Project Planning

17-313: Foundations of Software Engineering

https://cmu-313.github.io

Josh Sunshine and Michael Hilton

Spring 2026

Smoking Section

- Last full row

Administrivia

- Some teams might have to be re-balanced, we will be in touch
- Project 2A:
  - Team Planning and Process
  - Due**: Friday, Jan 30**
- Extra Credit: go out for an activity

Project 1: Retrospective

P1: Retrospective (1/3)

- You had to **fork, build, and test** an unseen codebase 
  - gained experience with lots of tools you will use throughout the semester
  - had to work some tricky setup issues
  - you should now have a good foundation for P2

- You had to **change the code** to remove a code smell 
  - some changes were harder to make than others
    - API changes → need to change all API uses
  - did you use AI to help you fix the smell? *(this was OK!)*

P1: Retrospective (2/3)

- Verifying those changes **is much harder!** ✅
  - did you break something? are you confident?
  - what assurances do you have?

- You used **code archaeology** to find out how to exercise them
  - added probes to find out what code is being executed (e.g., console.log)
  - used code search to work backwards from code to frontend HTML
  - some issues were very difficult to trigger! did you swap to another one?

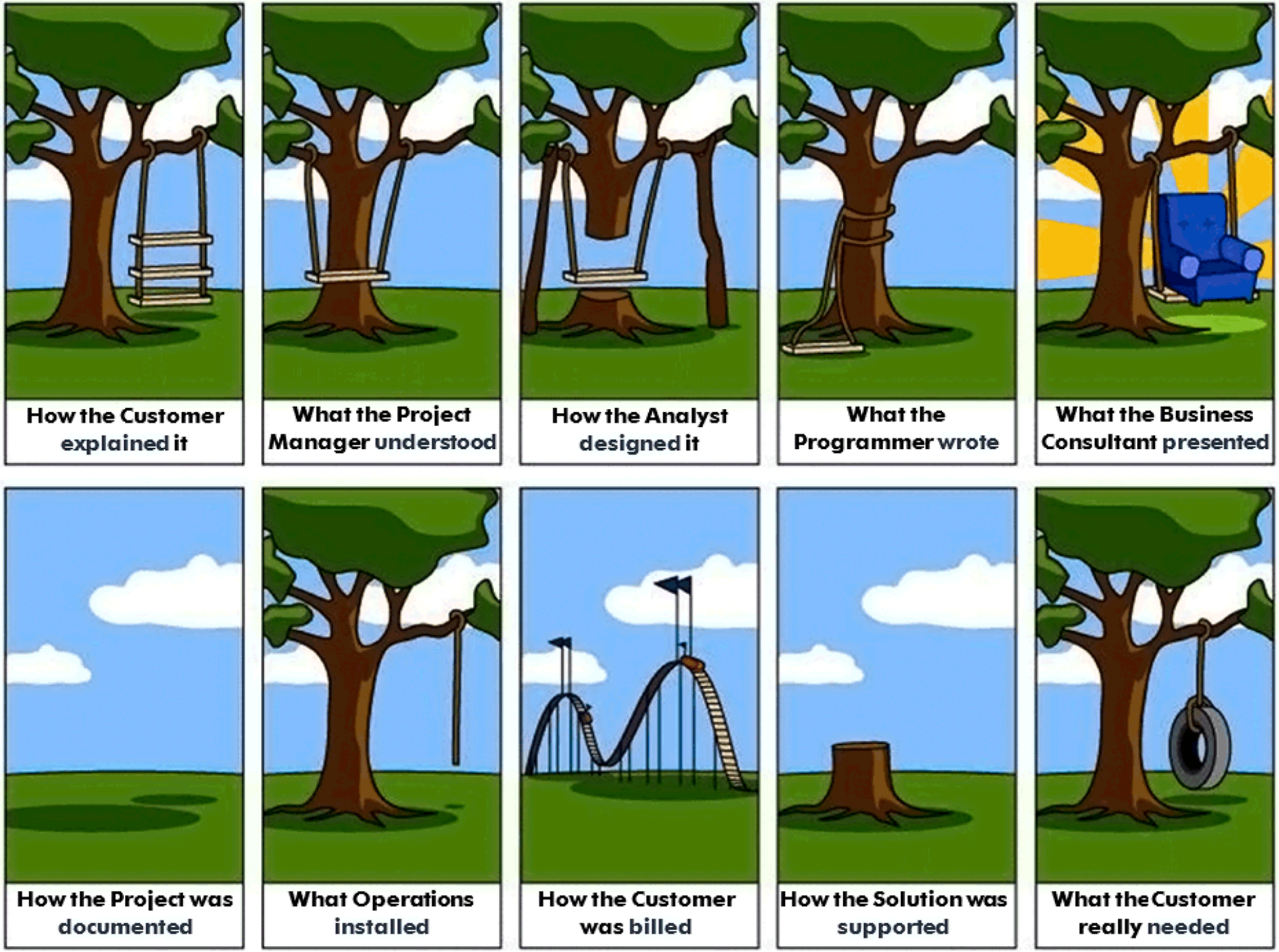
P1: Retrospective (3/3)

- You fixed the issue, but **did you really** fix the issue? 🤷
  - did you **introduce more problems** into the code? extraneous changes?
  - **should you** have fixed the issue? was it even an issue to begin with?
    - did you make changes to appease the linter?

- If I merge every PR, **would the codebase be better?** 🤔
  - how do I review these PRs?
  - how do I make PRs that are more likely to be accepted?
  - is there a better way to confidently make large changes?

Today's Learning Goals

- Recognize the importance of process
- Identify why software development has project characteristics
- Understand the elements of Scrum
- Create and evaluate user stories
- Use milestones for planning and progress measurement
- Understand the difficulty of measuring progress


How the Customer explained it
What the Project Manager understood
How the Analyst designed it
What the Programmer wrote
What the Business Consultant presented
How the Project was documented
What Operations installed
How the Customer was billed
How the Solution was supported
What the Customer really needed

Software Process

“The set of activities and associated results that produce a software product”

Sommerville, SE, ed. 8

What does this mean?
What else can we do apart from coding?
Processes are **key** concerns.

Software En… Principles, practices (technical and non-technical) for confidently building high-quality software.

How to develop software???

