Carnegie Mellon University Qatar

Software Risk Management

17-313 Fall 2023

Foundations of Software Engineering

Administrivia

- Midterm: Tuesday, October 3
- Participation activity: Teamwork Survey due Thursday 11:59 pm

Risk


WE'LL NEED A RISK ANALYSIS ON THIS PROJECT BEFORE I CAN APPROVE IT.
S. Adams
www.unitedmedia.com
RISK 1: INDECISIVENESS
RISK 2: OVERANALYSIS
RISK 3: CLUELESSNESS
RISK 4: MICROMANAGE-MENT...
CLICK CLICK CLICK
11/8/97
© 1997 United Feature Syndicate, Inc.
I DON'T UNDERSTAND THESE RISKS.
THAT'S NUMBER THIRTY-SIX.

Definition: Risk

Risk is a measure of the potential inability to achieve overall program objectives within defined cost, schedule, and technical constraints.

Conrow, E. 2003. Effective Risk Management: Some Keys to Success, 2nd ed. Reston, VA, USA: American Institute of Aeronautics and Astronautics (AIAA).

Risk is defined by two key components

The **probability** (or likelihood) of failing to achieve a particular outcome

The **consequences** (or impact) of failing to achieve that outcomes

Conrow, E. 2003. Effective Risk Management: Some Keys to Success, 2nd ed. Reston, VA, USA: American Institute of Aeronautics and Astronautics (AIAA).

Internal vs. External Risk

Risks that we **can** control

Risks that we **cannot** control

Levels of Risk Management

1. **Crisis management:** Fire fighting; address risks only after they have become problems.
2. **Fix on failure:** Detect and react to risks quickly, but only after they have occurred.
3. **Risk mitigation:** Plan ahead of time to provide resources to cover risks if they occur, but do nothing to eliminate them in the first place.
4. **Prevention:** Implement and execute a plan as part of the software project to identify risks and prevent them from becoming problems.
5. **Elimination of root causes:** Identify and eliminate factors that make it possible for risks to exist at all.

“Rapid Development: Taming Wild Software Schedules,” Steve McConnell, 1996

S3D

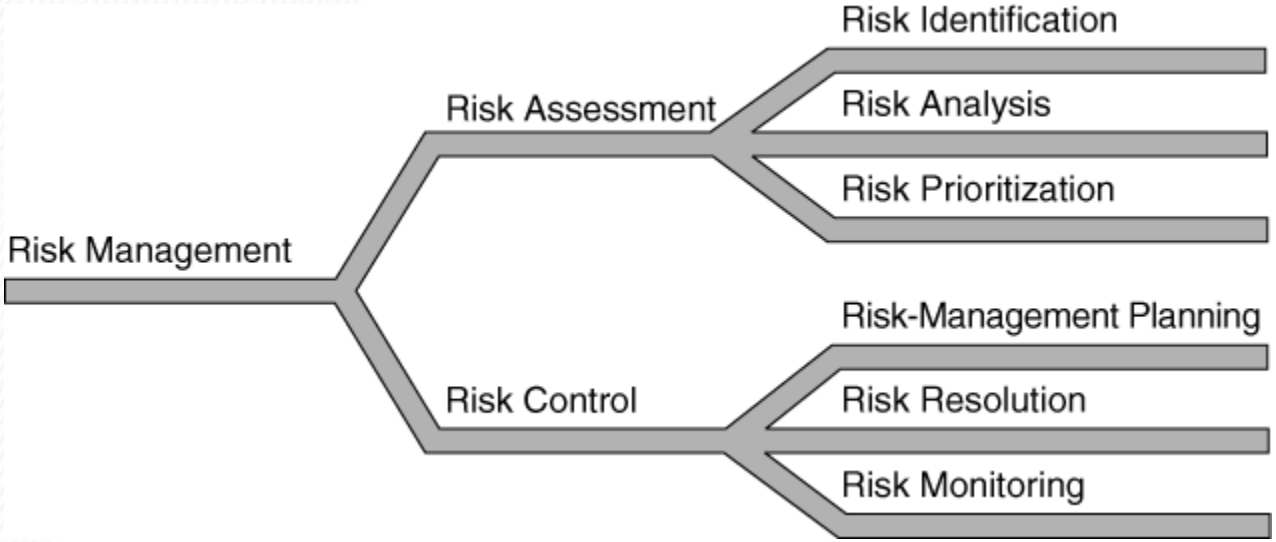
Levels of Risk Management

1. **Crisis management:** Fire fighting; address risks only after they have become problems.
2. **Fix on failure:** Detect and react to risks quickly, but only after they have occurred.
3. **Risk mitigation:** Plan ahead of time to provide resources to cover risks if they occur, but do nothing to eliminate them in the first place.
4. **Prevention:** Implement and execute a plan as part of the software project to identify risks and prevent them from becoming problems.
5. **Elimination of root causes:** Identify and eliminate factors that make it possible for risks to exist at all.

