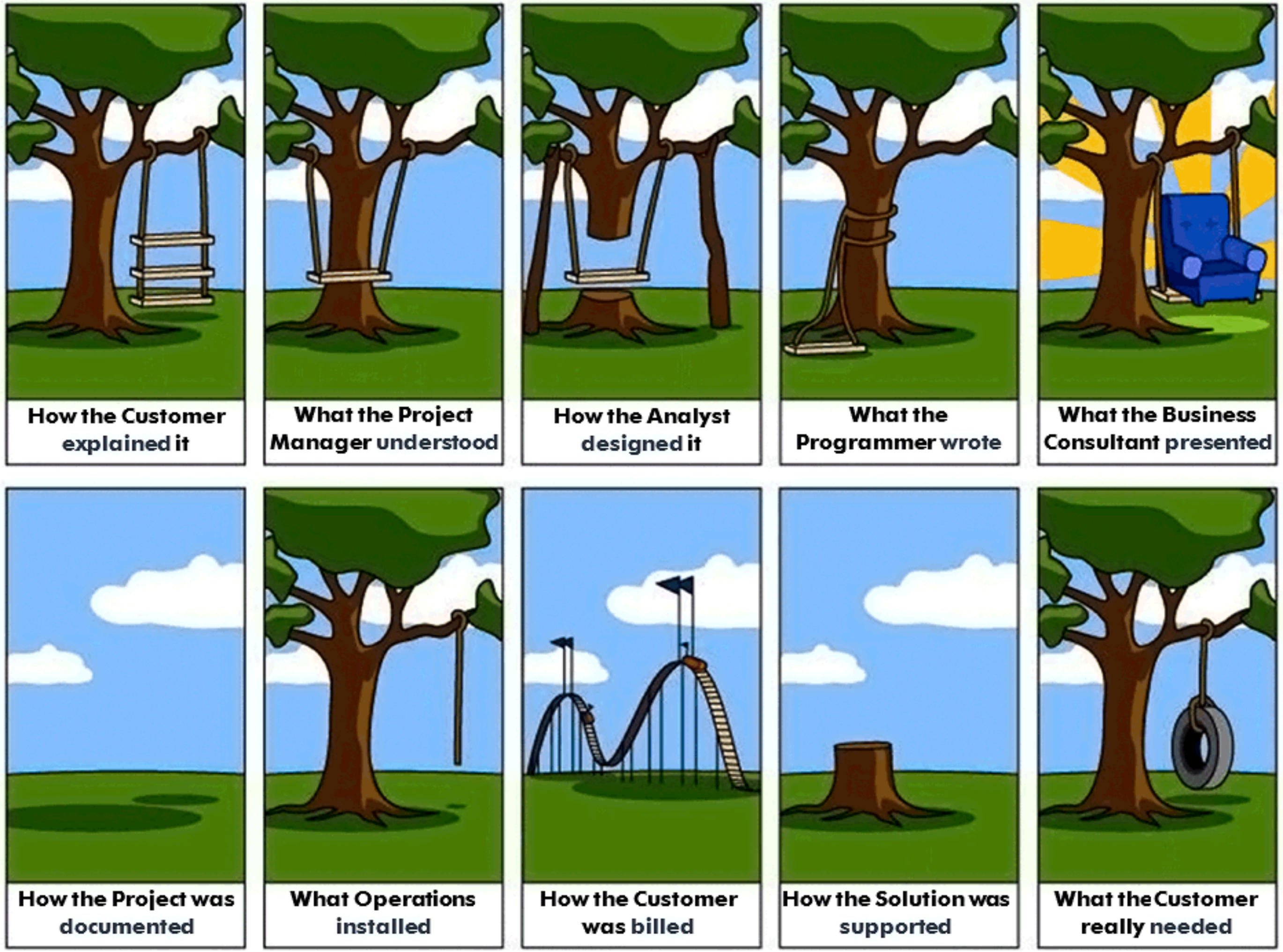

How the Customer explained it
What the Project Manager understood
How the Analyst designed it
What the Programmer wrote
What the Business Consultant presented
How the Project was documented
What Operations installed
How the Customer was billed
How the Solution was supported
What the Customer really needed