Discuss the software that needs to be written

Write some code

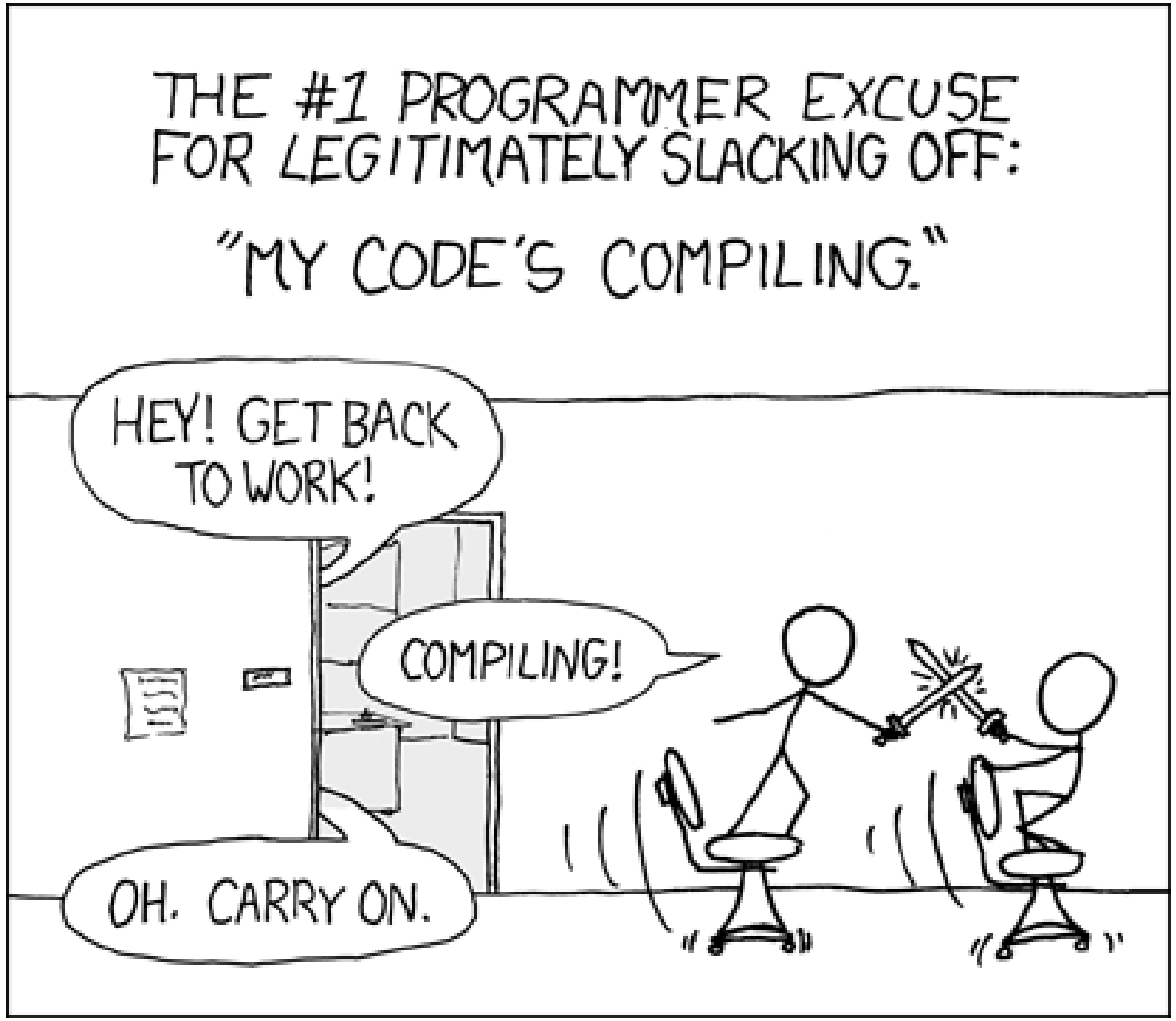
Test the code to identify the defects

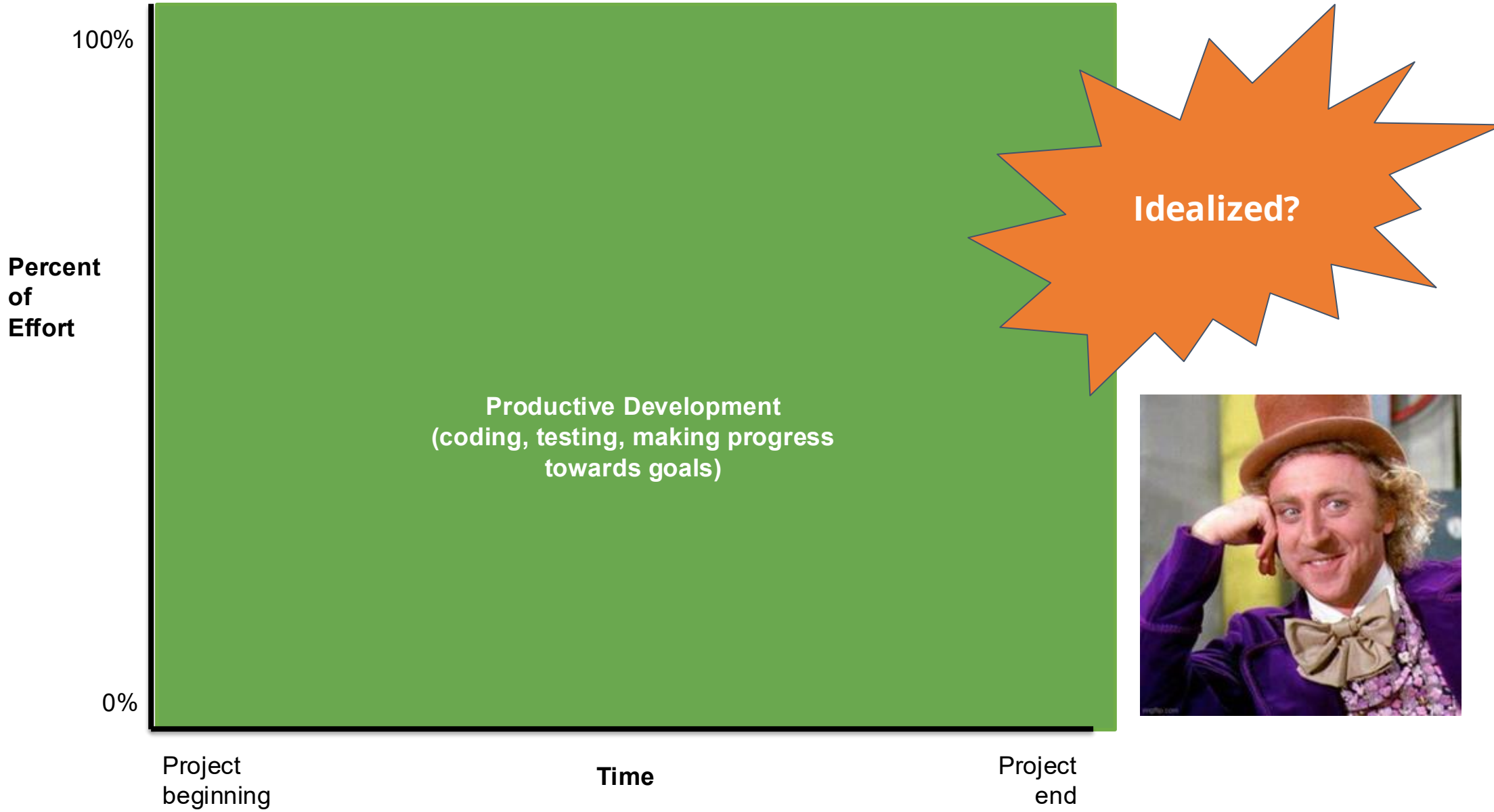
Debug to find causes of defects

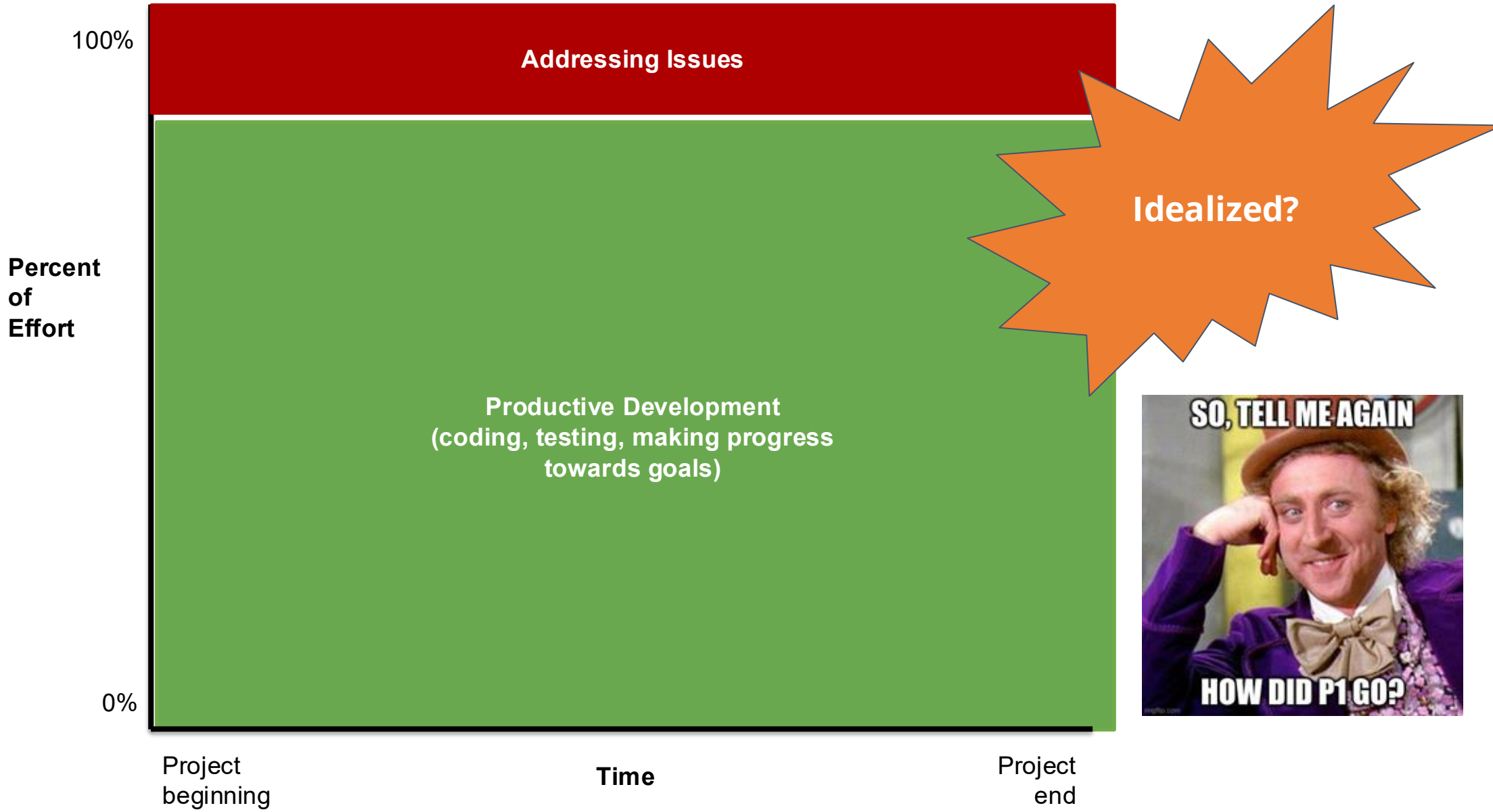
Fix the defects

What does a software engineer's day look like?

- How many hours do they spend in meetings, coding, testing, debugging, etc.?



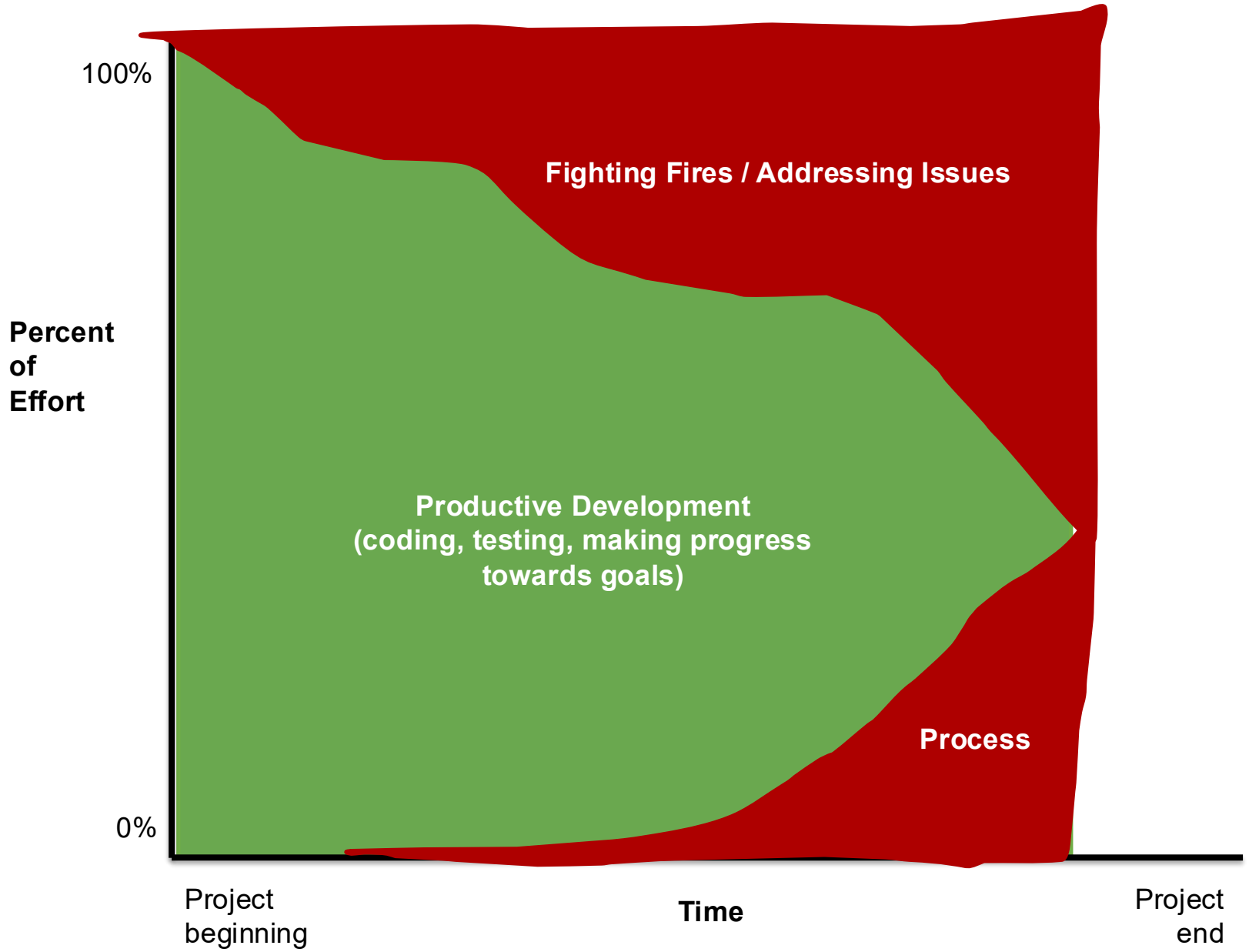

100%
Percent of Effort
0%
Productive Development
(coding, testing, making progress towards goals)
Idealized?
Project beginning
Time
Project end


100%
Addressing Issues
Idealized?
Percent of Effort
Productive Development
(coding, testing, making progress towards goals)
0%
Project beginning
Time
Project end

SO, TELL ME AGAIN
HOW DID P1 GO?