"Rapid Development: Taming Wild Software Schedules," Steve McConnell, 1996

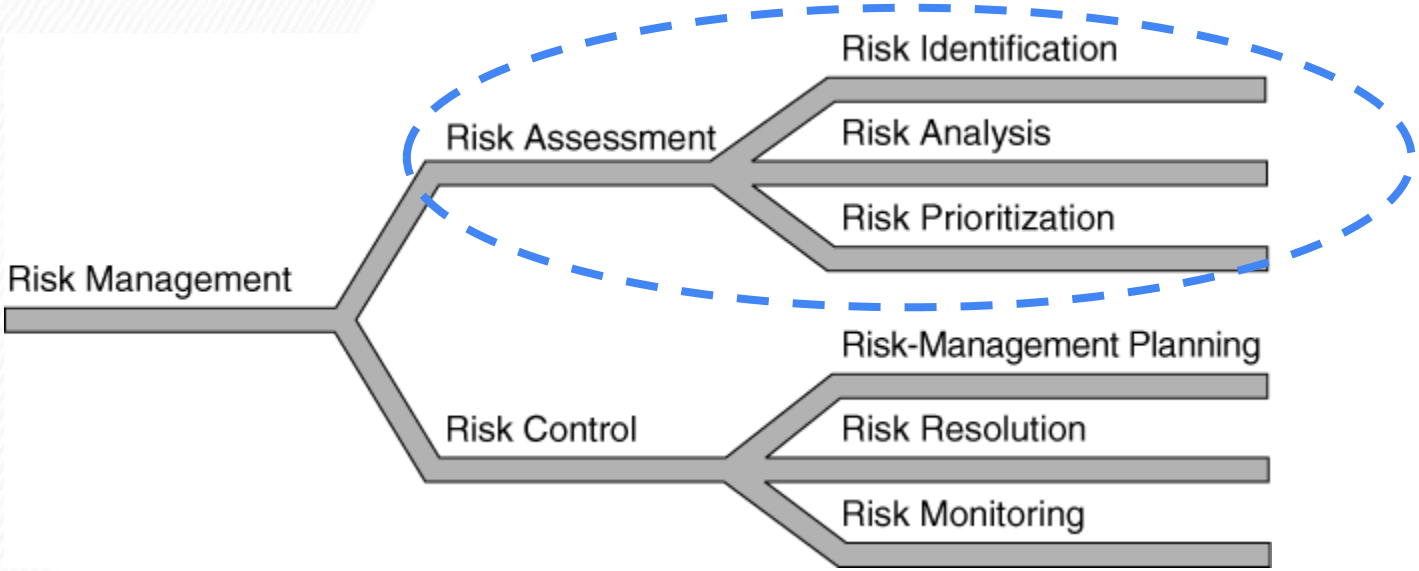
S3D

Risk Management Processes



"Rapid Development: Taming Wild Software Schedules," Steve McConnell, 1996

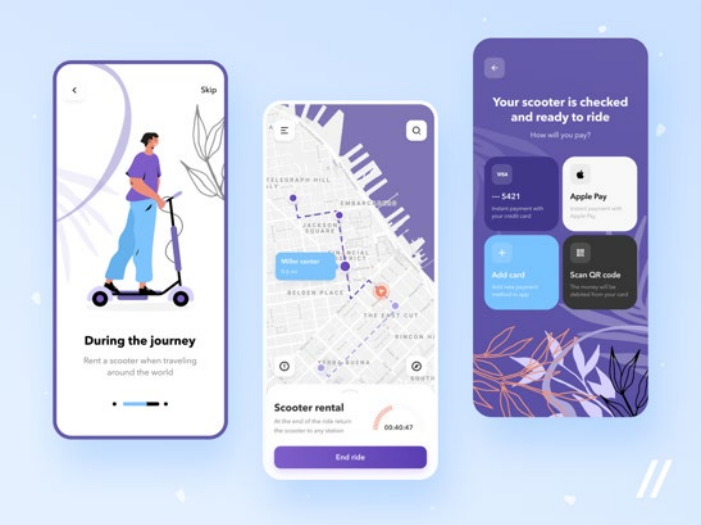
Risk Management Processes



“Rapid Development: Taming Wild Software Schedules,” Steve McConnell, 1996

Team Exercise: Risk Identification

- What risks exist for the development of the scooter app?



Risk assessment matrix

TABLE III. Risk assessment matrix

RISK ASSESSMENT MATRIX				
SEVERITY / PROBABILITY	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occasional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

- MIL-STD-882E

https://www.system-safety.org/Documents/MIL-STD-882E.pdf

Aviation failure impact categories

- **No effect** – failure has no impact on safety, aircraft operation, or crew workload
- **Minor** – failure is noticeable, causing passenger inconvenience or flight plan change
- **Major** – failure is significant, causing passenger discomfort and slight workload increase
- **Hazardous** – high workload, serious or fatal injuries
- **Catastrophic** – loss of critical function to safely fly and land

DO-178b, Software Considerations in Airborne Systems and Equipment Certification, RTCA, 1992

Risk Analysis

Risk	Probability (%)	Size of Loss (weeks)	Risk Exposure (weeks)
Overly optimistic schedule	50%	5	2.5
Additional features added by marketing (specific features unknown)	35%	8	2.8
Project approval takes longer than expected	25%	4	1.0
Management-level progress reporting takes more developer time than expected	10%	1	0.1
New programming tools do not produce the promised savings	30%	5	1.5
…	…	…	…
Total			12