Intro To Process

Milestones, Estimation, Planning

Slides adapted from CMU 17-313 (credit to Michael Hilton and others)

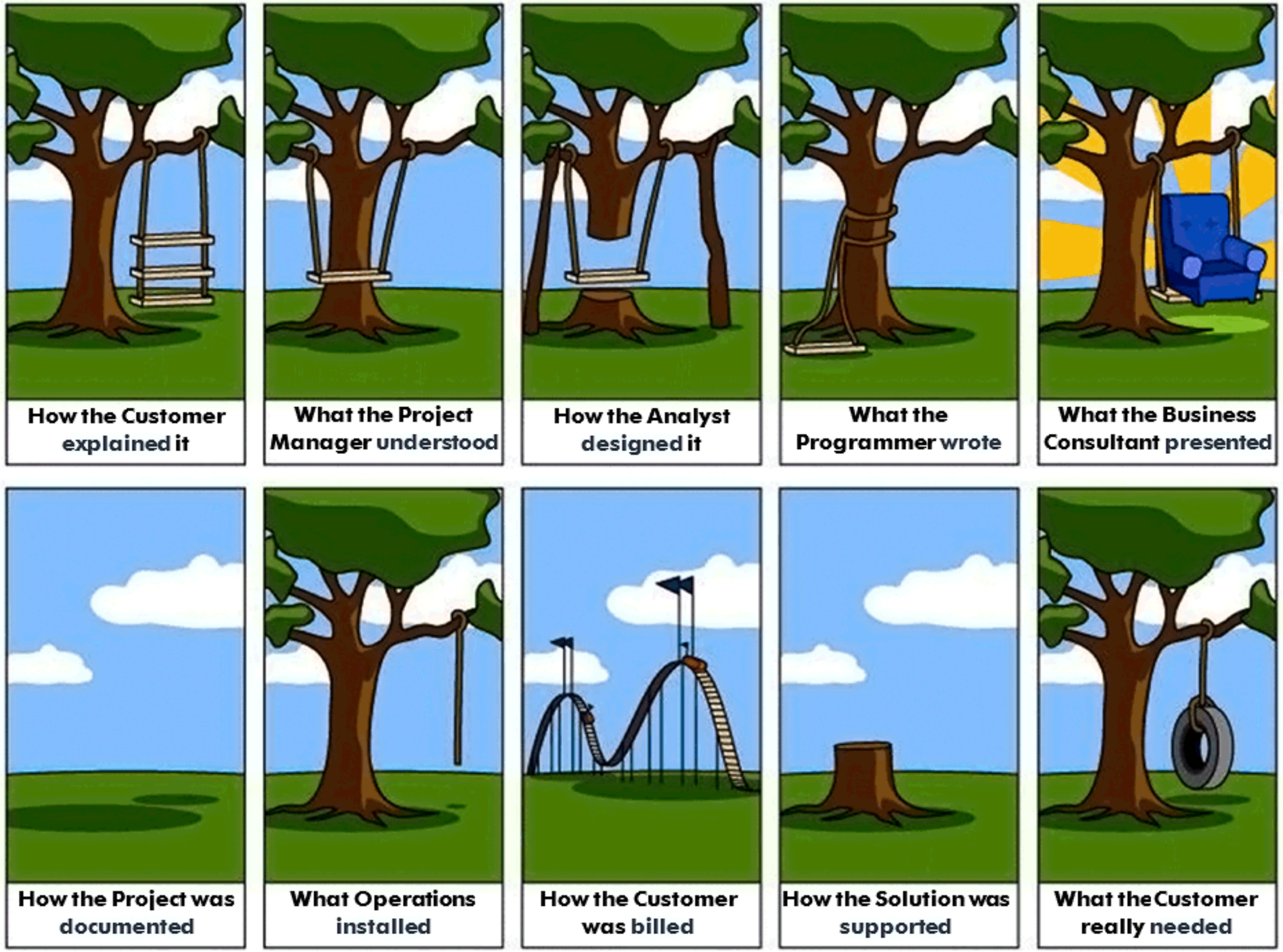
Learning Goals

- Today:
  - Recognize the importance of process
  - Understand the difficulty of measuring progress
  - Identify why software development has project characteristics
  - Use milestones for planning and progress measurement