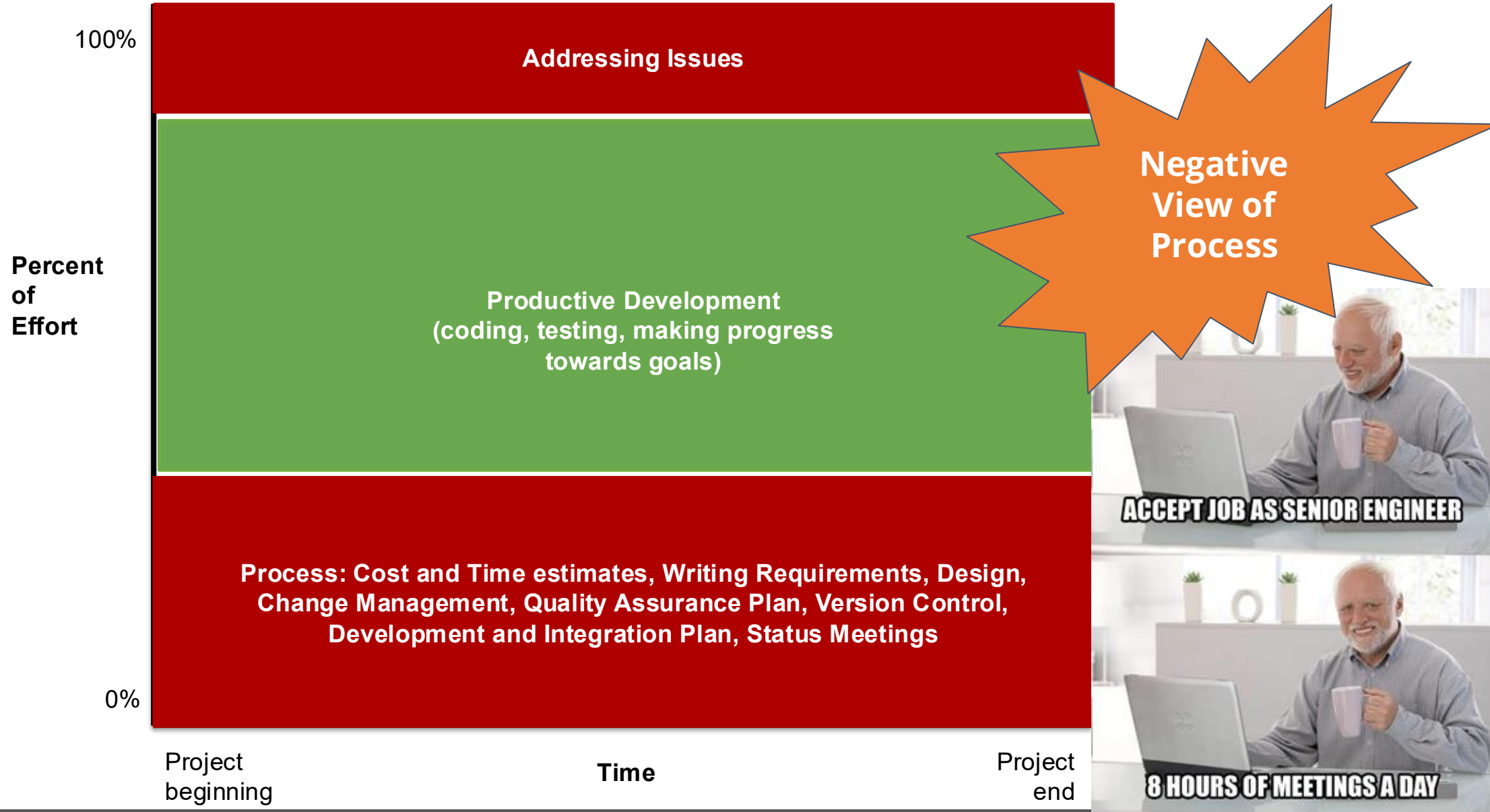
What happens when …

- **Uncontrolled Scope Creep:** Informal agreements balloon the project scope by 25-50%.
- **Late Failure:** Critical requirement and design flaws are discovered only during final testing.
- **Lost Defects:** Informal bug tracking (emails/hallway chats) leads to forgotten fixes.
- **Code Chaos:** Lack of version control leads to overwritten files and lost work.

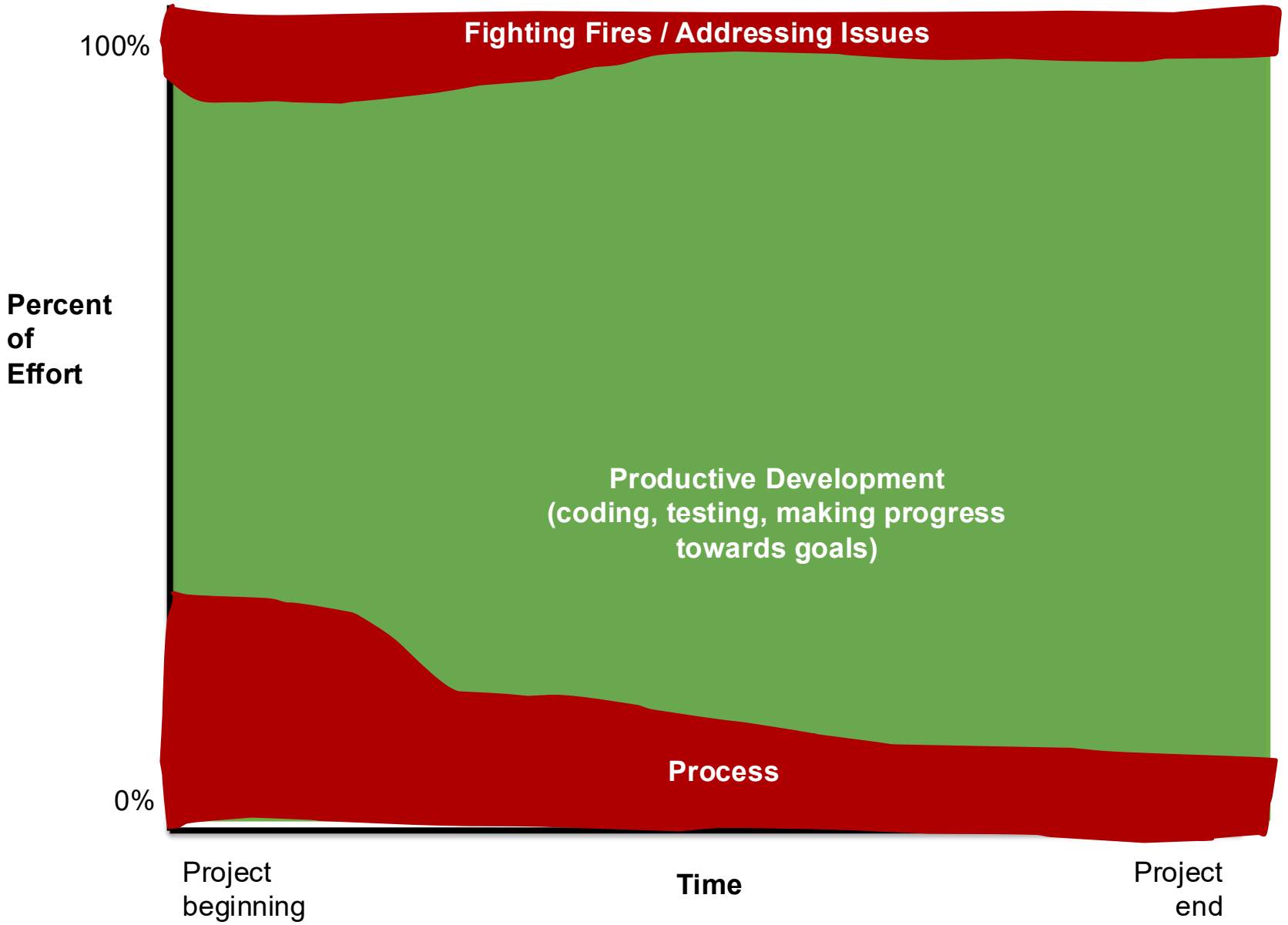

100%
Percent of Effort
0%
Fighting Fires / Addressing Issues
Productive Development
(coding, testing, making progress towards goals)
Process
Project beginning
Time
Project end

Let’s improve the reliability of this process

- Writing down all requirements
  - Review requirements
  - Require approval for all changes to requirements
- Use version control for all changes
  - Code Reviews
- Track all work items
  - Break down development into smaller tasks
  - Write down and monitor all reported bugs
  - Hold regular, frequent status meetings
- Plan and conduct quality assurance
- Employ a DevOps framework to push code between developers and operations

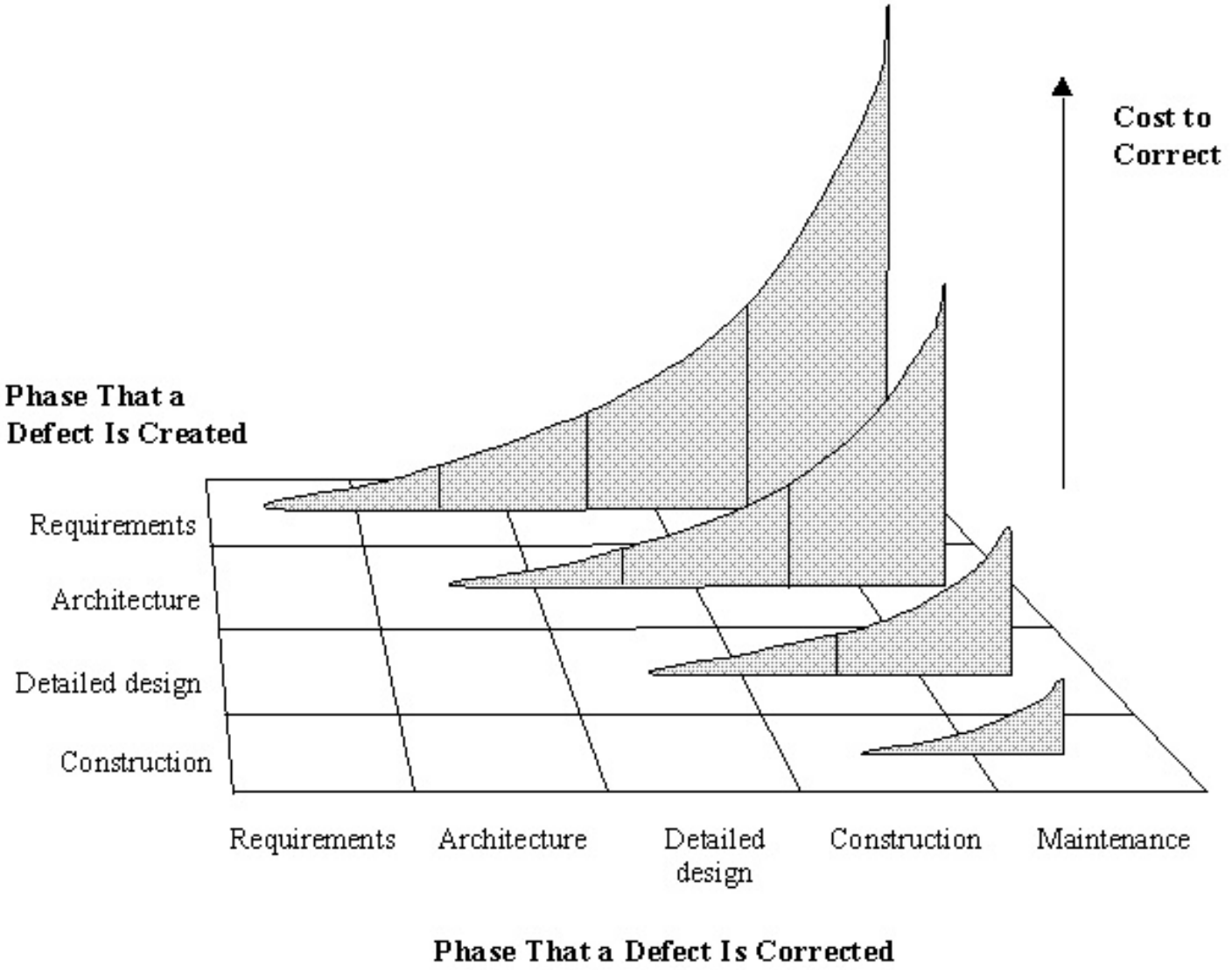

100%
Addressing Issues
Percent of Effort
Productive Development (coding, testing, making progress towards goals)
Process: Cost and Time estimates, Writing Requirements, Design, Change Management, Quality Assurance Plan, Version Control, Development and Integration Plan, Status Meetings
0%
Project beginning
Time
Project end

