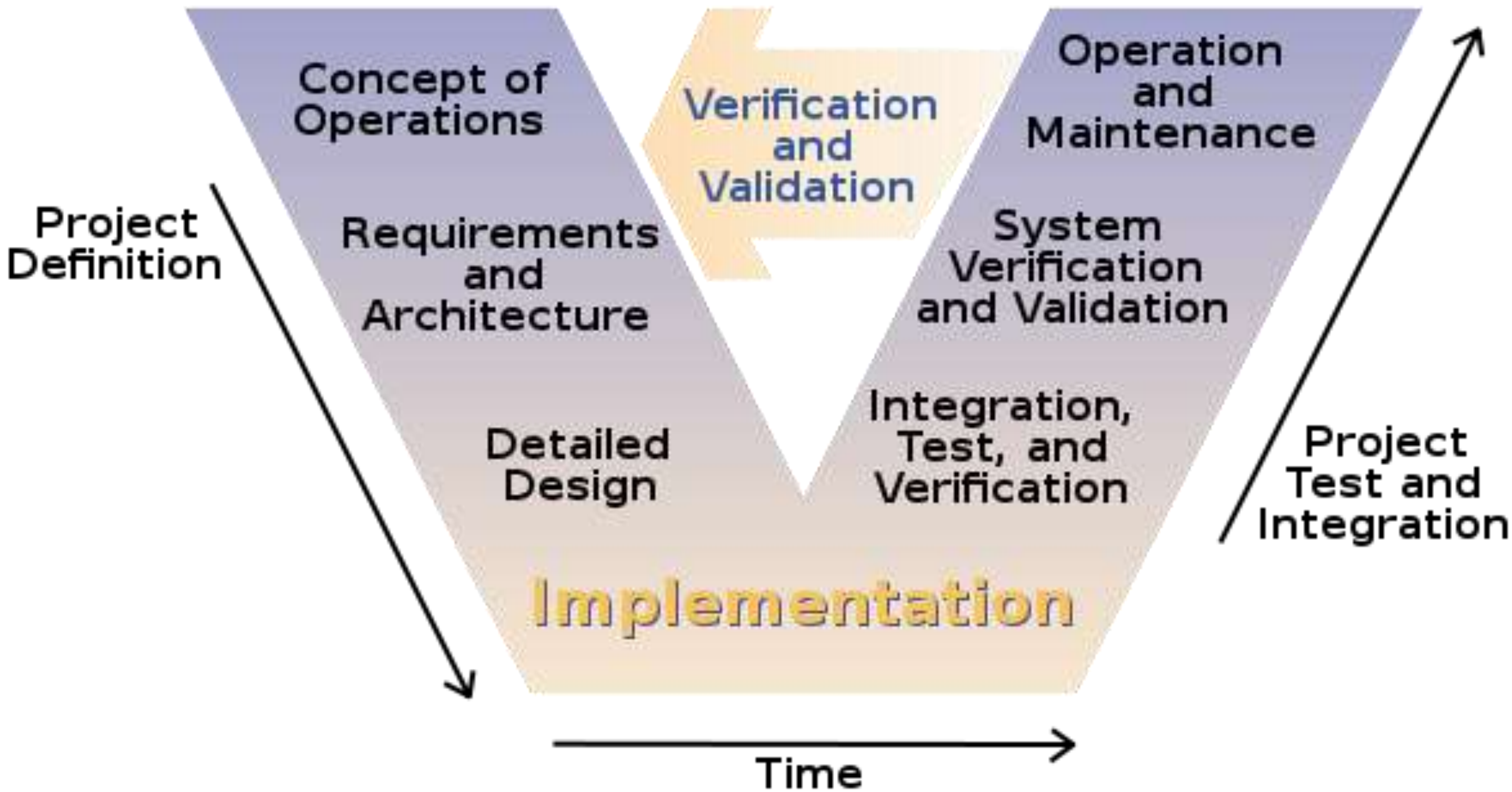
Stay up to date

RSS

Key challenge: Change

- Software seems changeable ("soft")
- Developers prone to changes and "extra features"
- Customers often do not understand what is easy to change and what is hard
- "Good enough" vs. "optimal"

The "V" Model (80s, 90s)



When is waterfall appropriate?

1. The requirements are known in advance.
2. The requirements have no unresolved, high-risk risks such as due to cost, schedule, performance, safety, security, user interfaces, organizational impacts, etc.
3. The nature of the requirements will not change very much.
4. The requirements are compatible with all the key system stakeholders' expectations.
5. The architecture for implementing the requirements is well understood.
6. There is enough time to proceed sequentially.

Early improvement: sequencing

- Enforce earlier software considerations
- Waterfall instituted at TRW in 70s, with several additional recommendations for iterations (like prototypes).
- Modeled after traditional engineering
  - blueprints before construction
  - decide what to build, build it, test it, deploy
  - Reduce change
- Successful model for routine development
- Problematic at large scale
  - Requirements -> Delays -> Surprise!

A natural engineering process?

- Decide what to build
- Build it
- Test it
- Deploy it

- Don't know what to build in advance
- Don't know all details how to build
- Struggling with testing and evaluation
- Deploy, evolve, redeploy

-> Early and frequent feedback
-> Support for constant adaptation

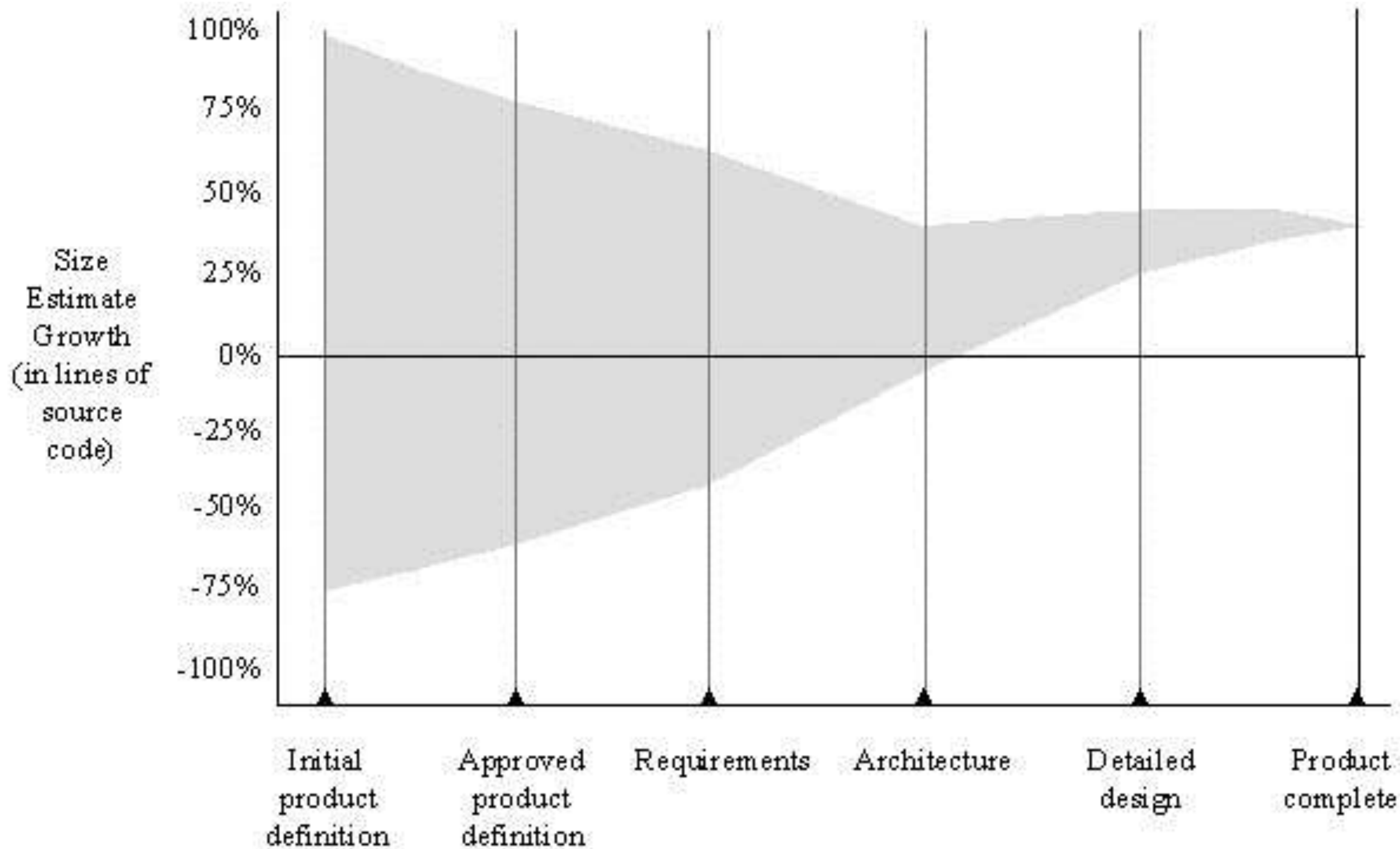
Iteration!