Software Process

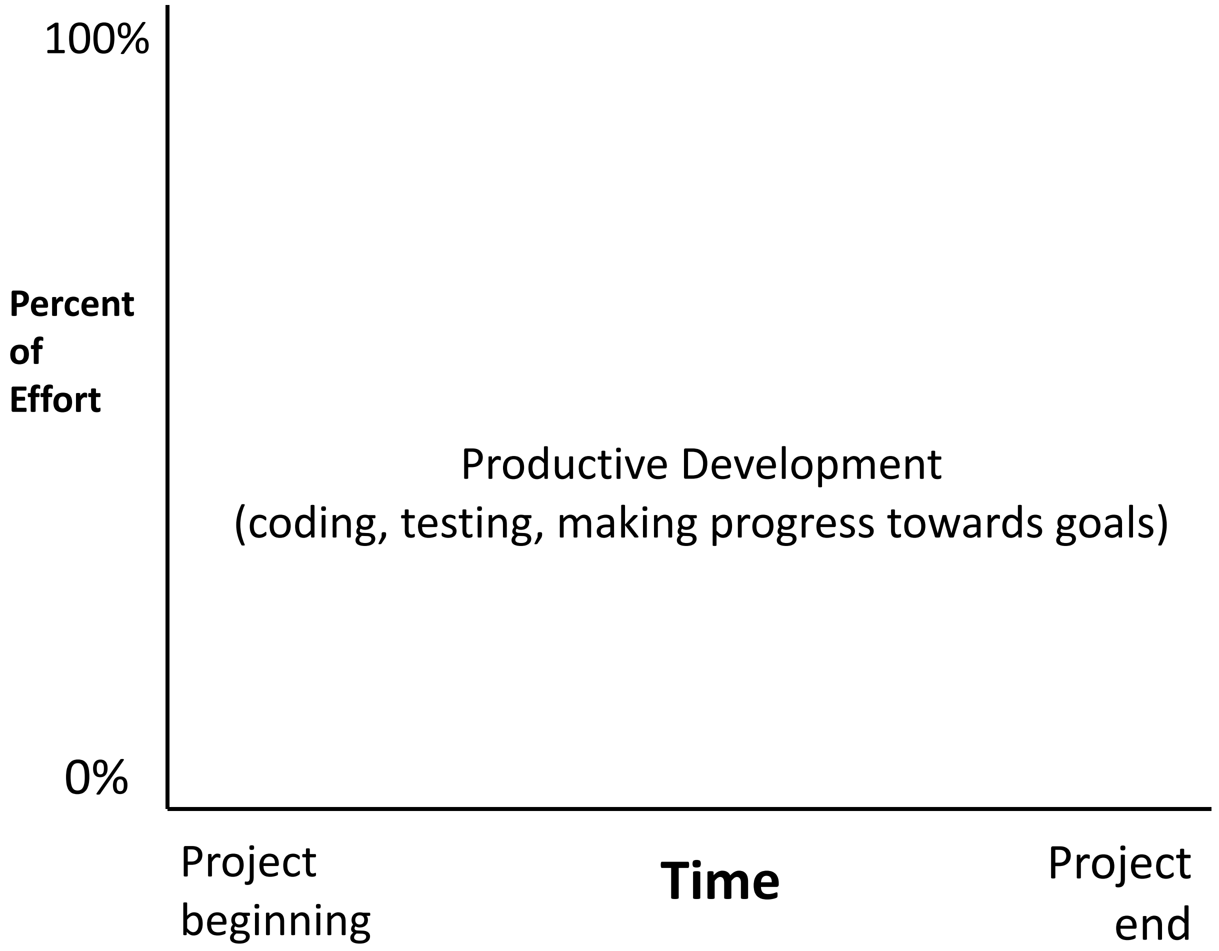
“The set of activities and associated results that produce a software product”