“Rapid Development: Taming Wild Software Schedules,” Steve McConnell, 1996

S3D

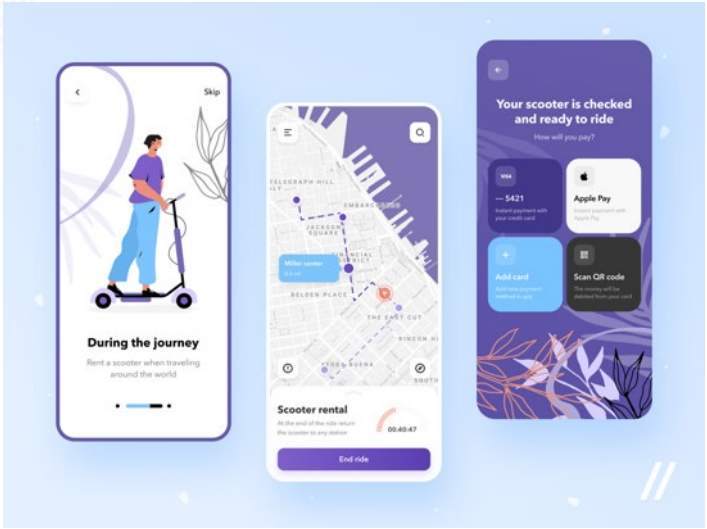
Risk Analysis Estimations

- Size of Loss
  - Use consensus-based approaches
- Probability
  - **This is much harder to estimate!**
  - Use a group-consensus approach (e.g., Planning Poker)
  - Use adjective calibration: Label each risk as “Very likely”, “Likely”, “Somewhat likely”, “Unlikely”, then convert labels into approximate quantitative values.

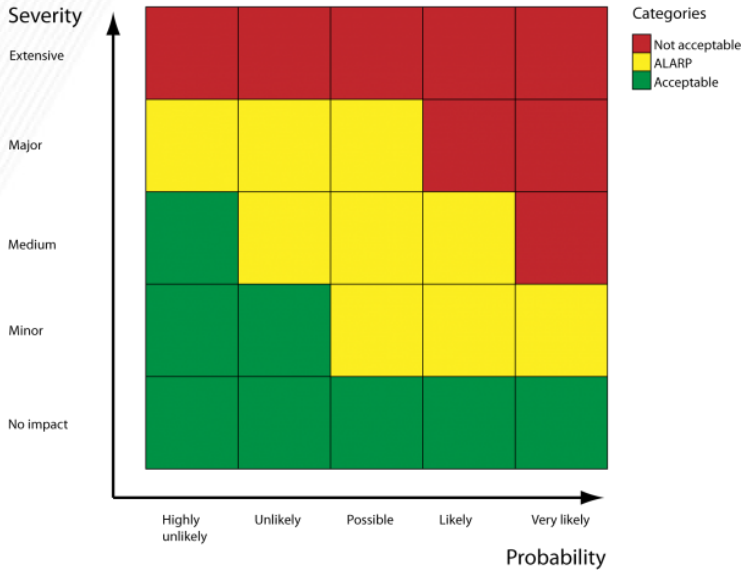
S3D

Exercise: Risk Analysis

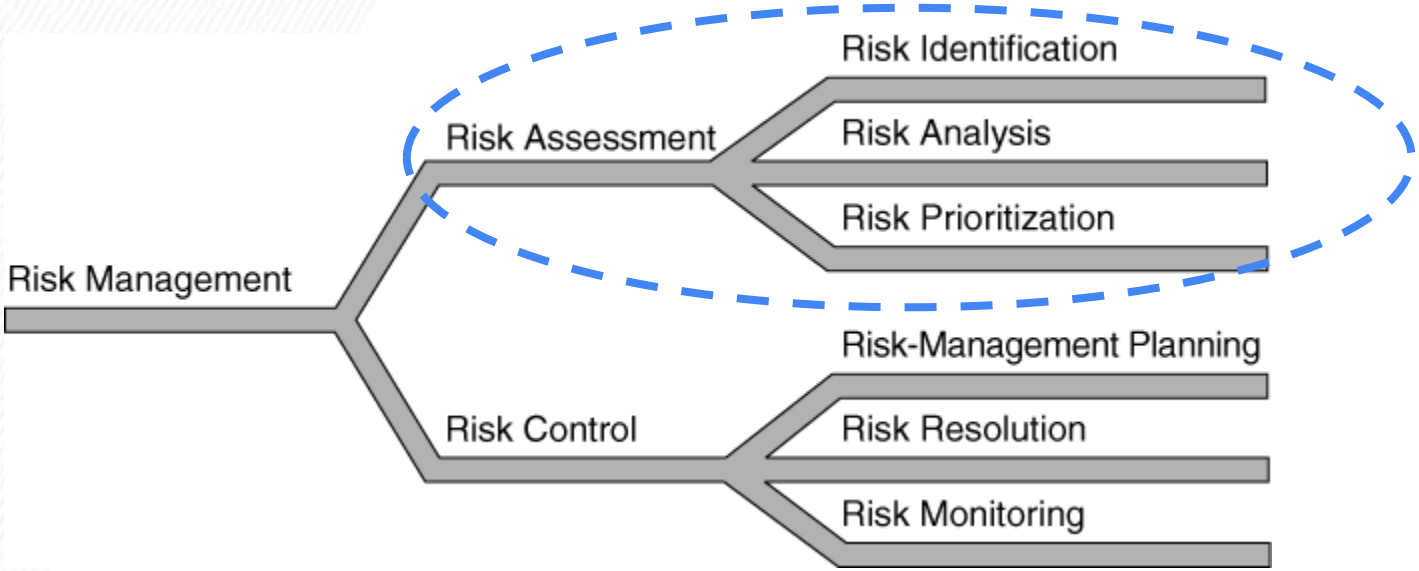
- What is the risk severity for the development of the scooter app?