-> Early and frequent feedback
-> Support for constant adaptation
-> Address risks first

Software Engineering *Risks*

- Project risks
  - Projects late, buggy, cost overruns
- System risks
  - Security and safety issues
  - e.g. Toyota case
- Engineering risks
  - Unsuitable technology choices, validation issues, usability issues, scalability issues …

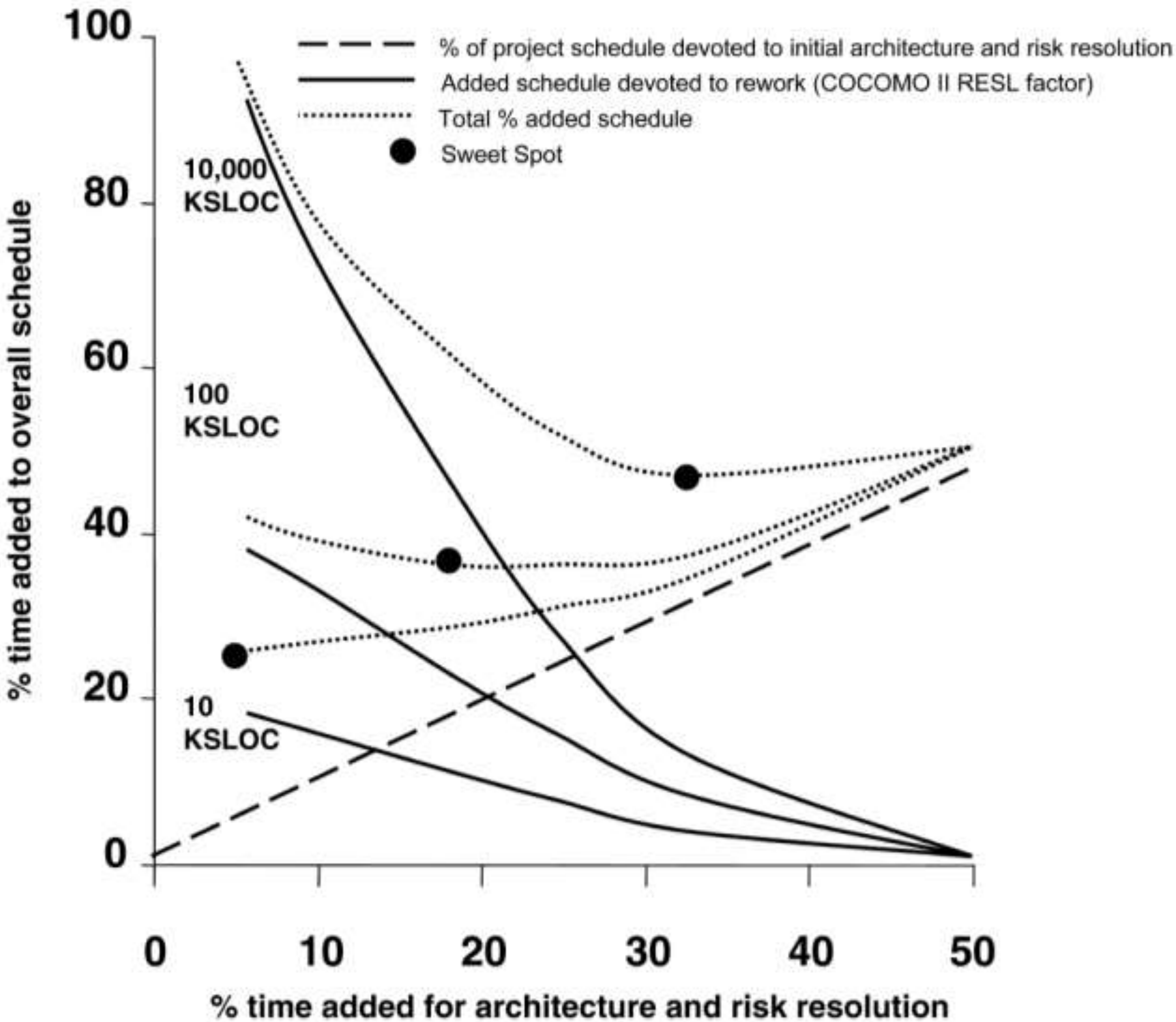
Cone of Uncertainty



Mitigation of risk through process interventions (examples)

- Risk-driven process
  - Prioritization and prototyping
- Architecture and design
  - Isolate/encapsulate risks
  - Follow industry standards
- Design for assurance
  - Preventive engineering
  - Codevelopment of system and evidence
- Functionality and usability
  - Prototypes , early usability labs