Negative View of Process
ACCEPT JOB AS SENIOR ENGINEER
8 HOURS OF MEETINGS A DAY
Posted in r/ProgrammerHumor
reddit

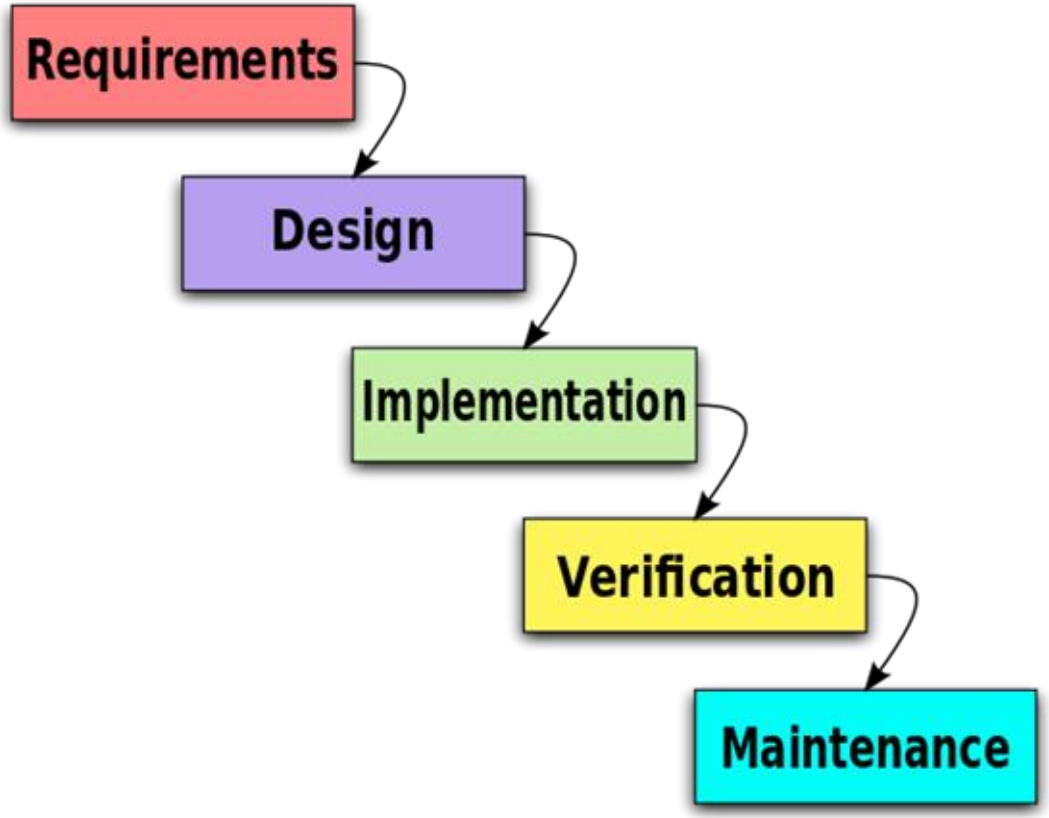


Hypothesis: Process increases flexibility and efficiency

Ideal Curve: Upfront investment for later greater returns

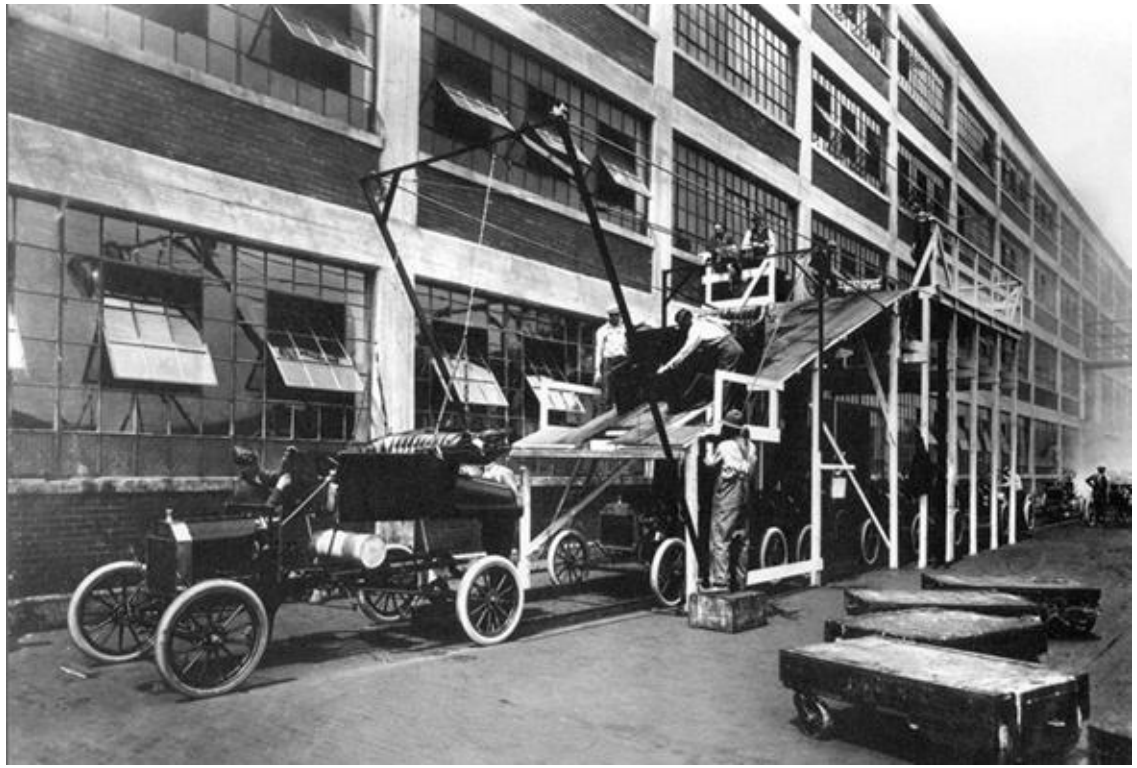


Waterfall model was the original software process

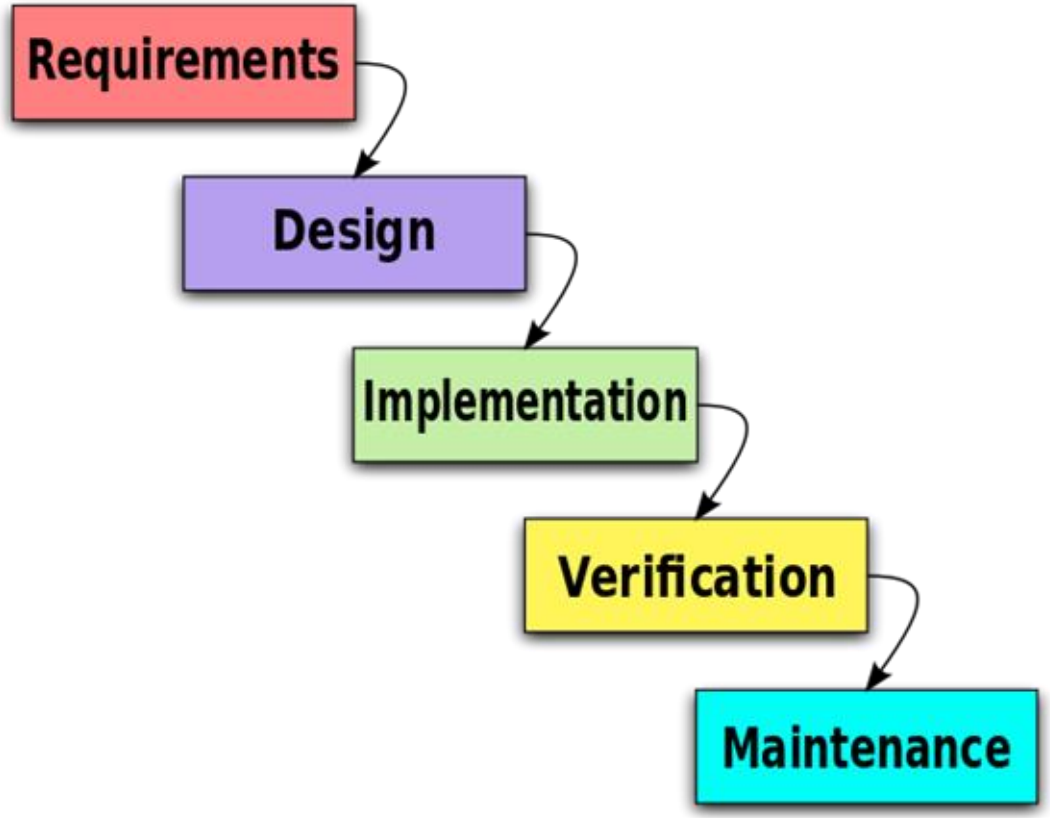


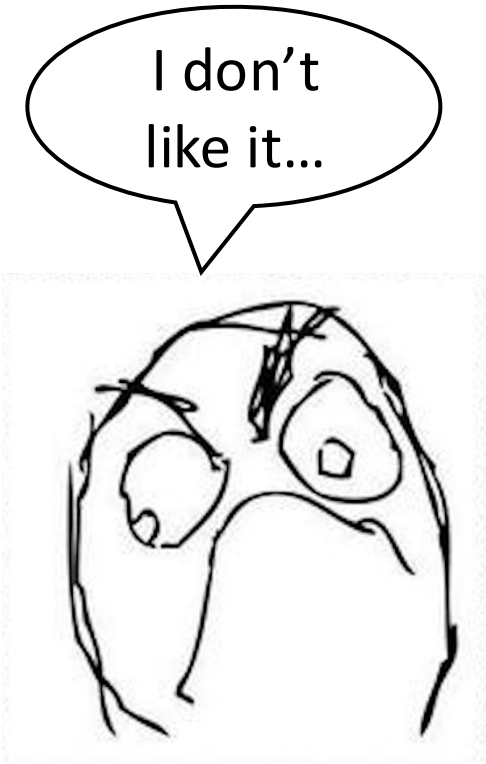
Waterfall diagram CC-BY 3.0 **Paulsmith99** at **en.wikipedia**

… akin to processes pioneered in mass manufacturing (e.g., by Ford)