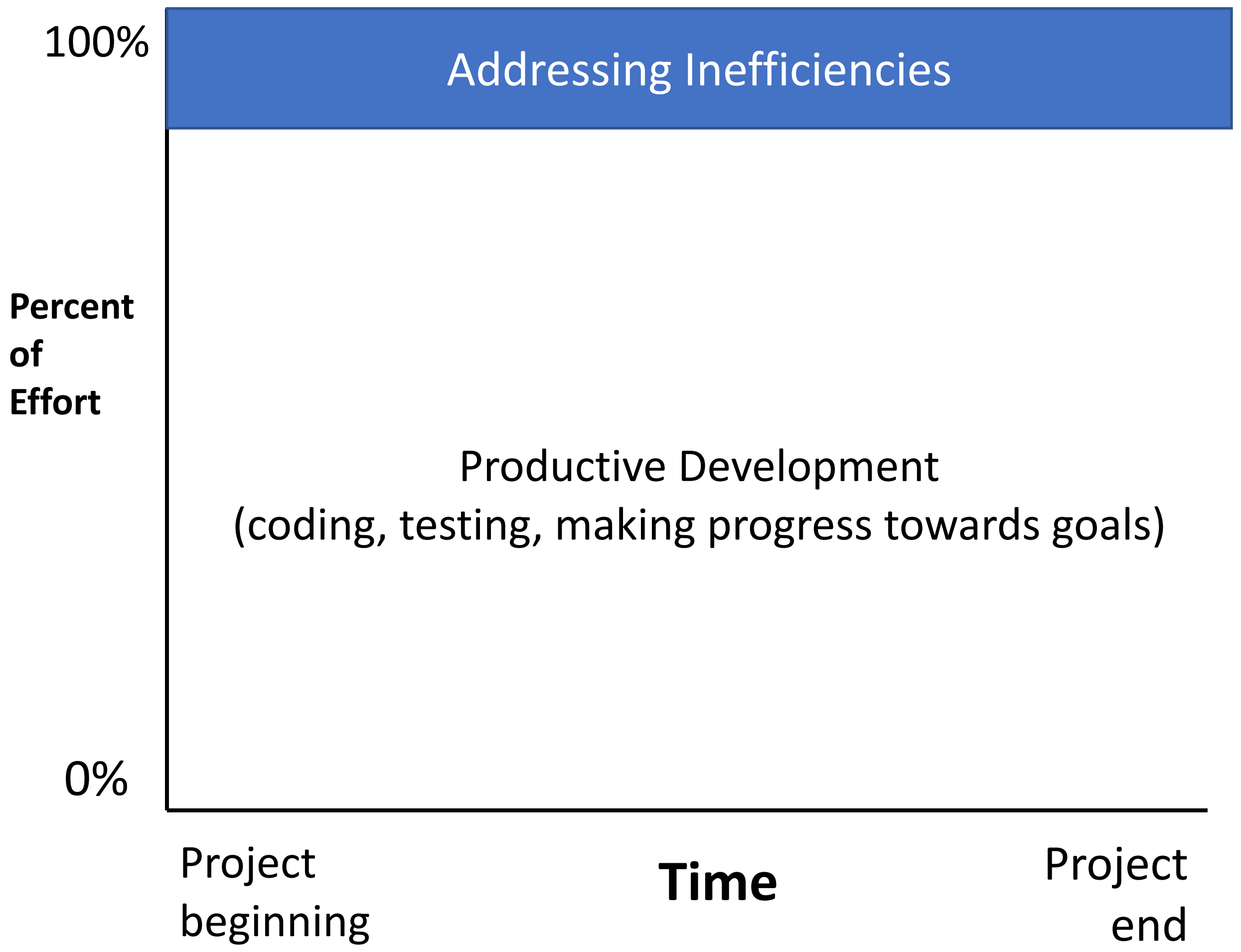
Sommerville, SE, ed. 8


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

How to develop software?