Risk Prioritization
Focus on risks with the highest exposure

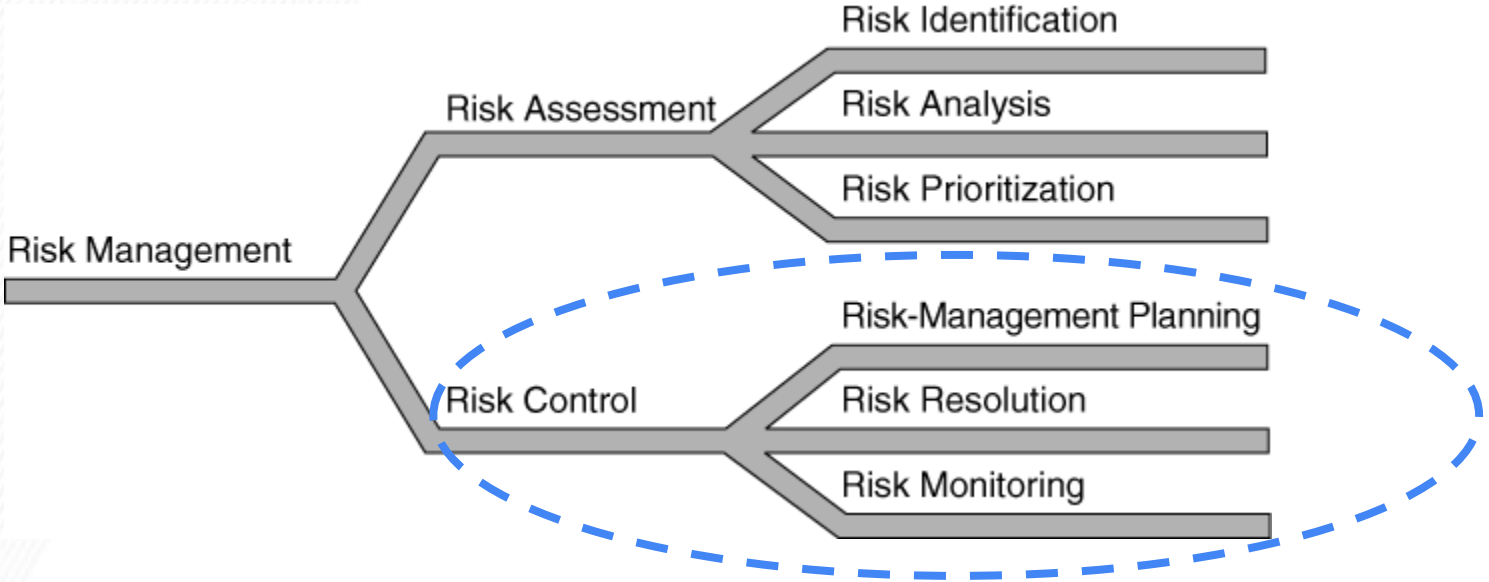


Risk Management Processes



"Rapid Development: Taming Wild Software Schedules," Steve McConnell, 1996

Risk Management Processes



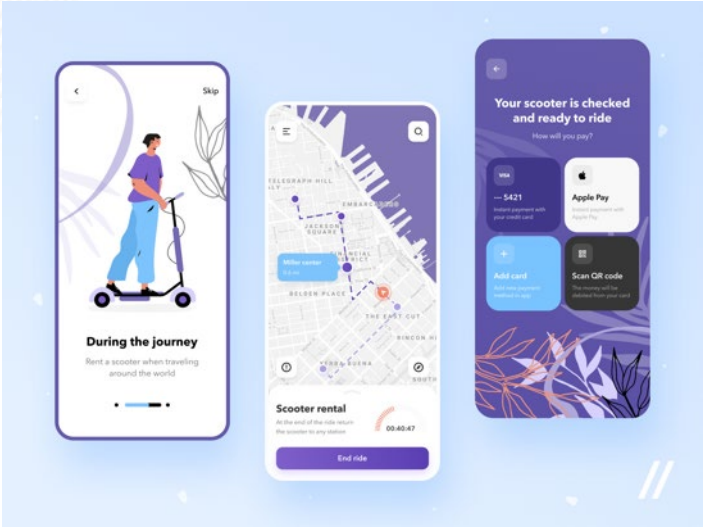
"Rapid Development: Taming Wild Software Schedules," Steve McConnell, 1996