The Role of Architecture

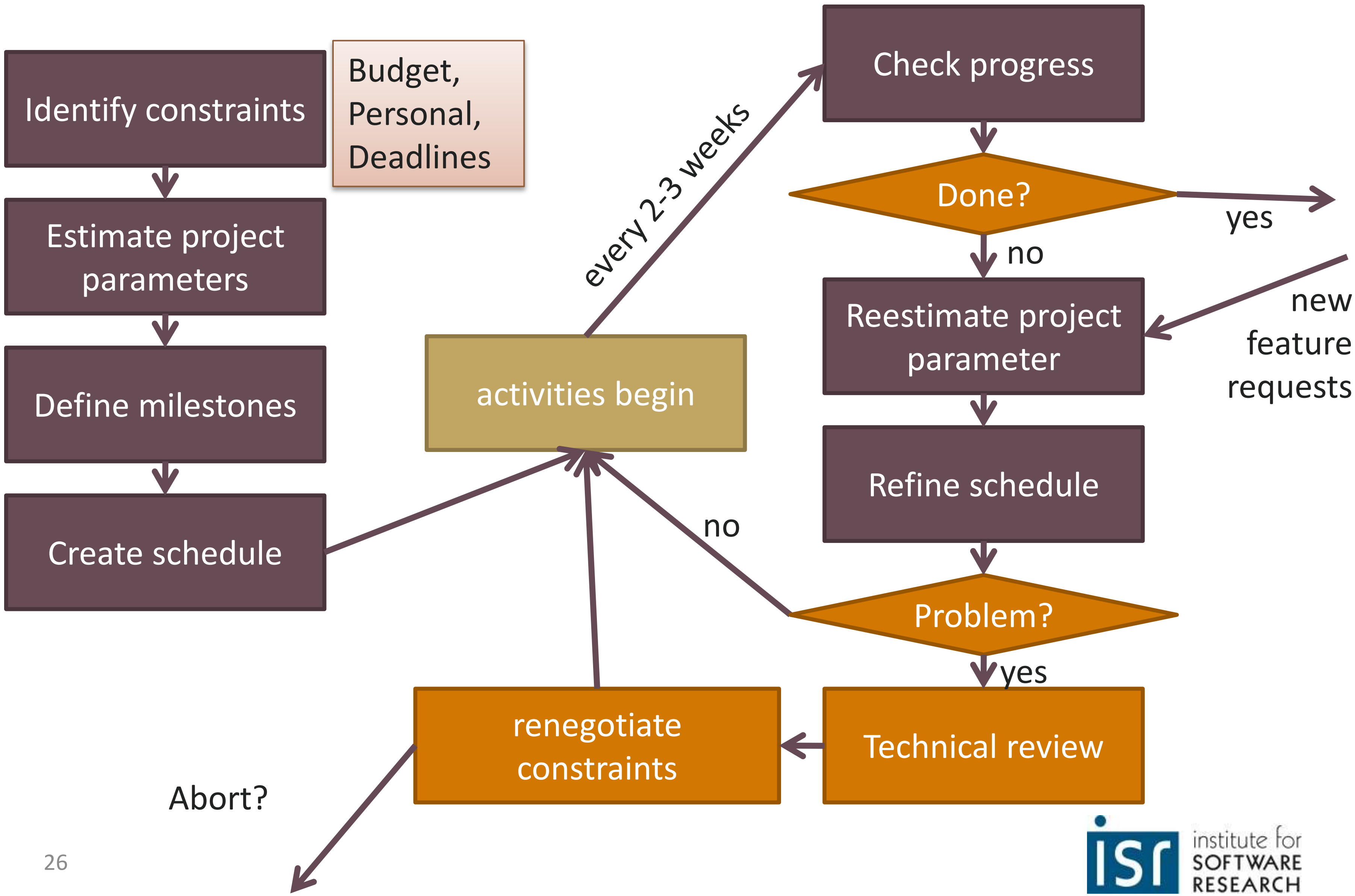


Source: Boehm, Valerdi, Honour, The ROI of Systems Engineering. 2008

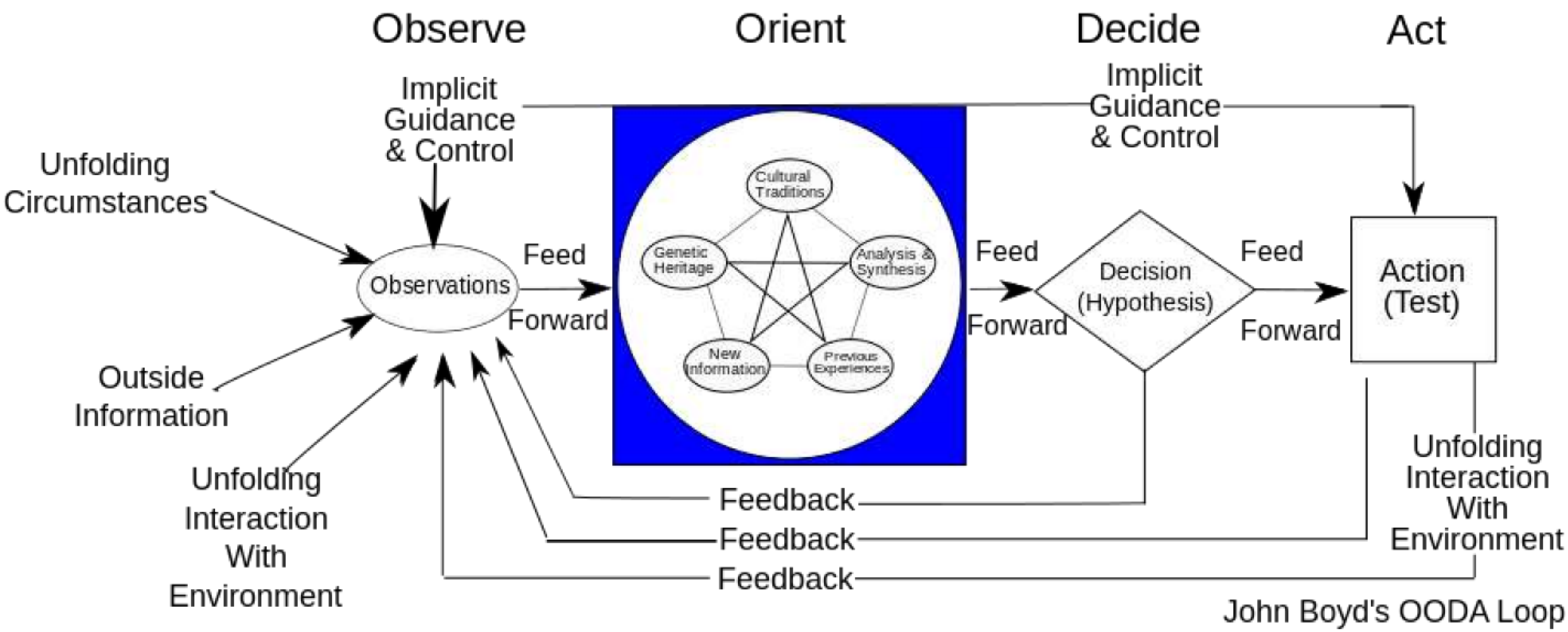
Key: Iterative Processes

- Interleaving and repeating
  - Requirements engineering, Risk assessment
  - Architecture and design
  - Implementation
  - Quality assurance
  - Deployment
- But when, in which sequence, and how often?
- What measurements can ground decisions?

Iteration in Project Management

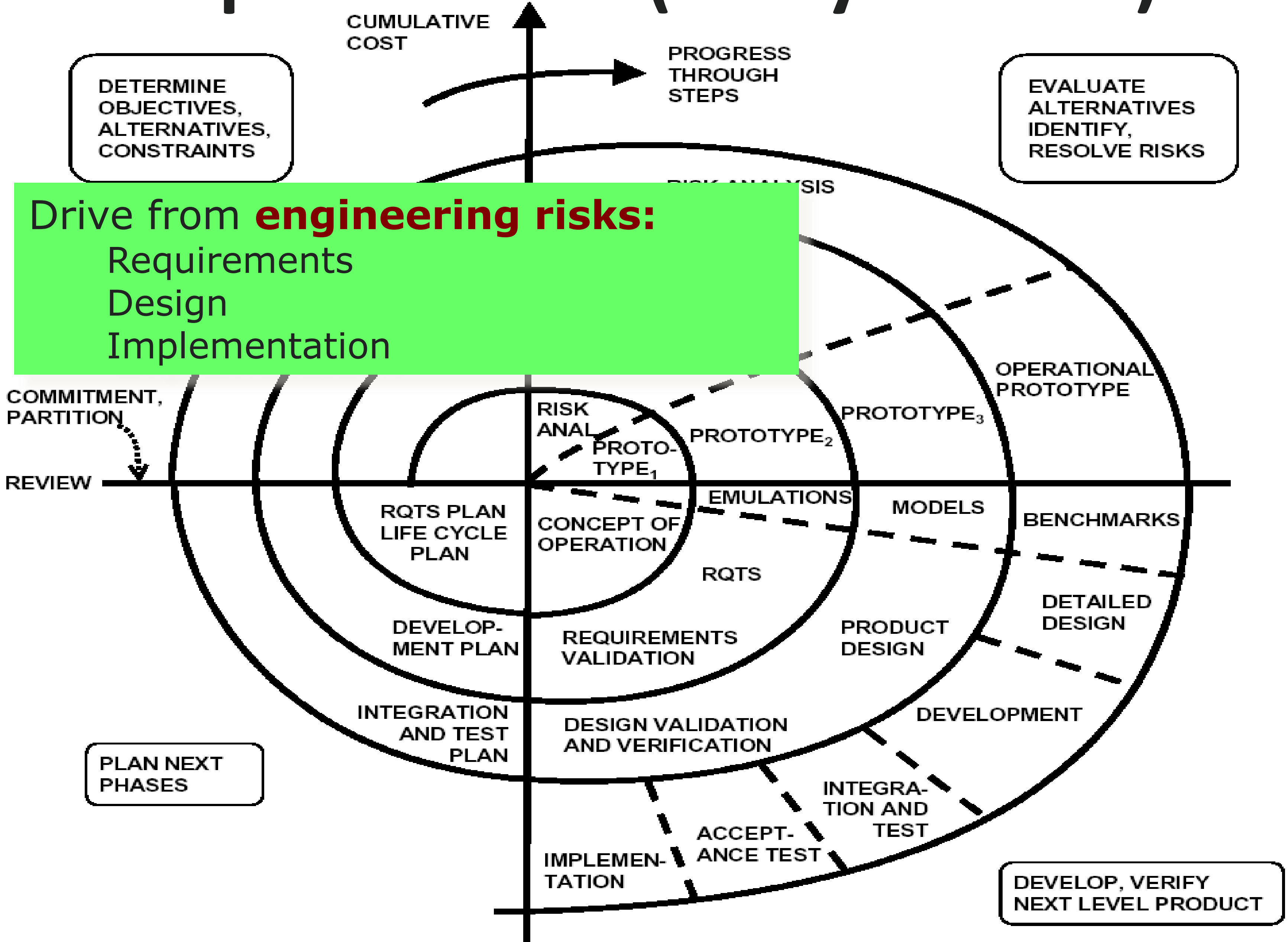


OODA Loop



institute for SOFTWARE RESEARCH

The Spiral Model (Barry Boehm)