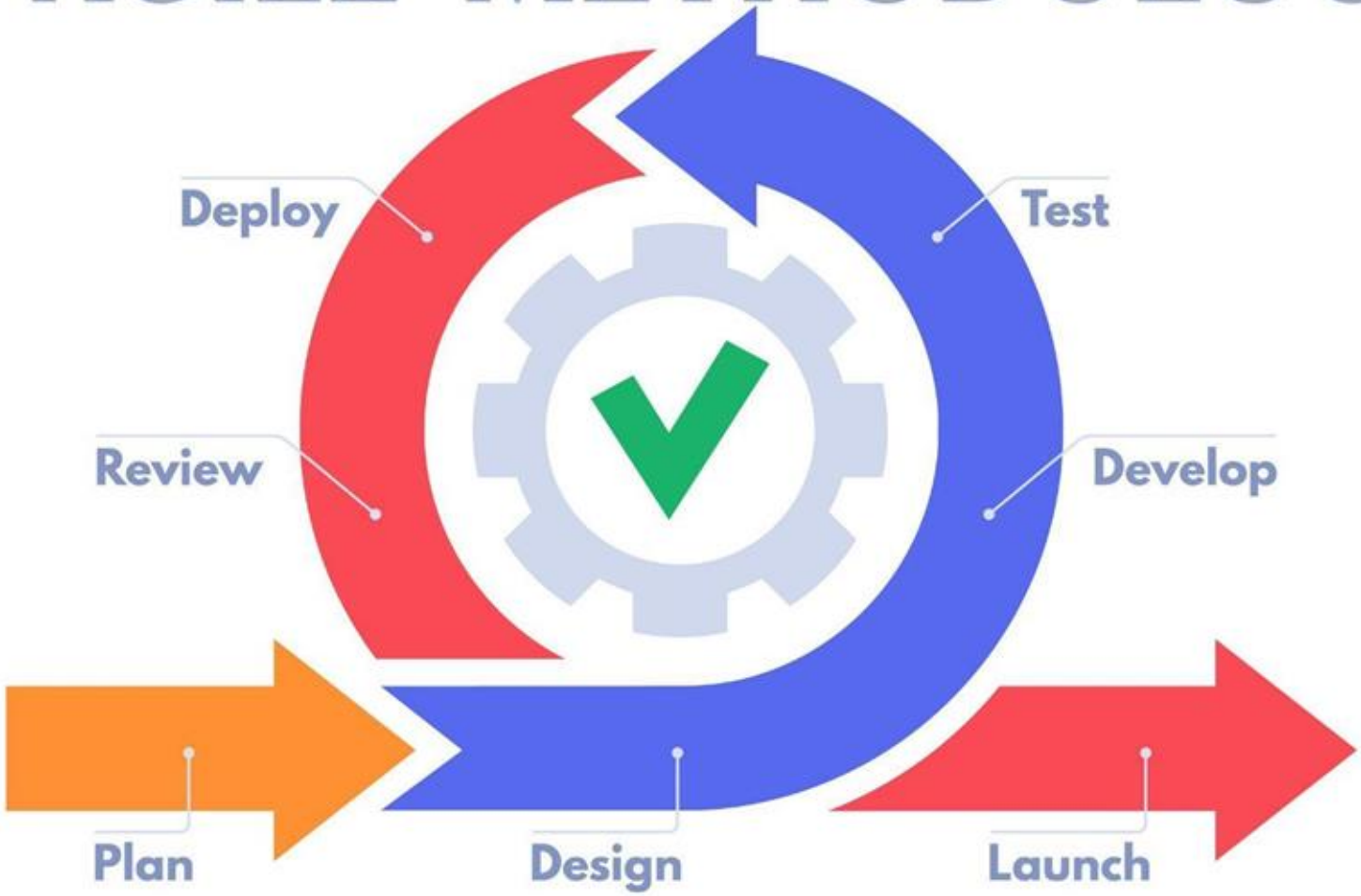
What could go wrong?





Waterfall diagram CC-BY 3.0 **Paulsmith99** at **en.wikipedia**

AGILE METHODOLOGY



Agile manifesto

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

https://agilemanifesto.org/

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

Agile manifesto

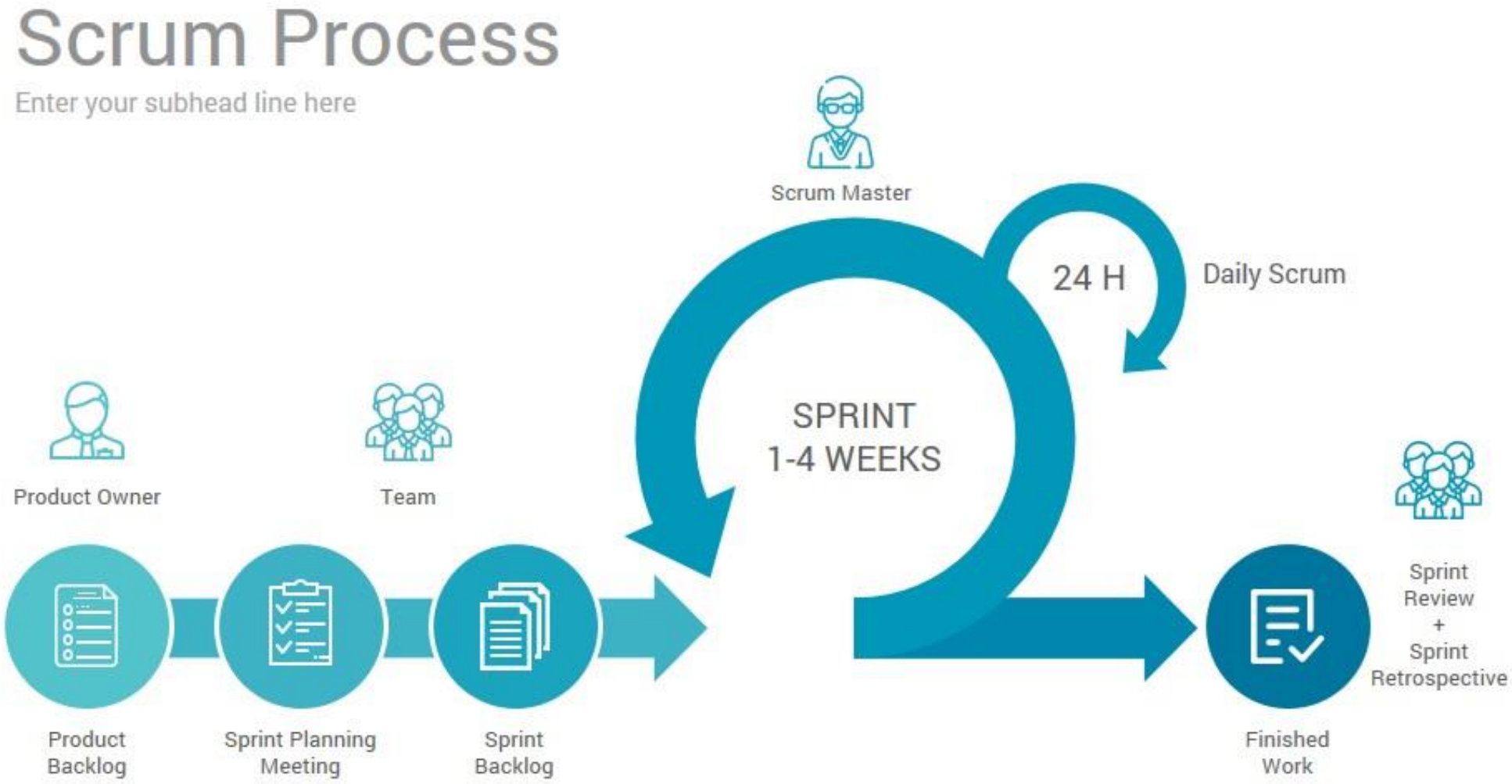
Twelve high-level principles, examples include:

- *“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”*
- *“Working software is the primary measure of progress”*
- *“Continuous attention to technical excellence and good design enhances agility.”*
- *“Simplicity—the art of maximizing the amount of work not done—is essential.”*

Scrum

(Only a brief intro)

Elements of Scrum



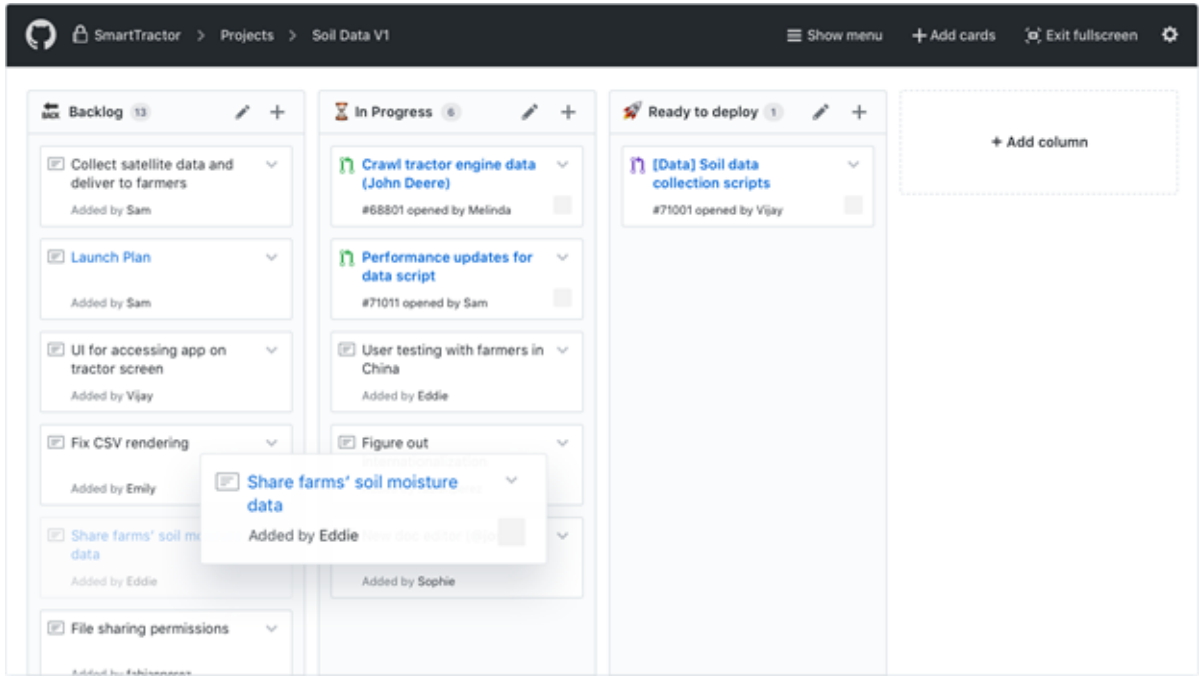
Backlogs

The **product backlog** is all the features for the product
The **sprint backlog** is all the features that will be worked on for that sprint. These should be broken down into discrete tasks:

- Fine-grained
- Estimated
- Assigned to individual team members
- Acceptance criteria should be defined