Risk Control

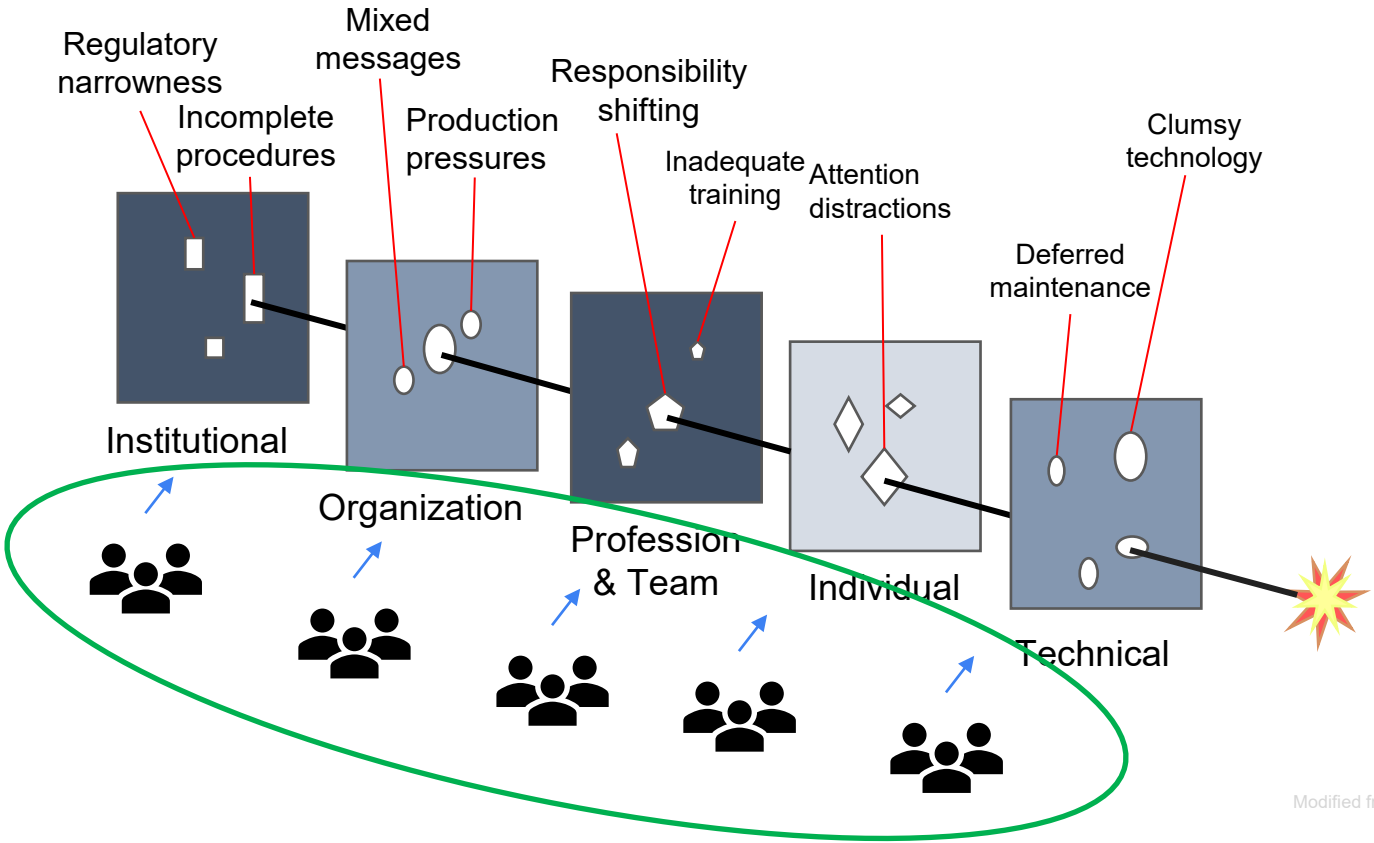
- What steps can be taken to avoid or mitigate the risk?
- Can you better understand and forecast the risk?
- Who will be responsible for monitoring and addressing the risk?
- Have risks evolved over time?
- Bake risks into your schedule
    - Don’t assume that nothing will go wrong between now and the end of the semester!

Discussion: Risk Elimination and Mitigation

- How can you eliminate/mitigate risk for the scooter app?