Drive from **engineering risks:**
- Requirements
- Design
- Implementation

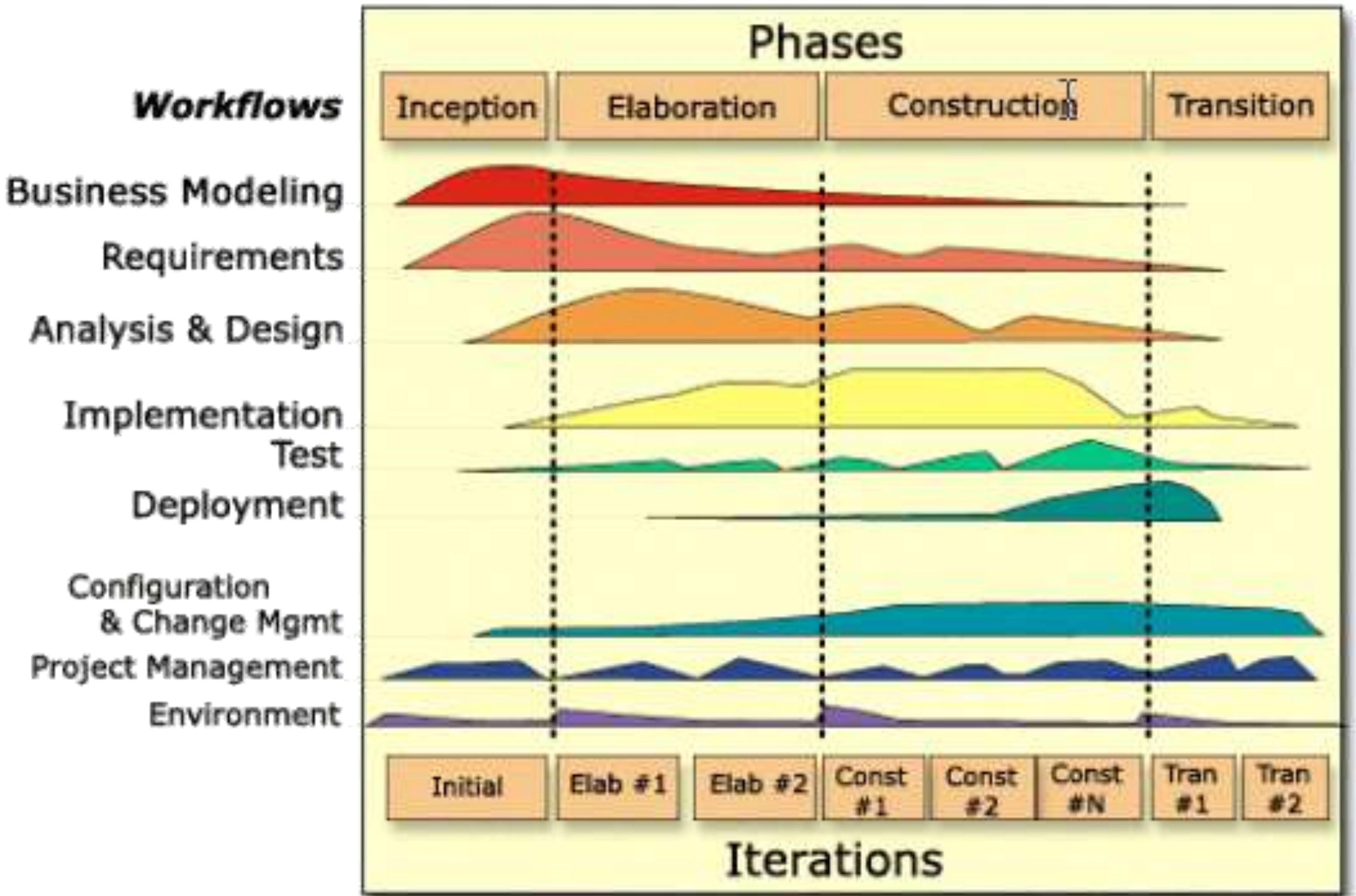
Iteration decision

- Too slow?
- Too fast?
- -> Drive via risks and measurement data; per project decision
- Contracts?

Iteration decision

- Too slow?
  - Late reaction, reduce predictability
- Too fast?
  - Overhead, reduce innovation
- "Death spiral"
  - deferred commitment, prototypes without conclusions, missing feedback loops
- -> Drive by risks and measurement data; per project decision
- Contracts?

Rational Unified Process (UP)



from Rational Software

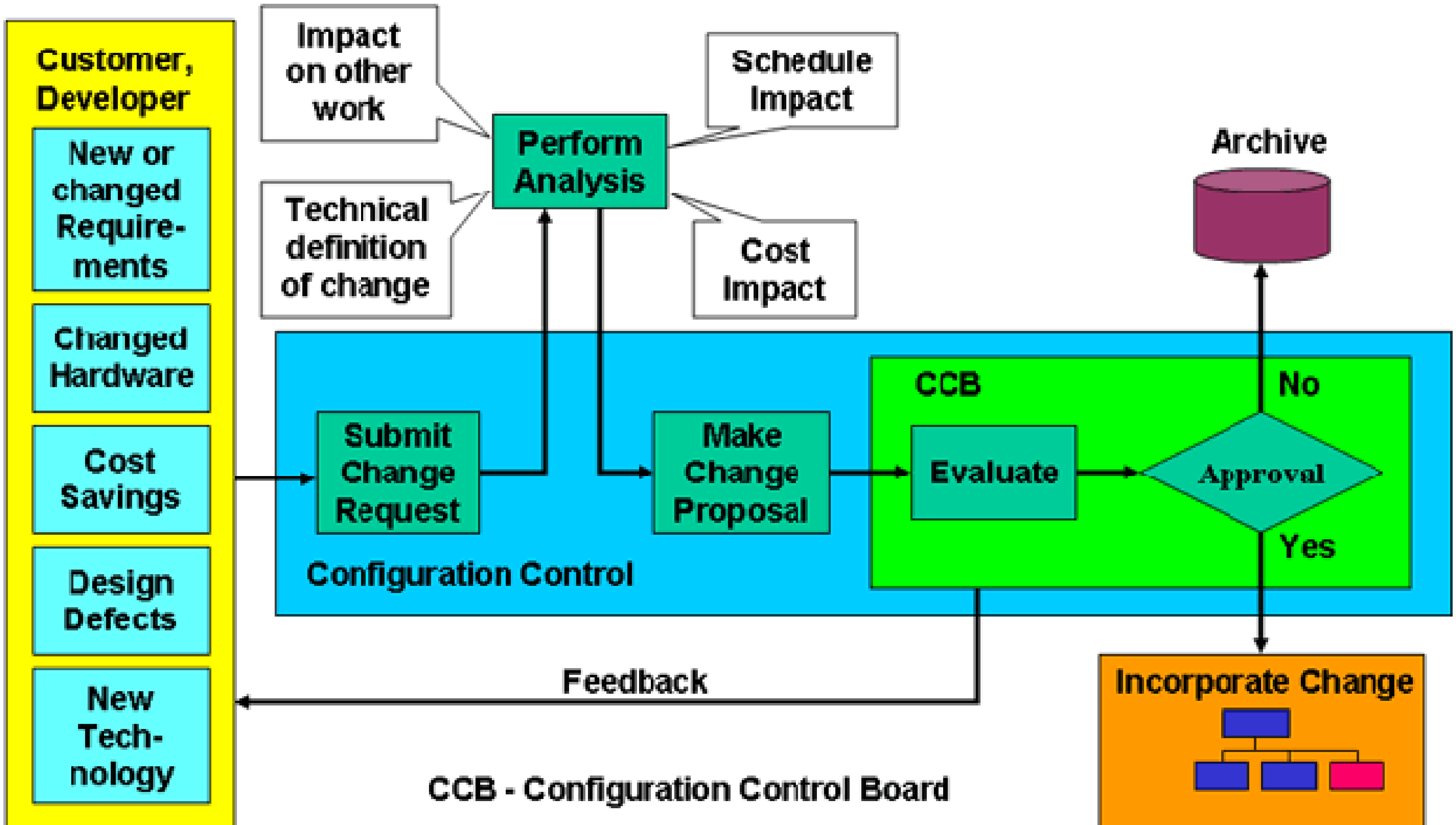
institute for SOFTWARE RESEARCH

(more on Agile, XP, Scrum, Kanban in a later lecture...)

Iterative vs. Incremental?

Change Control

Change Control Board



 www.chambers.com.au

institute for SOFTWARE RESEARCH

Change Request Form

Project: SICSA/AppProcessing **Number:** 23/02
Change requester: I. Sommerville **Date:** 20/01/09
Requested change: The status of applicants (rejected, accepted, etc.) should be shown visually in the displayed list of applicants.

Change analyzer: R. Looek **Analysis date:** 25/01/09
Components affected: ApplicantListDisplay, StatusUpdater

Associated components: StudentDatabase

Change assessment: Relatively simple to implement by changing the display color according to status. A table must be added to relate status to colors. No changes to associated components are required.

Change priority: Medium
Change implementation:
Estimated effort: 2 hours
Date to SGA app. team: 28/01/09 **CCB decision date:** 30/01/09
Decision: Accept change. Change to be implemented in Release 1.2
Change implementor: **Date of change:**
Date submitted to QA: **QA decision:**
Date submitted to CM:
Comments:

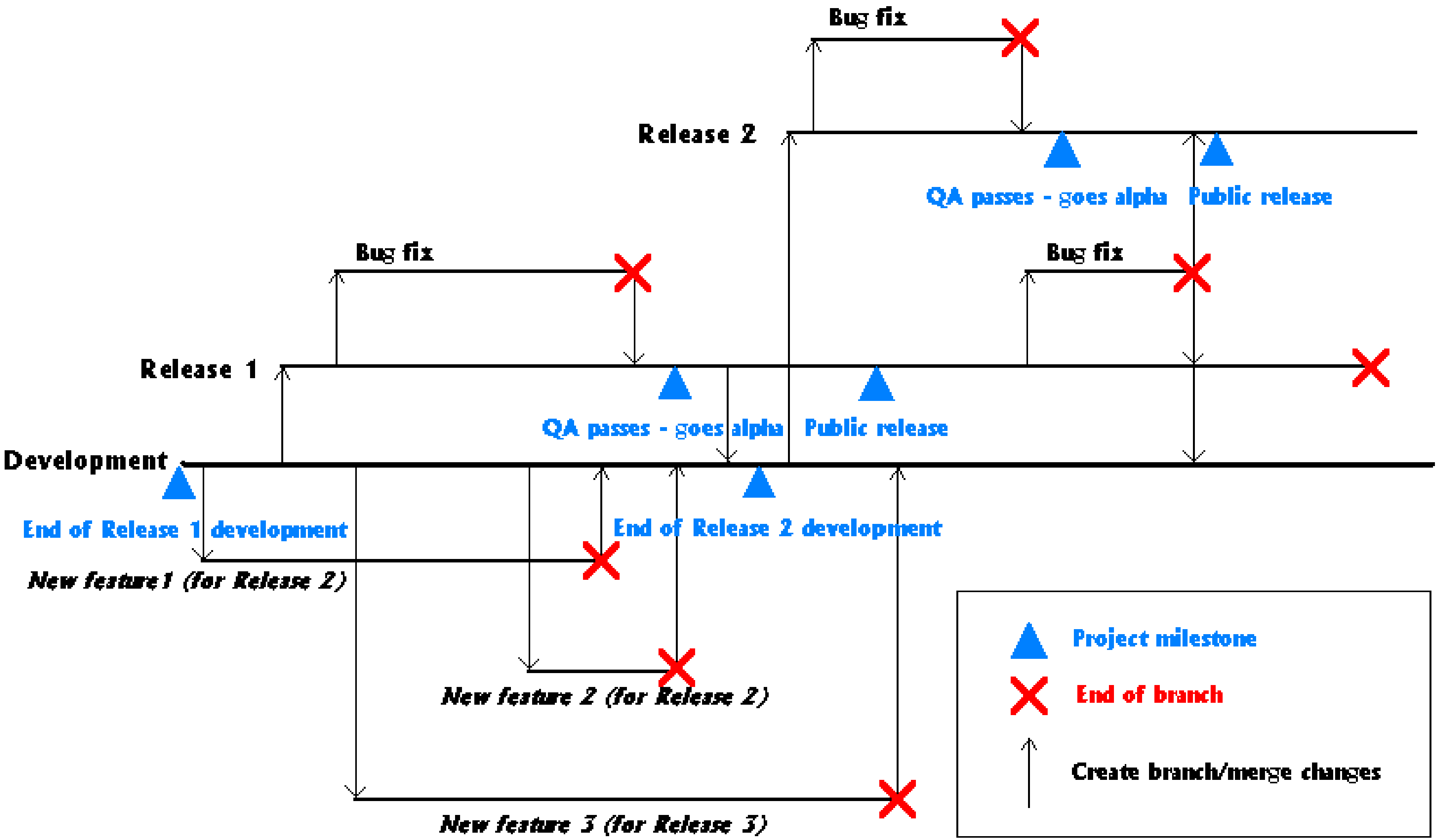
Change Impact Analysis

- Estimate effort of a change
- Analyze requirements, architecture, and code dependencies
- Tractability very valuable if available
- Various tools exist, e.g., IDE call graphs

Feature Freeze

- Pre-release phase
- Do not allow any changes except bug fixes
- Avoid destabilization

Release Planning with Branches



Case Study: Microsoft

- Microsoft plans software in features
- 3-4 milestones per release
- After each milestone reconsider which features should still be implemented
- Stabilization and freeze at end of milestone

Cusumano and Selby. Microsoft Secrets.

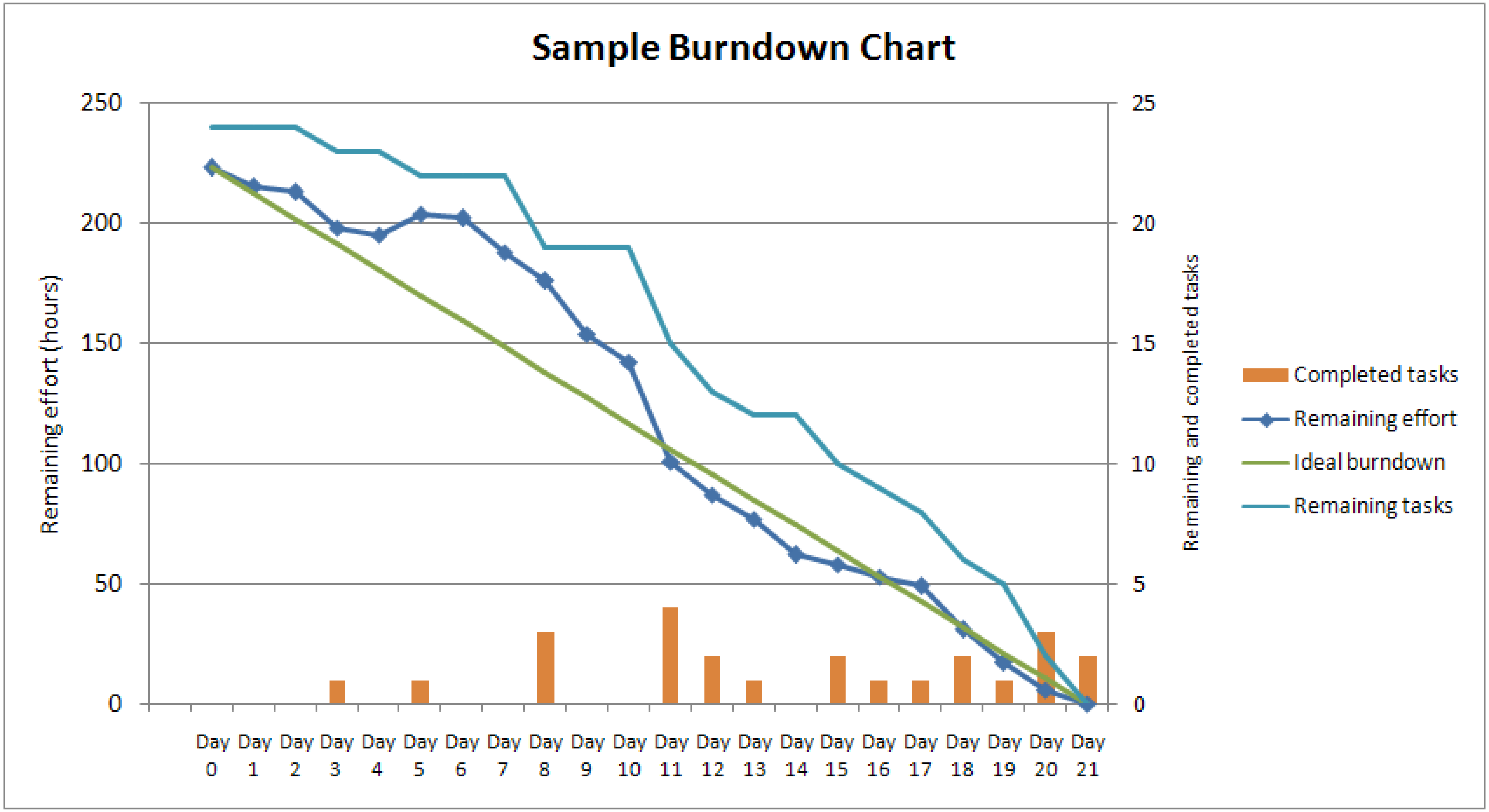
How much iteration? How much change control? (3 cases)


AK
Juneau
Honolulu
HI
WA
Olympia
Salem
OR
ID
Helena
MT
Boise
WY
Cheyenne
Sacramento
Carson City
NV
CA
Salt Lake City
UT
Denver
CO
AZ
Phoenix
Santa Fe
NM
ND
Bismark
SD
Pierre
MN
St. Paul
NE
Lincoln
KS
Topeka
IA
Des Moines
MO
Jefferson City
OK
Oklahoma City
TX
Austin
AR
Little Rock
LA
Baton Rouge
MS
Jackson
WI
Madison
IL
Springfield
MI
Lansing
IN
Indianapolis
OH
Columbus
KY
Frankfort
TN
Nashville
AL
Montgomery
GA
Atlanta
FL
Tallahassee
SC
Columbia
NC
Raleigh
WV
Charleston
Richmond, VA
PA
Harrisburg
NY
Albany
Montpelier, VT
Augusta
Concord
Boston, MA
Providence
Hartford, CT
Trenton, NJ
Dover, DE
Annapolis, MD