User Stories are often used

Kanban boards



Scrum Meetings

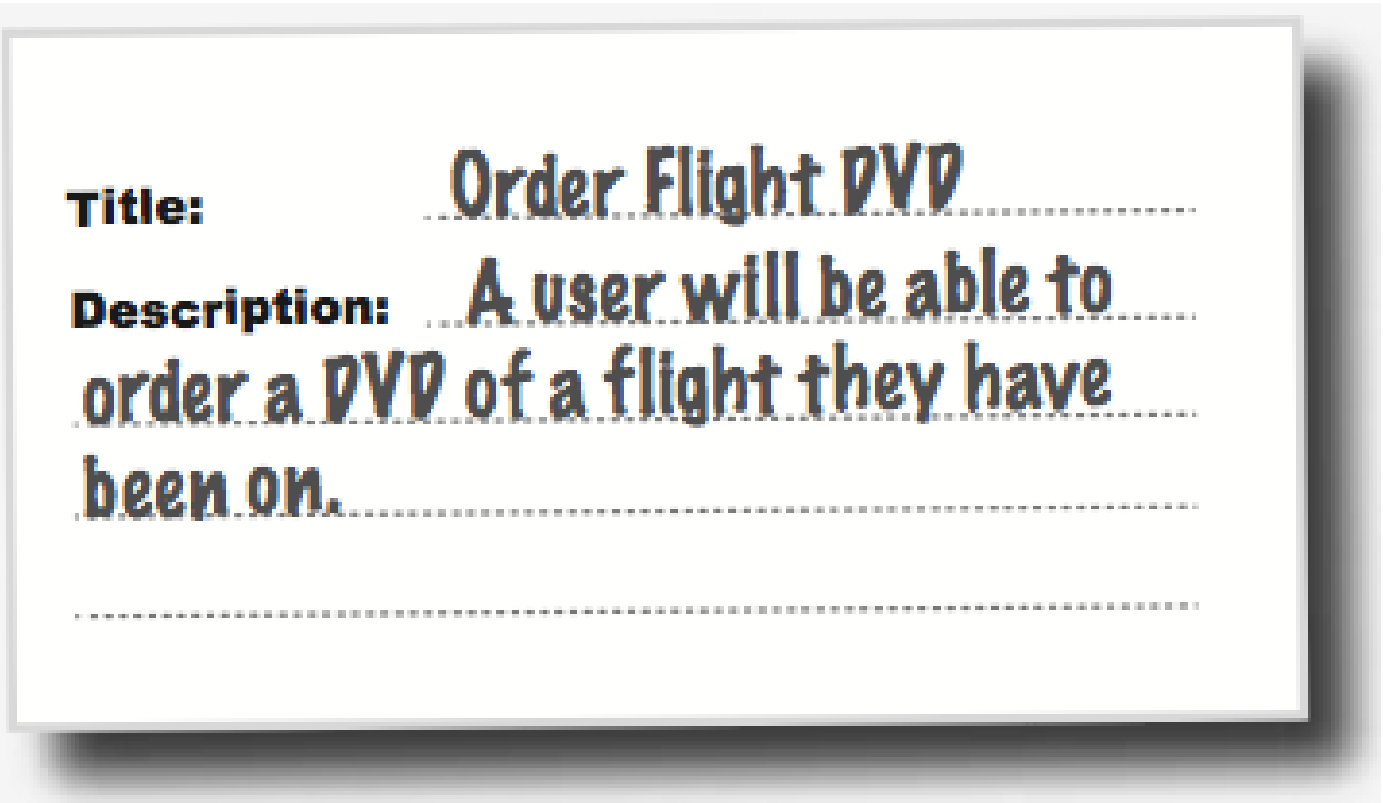
- Sprint Planning Meeting
  - Entire Team decides together what to tackle for that sprint
- Daily Scrum Meeting
  - Quick Meeting to touch base on :
    - What have I done? What am I doing next? What am I stuck on/need help?
- Sprint Retrospective
  - Review sprint process
- Sprint Review Meeting
  - Review Product

Standups

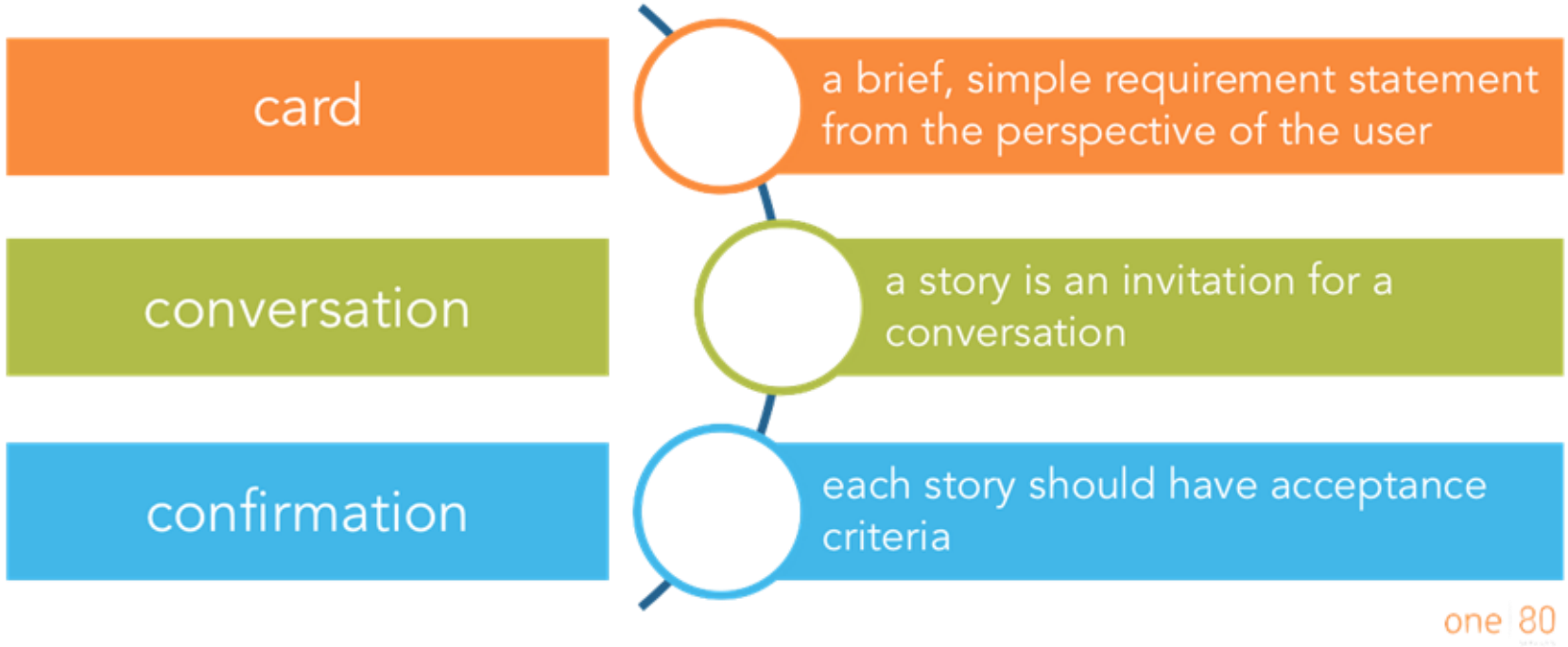
User stories

- Plan using units of customer-visible functionality

Example


Title:
Order Flight DVD
Description:
A user will be able to order a DVD of a flight they have been on.

User Stories



Source: http://one80services.com/user-stories/writing-good-user-stories-hint-its-not-about-writing/

User story cards (3”x5”)

“As a [role], I want [function], so that [value]”

Conversation

- Developers, product managers, etc.
- Is it clear to everyone?
- What must a developer do to implement this user story?

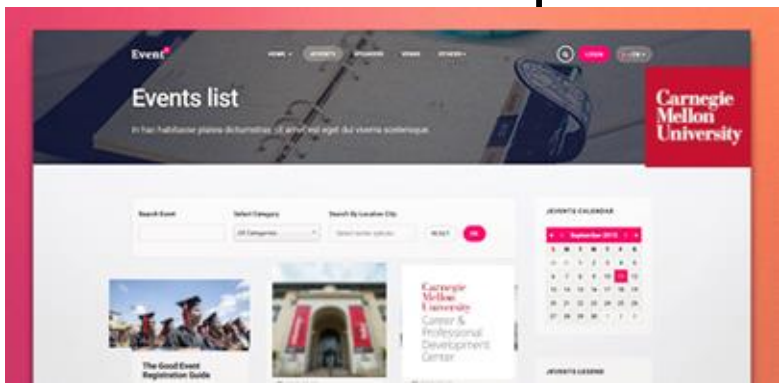
Acceptance criteria

- How can we tell that the user story has been achieved?
- It’s easy to tell when the developer finished the code.
- But, how do you tell that the customer is happy?

Example

"As a [role], I want [function], so that [value]"

The university is looking to enhance student and staff engagement by creating an online platform where all university-related events are easily accessible. The goal is to provide a user-friendly website that serves as a central hub for information on various activities, ranging from academic seminars to sports events and club meetings.



Participation activity: write a user story

- Project to consider: university event website introduced on previous slide
- Write one story on the paper we give you
- Keep it and we'll revisit it later in the lecture

How to evaluate user story?

Source: http://one80services.com/user-stories/writing-good-user-stories-hint-its-not-about-writing/

one 80

Independent

- Schedule in any order.
- Not always possible

Counterexample

As a student, **I want to** receive notifications for events that are about to start, for those I have shown interest in, **so I** don't miss them.

Acceptance Criteria:

- An option is provided to 'Set a Reminder' for each event.
- Notifications are sent to users who have opted for reminders, shortly before the event starts.

Negotiable

- Details to be negotiated during development
- Good Story captures the essence, not the details

Counterexample

As a student, **I want to** view the upcoming events at the university, **so I** can decide which ones to attend.

Acceptance Criteria:

- Add an interactive grid layout of upcoming events at the top of the homepage.
- Each event card in the grid is visible for a 2 seconds before automatically rotating to display the next set of events.
- Each card in the grid includes the event's name, type (e.g., seminar, sports game), duration, a brief description, and scheduled times.
- This grid of events is displayed under a prominent H1 heading that reads “Discover What’s Happening on Campus!”

Valuable

- This story needs to have value to someone (hopefully the customer)
- Easy to forget *why* you are doing what you are doing

Counterexample

As the Events Coordinator, **I want** a database to store details of students and staff interested in university events.

Acceptance Criteria:

- A database is constructed to manage user information.
- The database stores details such as name, email, phone number, favorite event types, date of birth, and history of event attendance or registrations.

Estimable

- Helps keep the size small
- It should provide enough details to estimate the amount of effort needed
- More on estimates later…

Counterexample

As an undergraduate student, **I want to** be able to filter university events, **so I** can choose the ones that align with my interests.

Acceptance Criteria:

- Filters are added to the event listings on the website.

Small

- Fit on 3x5 card
- At most two person-weeks of work (one sprint)
- Too big == unable to estimate

Counterexample

As a student, **I want to** easily find information about upcoming events, **so I** can participate in activities that interest me.

Acceptance criteria:

- A homepage is created displaying the university's name, motto, location, email, and contact information.
- The homepage features a calendar of upcoming university events.
- The event calendar includes details such as the event title, type (e.g., seminar, sports game, club meeting), a brief description, location, date, and time.
- Users can filter the event list by event type, date, and hosting department or club.
- The admin can update the event calendar as new events are planned or existing events are modified.