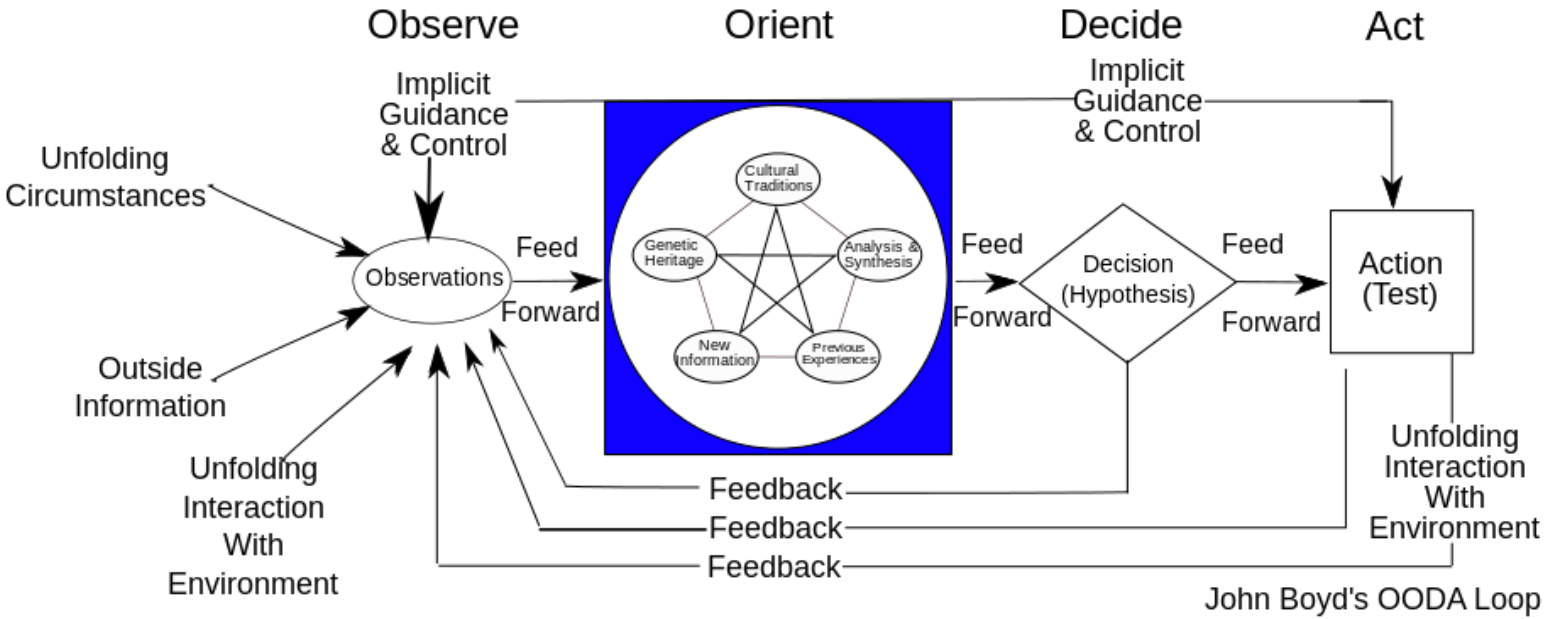


The Swiss cheese model



Modified from Reason, 1999, by R.I. Crook

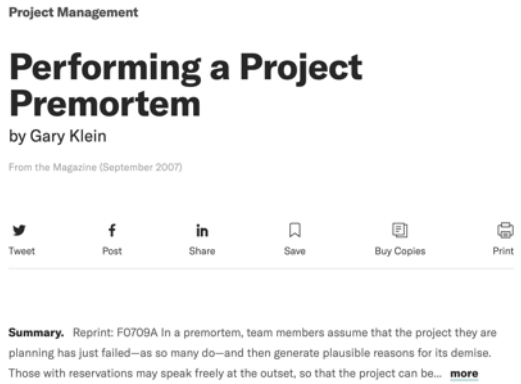
OODA Loop



By Patrick Edwin Moran - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=3904554

Pre-mortems

- "unlike a typical critiquing session, in which project team members are asked what *might* go wrong, the premortem operates on the assumption that the 'patient' has died, and so asks what *did* go wrong."



Project Management

Performing a Project Premortem

by Gary Klein

From the Magazine (September 2007)

Tweet Post Share Save Buy Copies Print

Summary. Reprint: F0709A In a premortem, team members assume that the project they are planning has just failed—as so many do—and then generate plausible reasons for its demise. Those with reservations may speak freely at the outset, so that the project can be... **more**

What are things that can go wrong?

Can we remove human error?

Generalization

- …in the words of psychologist Tom Stafford, we can’t find our typos because we’re engaging in a high-level task in writing. **Our brains generalize simple, component parts to focus on complex tasks**, so essentially we can’t catch the small details because we’re focused on a large task.