Process metrics

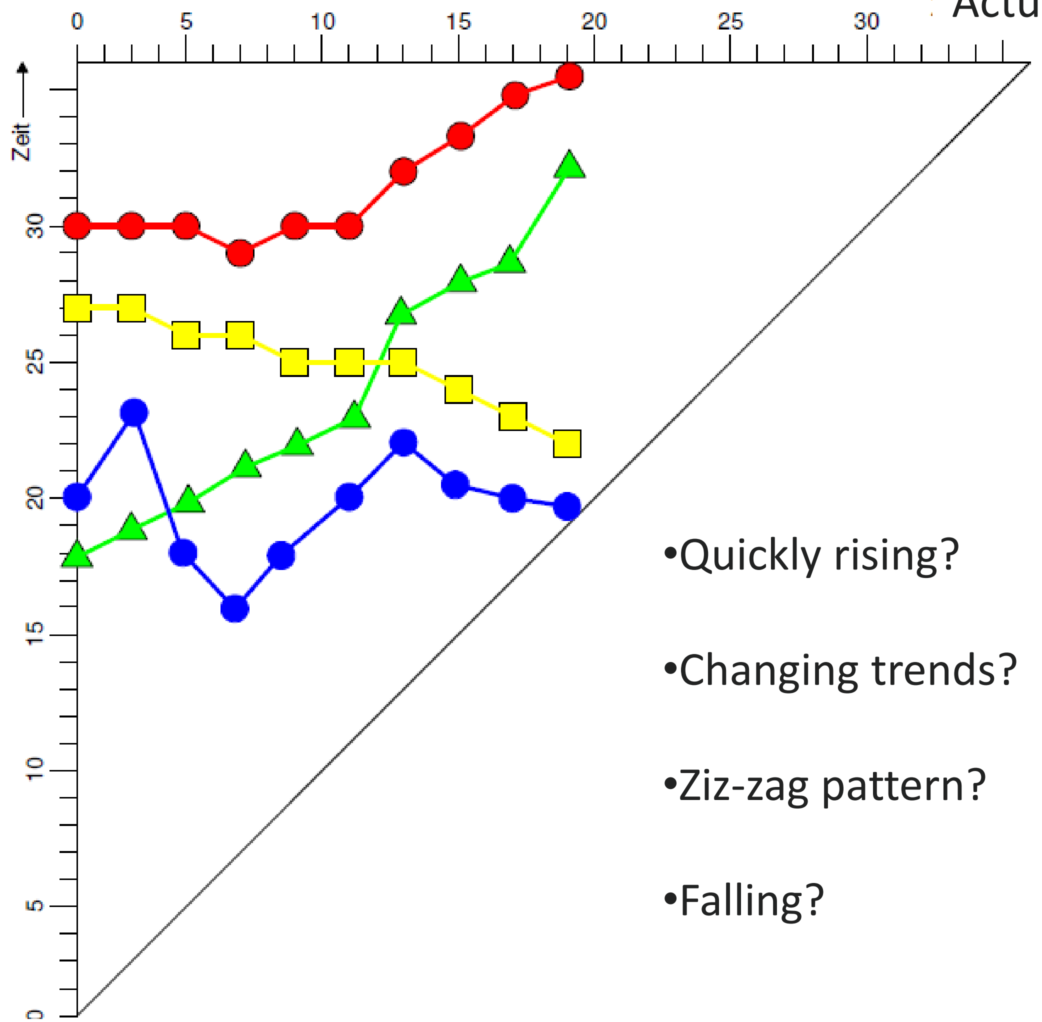
Discussion: what is the purpose of tracking process?

Burn Down Charts



institute for SOFTWARE RESEARCH

Milestone Trend Analysis



Estimated completion time

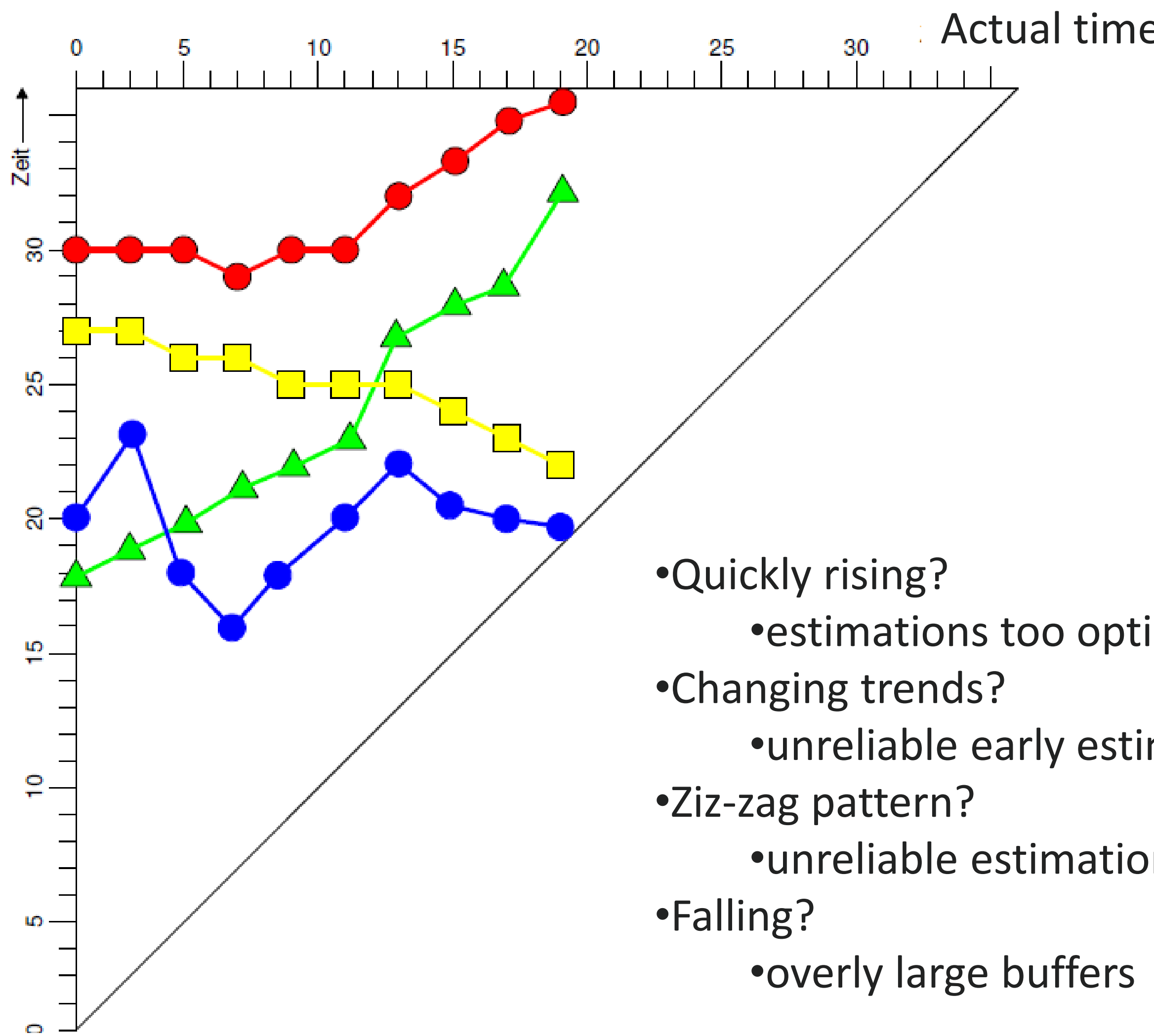
•Quickly rising?

•Changing trends?

•Ziz-zag pattern?

•Falling?

Milestone Trend Analysis



Estimated completion time

- Quickly rising?
  - estimations too optimistic
- Changing trends?
  - unreliable early estimations
- Ziz-zag pattern?
  - unreliable estimations
- Falling?
  - overly large buffers

Process metrics: Quality

- Bugs reported?
- Bugs fixed?
- Evidence of completed QA activities
  - "Test coverage", inspection completed, usability study, …
- Performance analysis?

Goodhart's law

"When a measure becomes a target, it ceases to be a good measure."

Process quality.

Discussion: what makes a good process?