Testable

- Ensures understanding of task
- We know when we can mark task “Done”
- Unable to test == do not understand

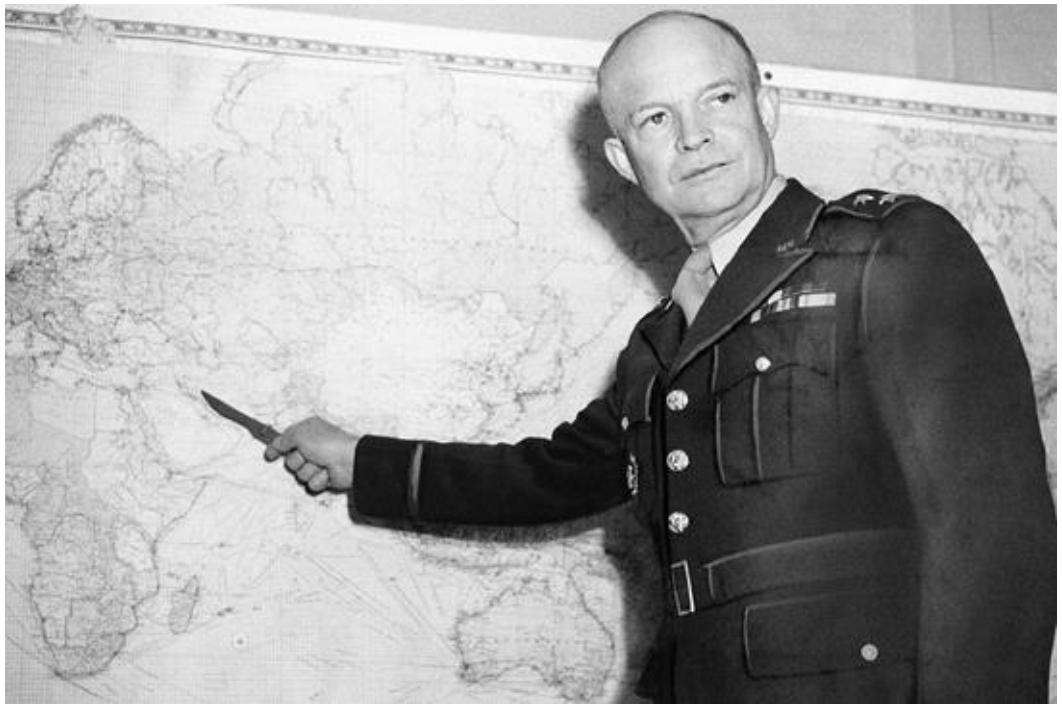
Counterexample

As a student, **I want to** easily view promotional videos or trailers of university events, **so I** can decide which events to attend.

Acceptance Criteria:

- Promotional videos can be embedded on each event detail page.
- Videos are of high quality.
- The embedded video is well-integrated into the page design.
- The video size is large enough to ensure clarity.
- The video controls are user-friendly.

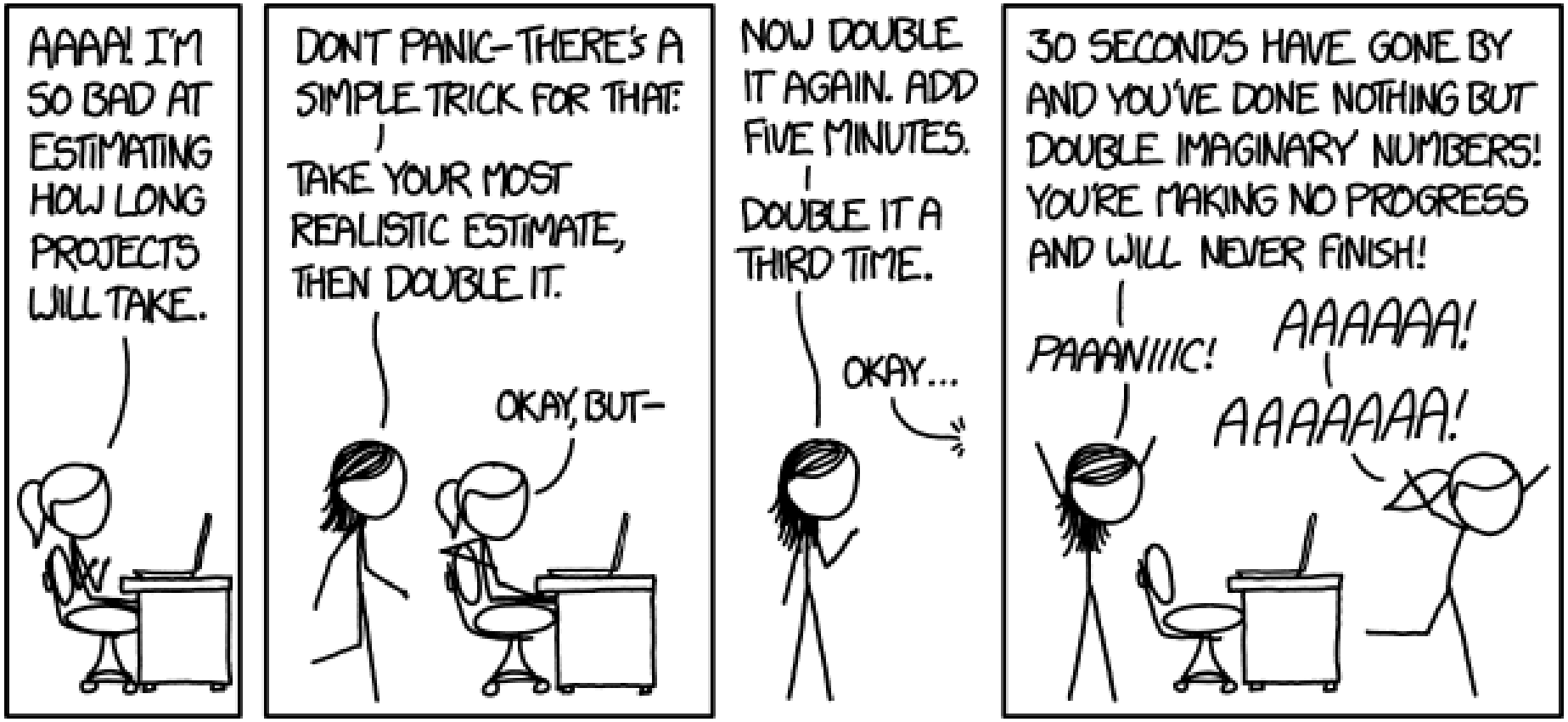
Activity: Evaluate using INVEST



“Plans are nothing, planning is everything”