https://medium.com/swlh/why-we-miss-our-own-typos-96ab2f06afb7

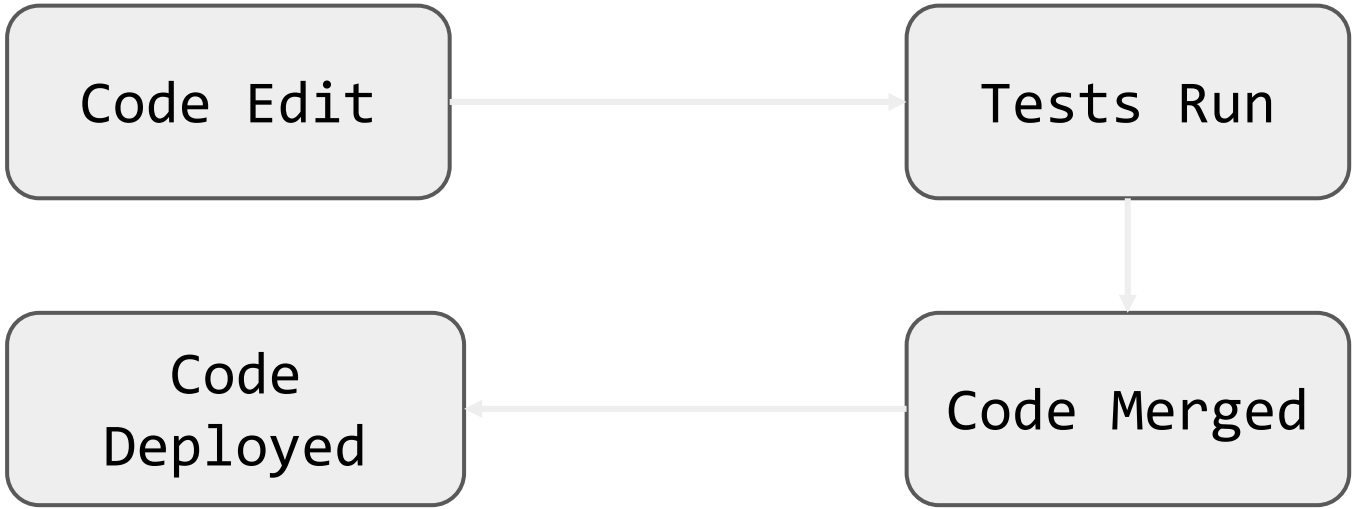
S3D

Can we ~~remove~~ catch human error?

Can we catch human error before we ship our code?
Can we automate tasks to prevent problems?

[State of Code Review 2017]

CI/CD Pipeline overview



Continuous Integration:

Catch mistakes before you push your code!

History of CI

(1999) Extreme Programming (XP) rule: “Integrate Often”

(2000) Martin Fowler posts “Continuous Integration” blog

(2001) First CI tool

(2005) Hudson/Jenkins

(2011) Travis CI

(2019) GitHub Actions

Sample CI Workflow

Create Pull Request

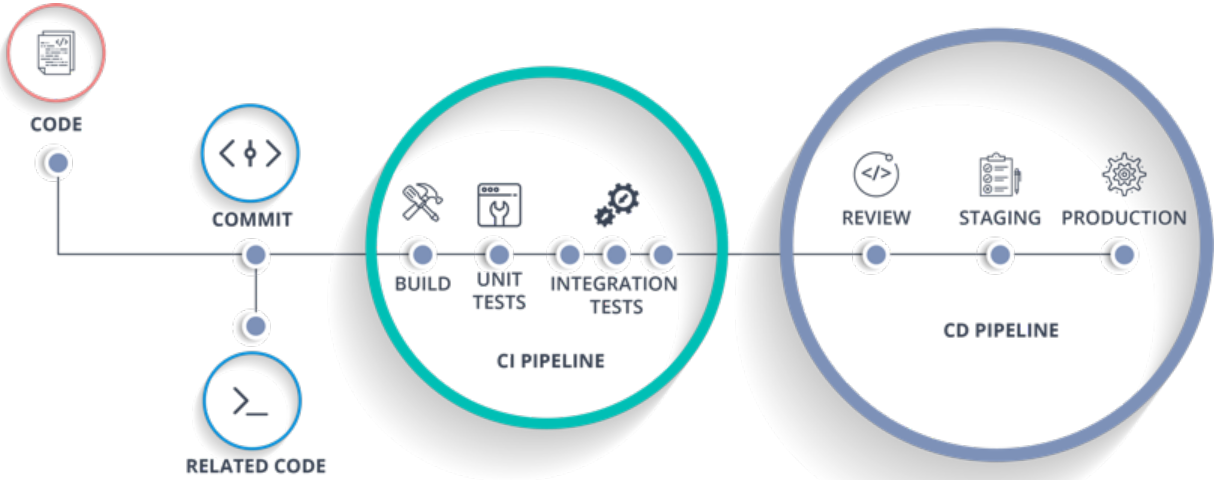
GitHub tells Travis CI build is mergeable

It builds and passes tests

Travis updates PR

PR is merged

Example CI/CD Pipeline



CI Research

Trade-Offs in Continuous Integration: Assurance, Security, and Flexibility

Michael Hilton
Oregon State University, USA
mhilton@cmu.edu

Nicholas Nelson
Oregon State University, USA
nelsonni@oregonstate.edu

Timothy Tunnell
University of Illinois, USA
tunnell2@illinois.edu

Darko Marinov
University of Illinois, USA
marinov@illinois.edu

Danny Dig
Oregon State University, USA
digd@oregonstate.edu

"523 complete responses, and a total of 691 survey responses from over 30 countries. Over 50% of our participants had over 10 years of software development experience, and over 80% had over 4 years of experience."

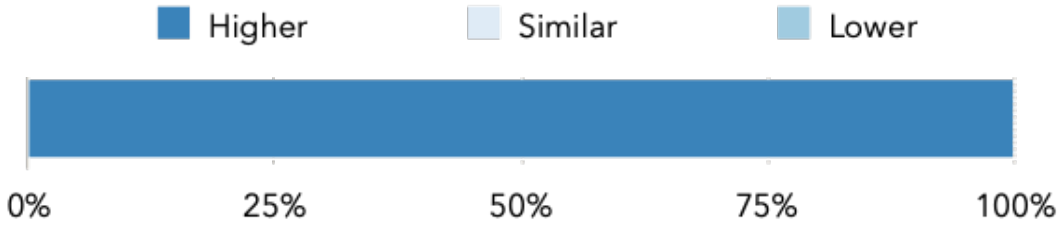
Developers say:

CI helps us catch bugs earlier
CI makes us less worried about breaking our builds
CI lets us spend less time debugging

“[CI] does have a pretty big impact on [catching bugs]. It allows us to find issues even before they get into our main repo, … rather than letting bugs go unnoticed, for months, and letting users catch them.”