Process evaluation

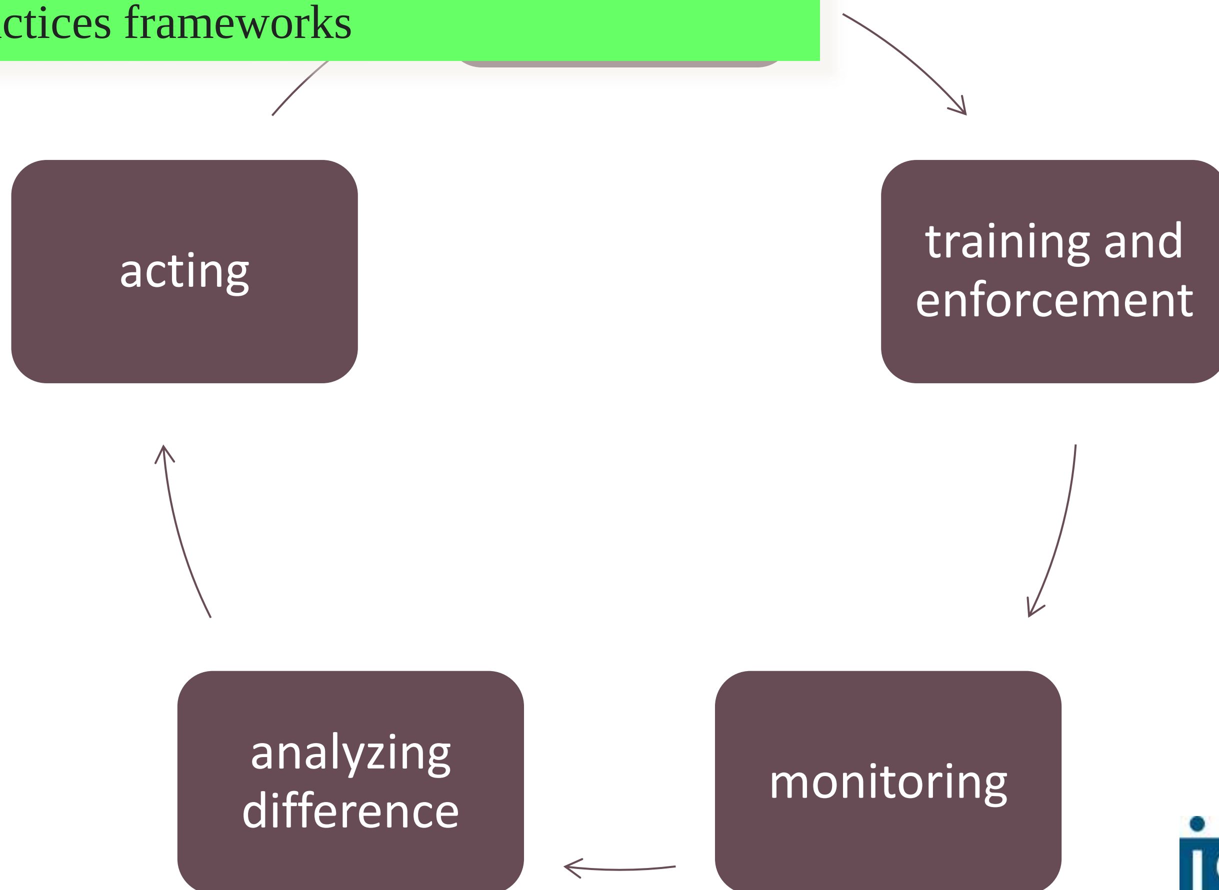
- How predictable are our projects?

- 33% of organizations collect productivity and efficiency data
- 8% collect quality data
- 60% do not monitor their processes

Process improvement loop

High-level approaches:

- Opportunistic, based on double-loop learning.
- Analytic, based on measurement + principles
- Best practices frameworks



Defect Prevention Process, IBM 1985

- When a mishap occurs:
  1. Take corrective action
  2. Conduct root cause analysis (Root cause(s): Management, people, process, equipment, material, environment):
     - Why did the mishap occur? Why was it not detected earlier?
     - Is there a trend indicating a broader problem? Can we address it?
     - What went right during this last stage? What went wrong?
  3. Implement preventive actions within the team context
- Successful changes are percolate up to corporate level.

Six Sigma, Motorola 1985

“Six Sigma seeks to improve the quality of process outputs, reducing the defects to 3.4 per million, by identifying and removing their causes and minimizing variability. It is applicable to manufacturing and services. It uses statistical methods, and creates a special infrastructure of people within the organization ("Champions", "Black Belts", "Green Belts”) who are experts in them.”

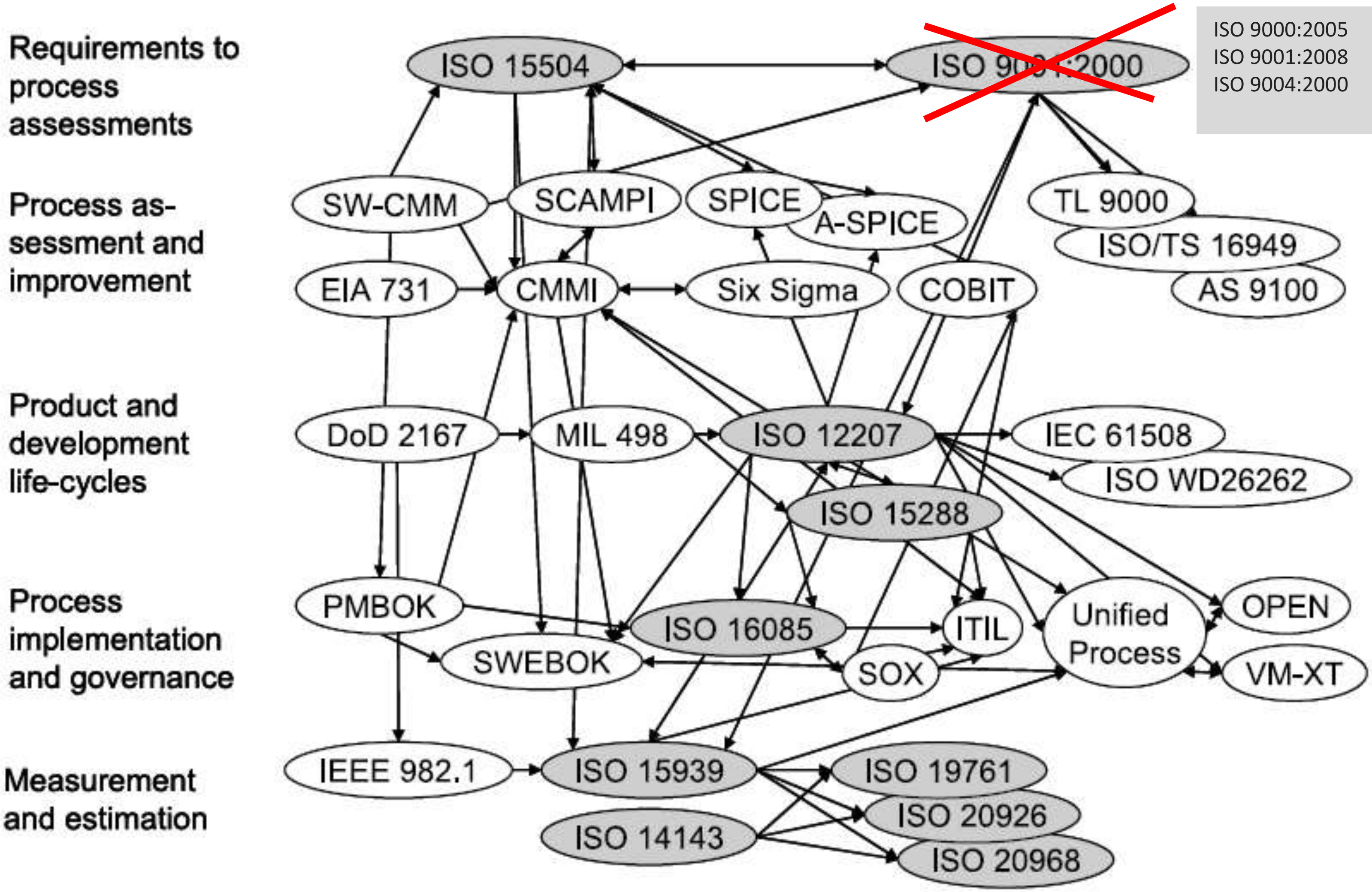
DMAIC, Existing products and services

- Define
- Measure
- Analyze
- Improve
- Control

DMADV & DFSS, New or redesigned products and services

- Define
- Measure
- Analyze
- Design
- Verify

Process standards…



C. Ebert and R. Dumke, Software Measurement,: Establish – Extract – Evaluate – Execute, 2007

SEI's Capability Maturity Model Integration

- Not a process, but a meta-process
  - Primarily used by the US government to control estimates from software vendors
  - Would prefer to accept a higher, more stable estimate.
- CMMI measures how well a company measures their own process