-Dwight D. Eisenhower

Time estimation



https://xkcd.com/1658/

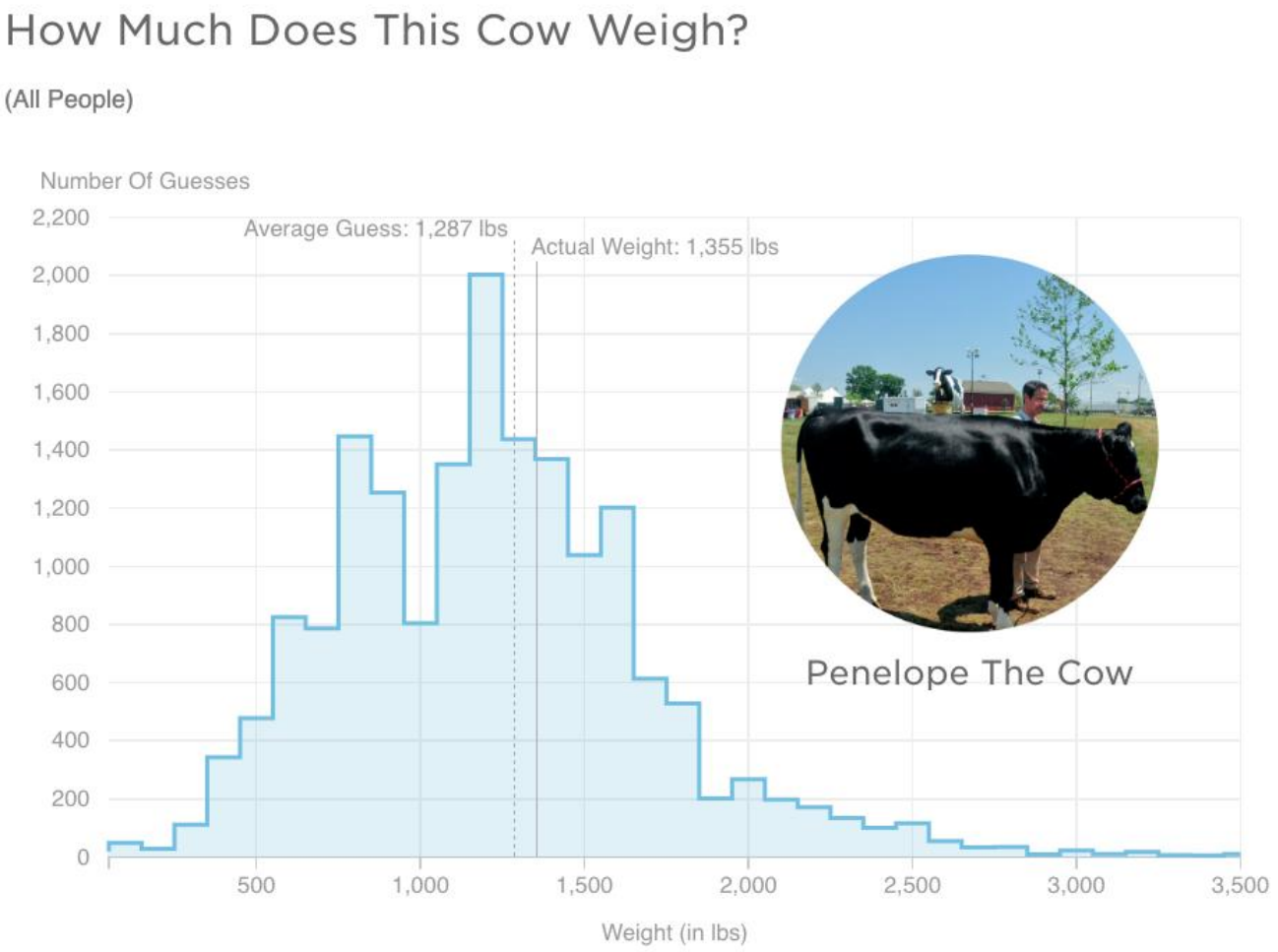
Time estimation

https://xkcd.com/612

Improving Time Estimates

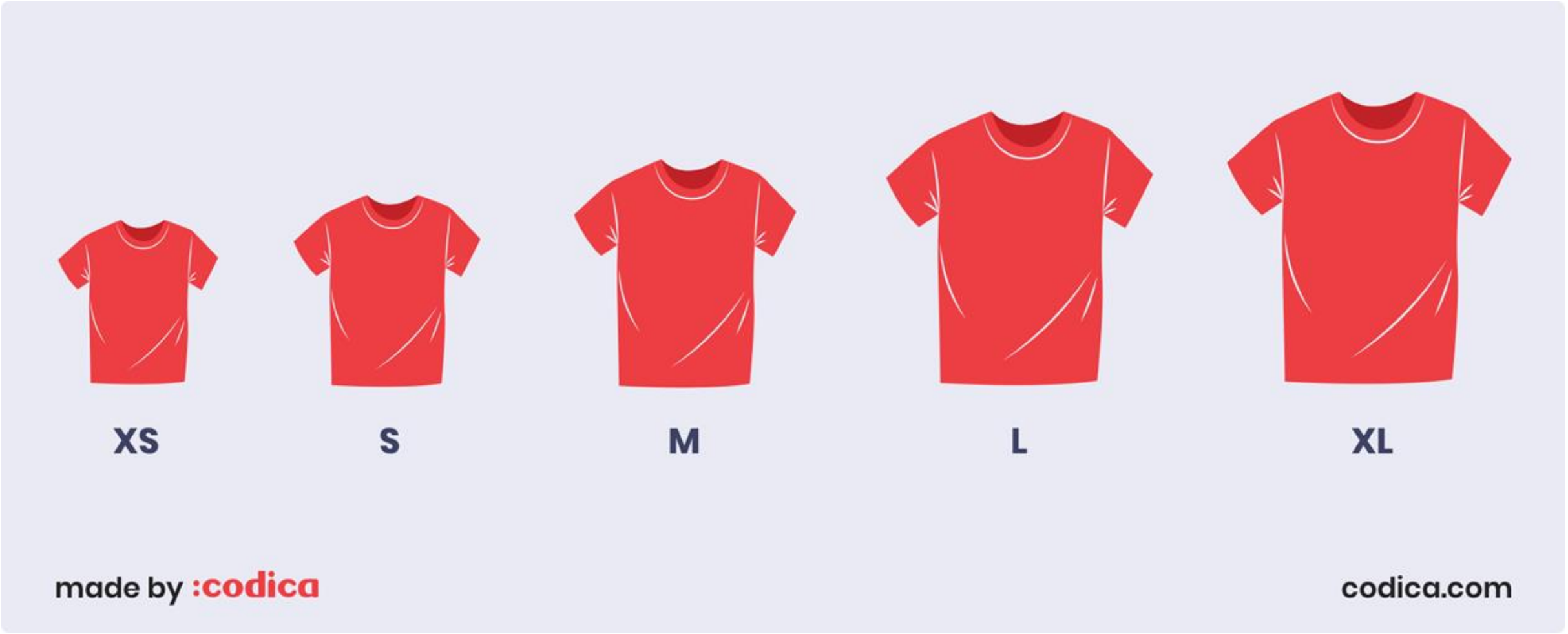
- Prevent conformity bias
- Do you have a comparable experience to base an estimate on?
- How much design do you need for each task?
- Break down the task into smaller tasks and estimate them.

Wisdom of the Crowd

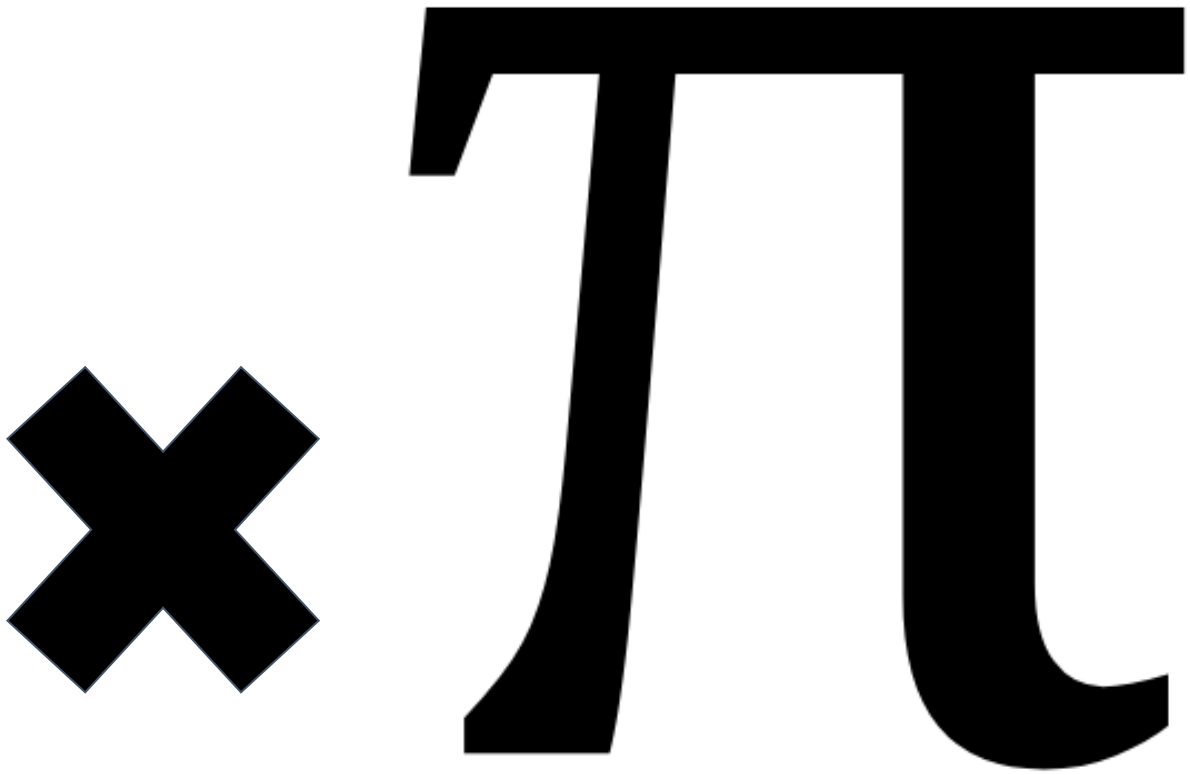


Source: The Internet.

Credit: Quoctrung Bui/NPR

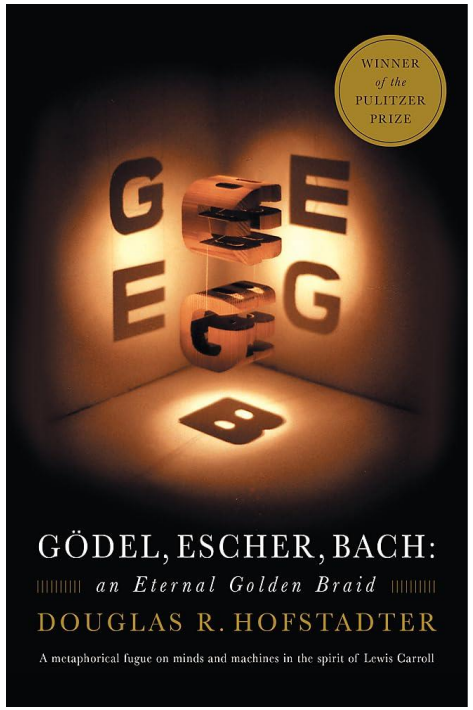

XS
S
M
L
XL
made by :codica
codica.com


1
2
3
5
8
13
21



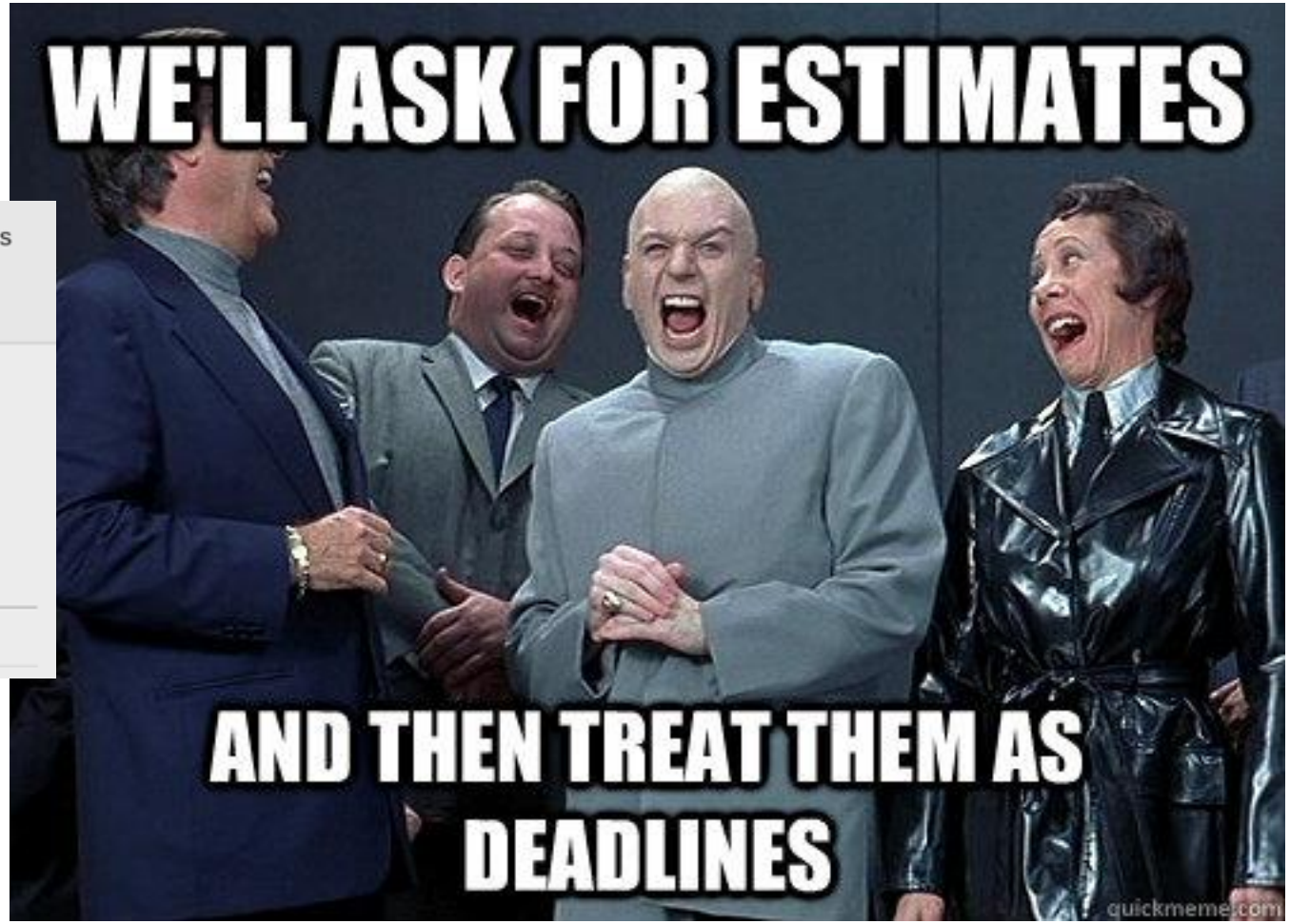
Hofstadter's Law

"It always takes longer than you expect, even when you take into account Hofstadter's Law"



Is Estimation Evil?

Ron Jeffries's essay [Estimation is Evil]



Milestones and deliverables make progress *observable*

Milestone: clear end point of a (sub)tasks

- For project manager
- Reports, prototypes, completed subprojects
- "80% done" is not a suitable milestone

Deliverable: Result for customer

- Similar to milestones, but for customers
- Reports, prototypes, completed subsystems

What you need to know

- Recognize the importance of having a software process
- Main ideas of Agile/Scrum
- Understand backlogs and user stories
- Understand the difficulty of estimating tasks and progress
- We use milestones for planning and progress measurement