Developers report:

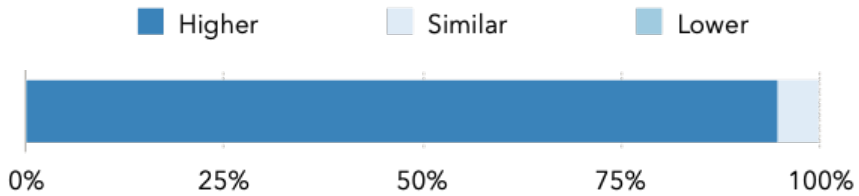
Do developers on projects with CI give (more/similar/less) value to automated tests?



Developers report:

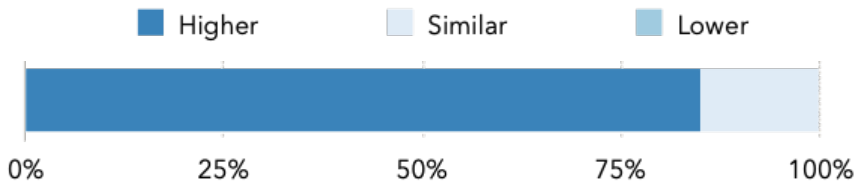
Do developers on projects with CI give (more/similar/less) value to automated tests?

Do projects with CI have (higher/similar/lower) test quality?



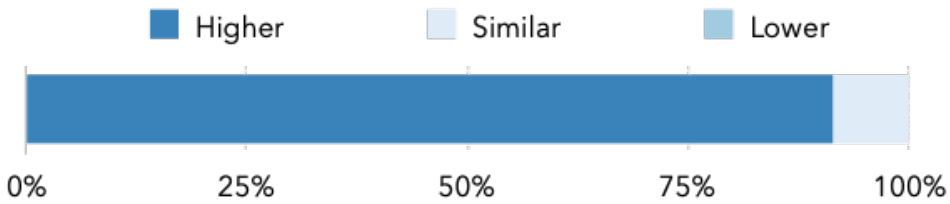
Developers report:

Do developers on projects with CI give (more/similar/less) value to automated tests?
Do projects with CI have (higher/similar/lower) test quality?
Do projects with CI have (higher/similar/lower) code quality?



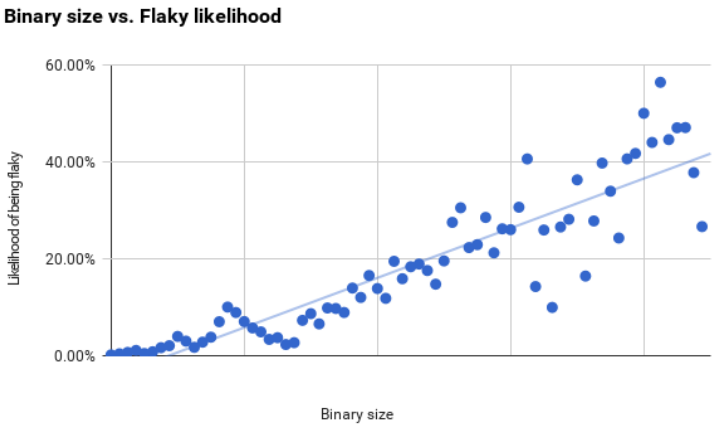
Developers report:

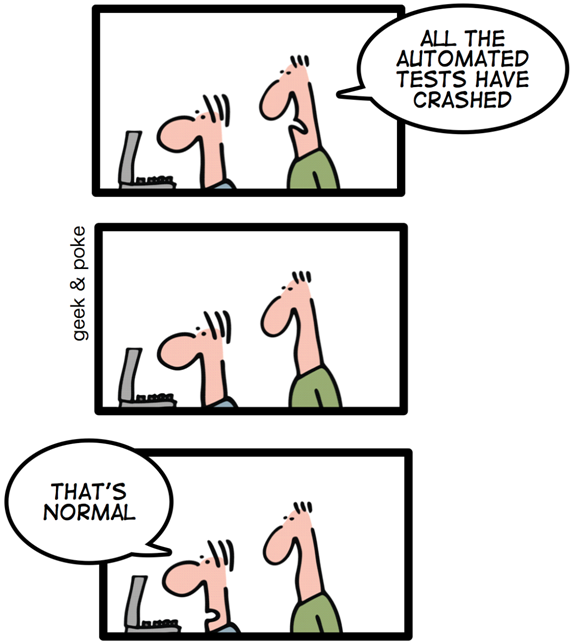
Do developers on projects with CI give (more/similar/less) value to automated tests?
Do projects with CI have (higher/similar/lower) test quality?
Do projects with CI have (higher/similar/lower) code quality?
Are developers on projects with CI (more/similar/less) productive?



Challenge: Flaky Tests

“Google has around 4.2 million tests that run on our continuous integration system. Of these, around 63 thousand have a flaky run over the course of a week”





https://testing.googleblog.com/2017/04/where-do-our-flaky-tests-come-from.html

Observation

CI helps us catch errors before others see them

Learning Goals

- Learn to discuss risk in a project
- Strategize about ways to mitigate risk
- Learn to get early feedback to reduce risk
- Find ways to catch our technical errors