institute for SOFTWARE RESEARCH

The CMMI Framework

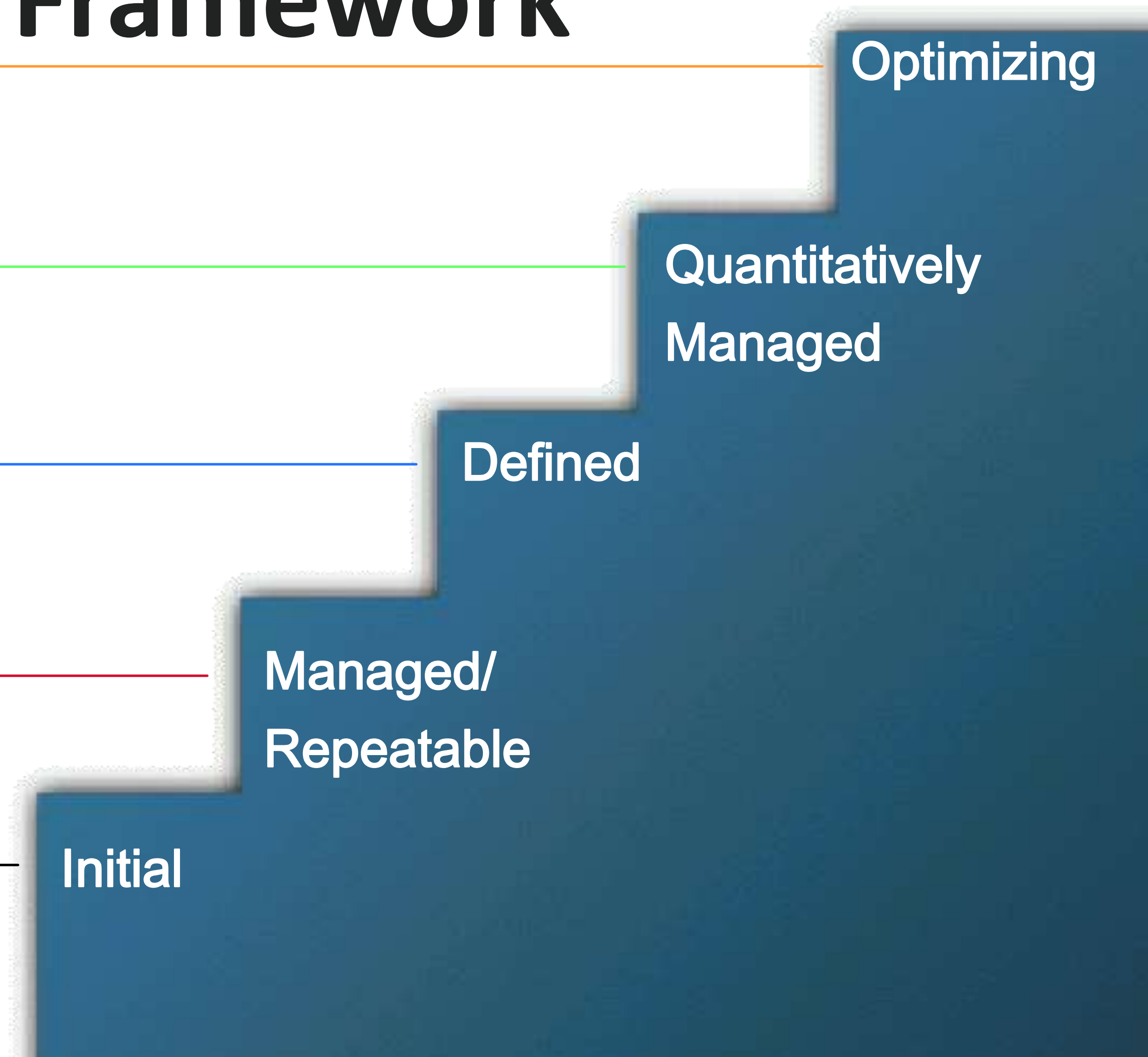
5 Focus on process improvement

4 Process measured and controlled

3 Process characterized, fairly well understood

2 Projects can repeat previously mastered tasks

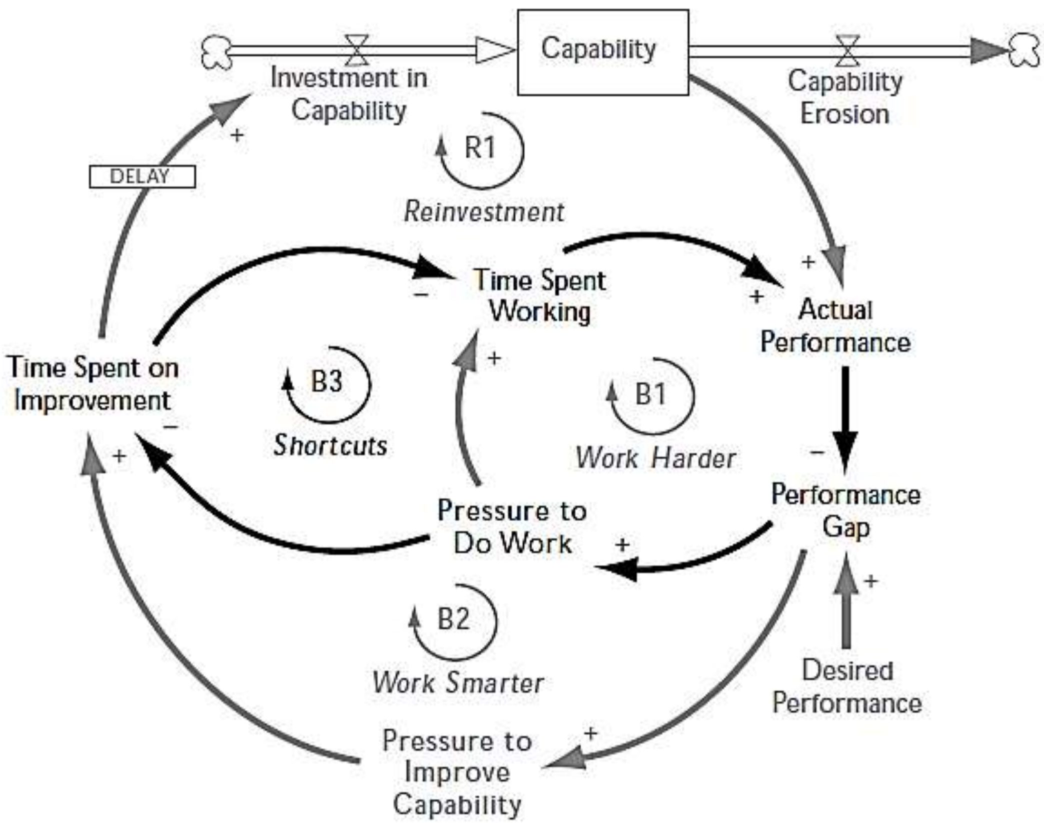
1 Process unpredictable and poorly controlled

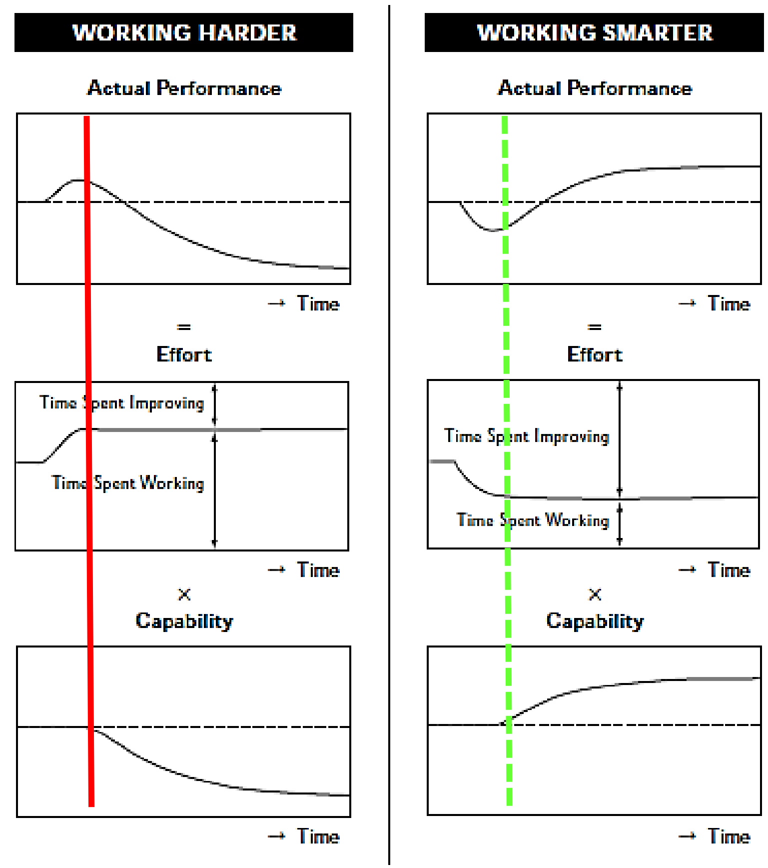


Process Tradeoffs

- (Note: Success stories in many industrial settings, eg. automobile industry.)
- Process vs product quality. Process Quality influences Product Quality, but does not guarantee it
- Following "best practices" as legal defense strategy
  - “Check box compliance”?

Increased output vs. increased process





N. Repening & J. Sterman, Nobody Ever Gets Credit For Fixing Problems That Never Happened: Creating And Sustaining Process Improvement, 2001

Scenario 1

- You work at an internet company on a large, existing code base.
- A bug manifests in a client-facing product, affecting profits.
- What do you do?

Scenario 2

- You run a small software firm with a handful of really smart engineers.
- Your employees keep having great ideas and building awesome products!
- …but they’re consistently beat to market by your competitors.
- What do you do?

Scenario 3

- You’re a large consulting firm that works on fixed cost engagements.
- A major client is threatening to cancel such a contract and cease contracting work to you in the future because you are late on several key milestones on two of your engagements with them.
- What do you do?

Summary

- Sequential process models emphasized "think before coding"
- Often too rigid, with changing requirements and environments
- Iteration to address risks
- Change management to control change
- Measure process, continuously improve process