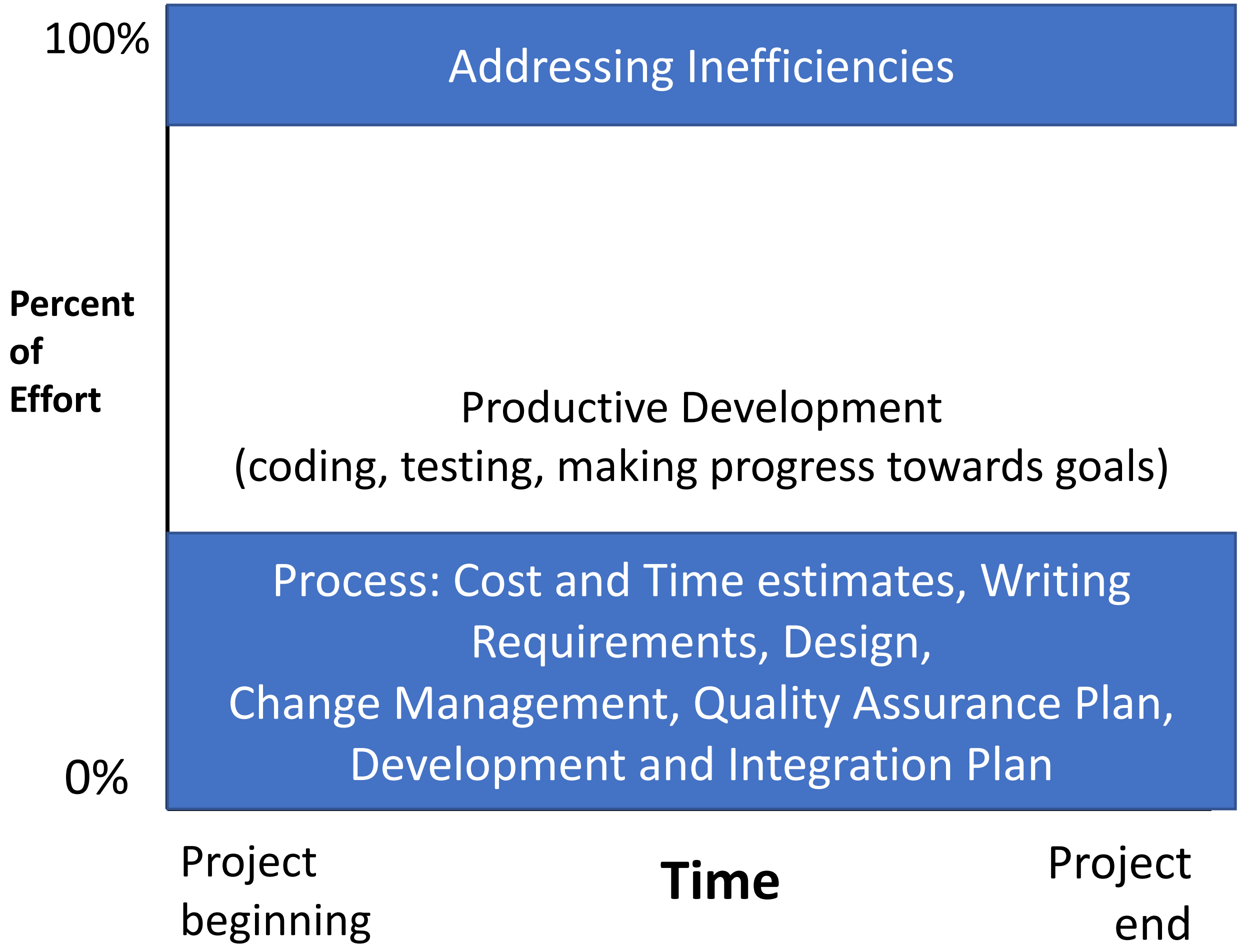
1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1

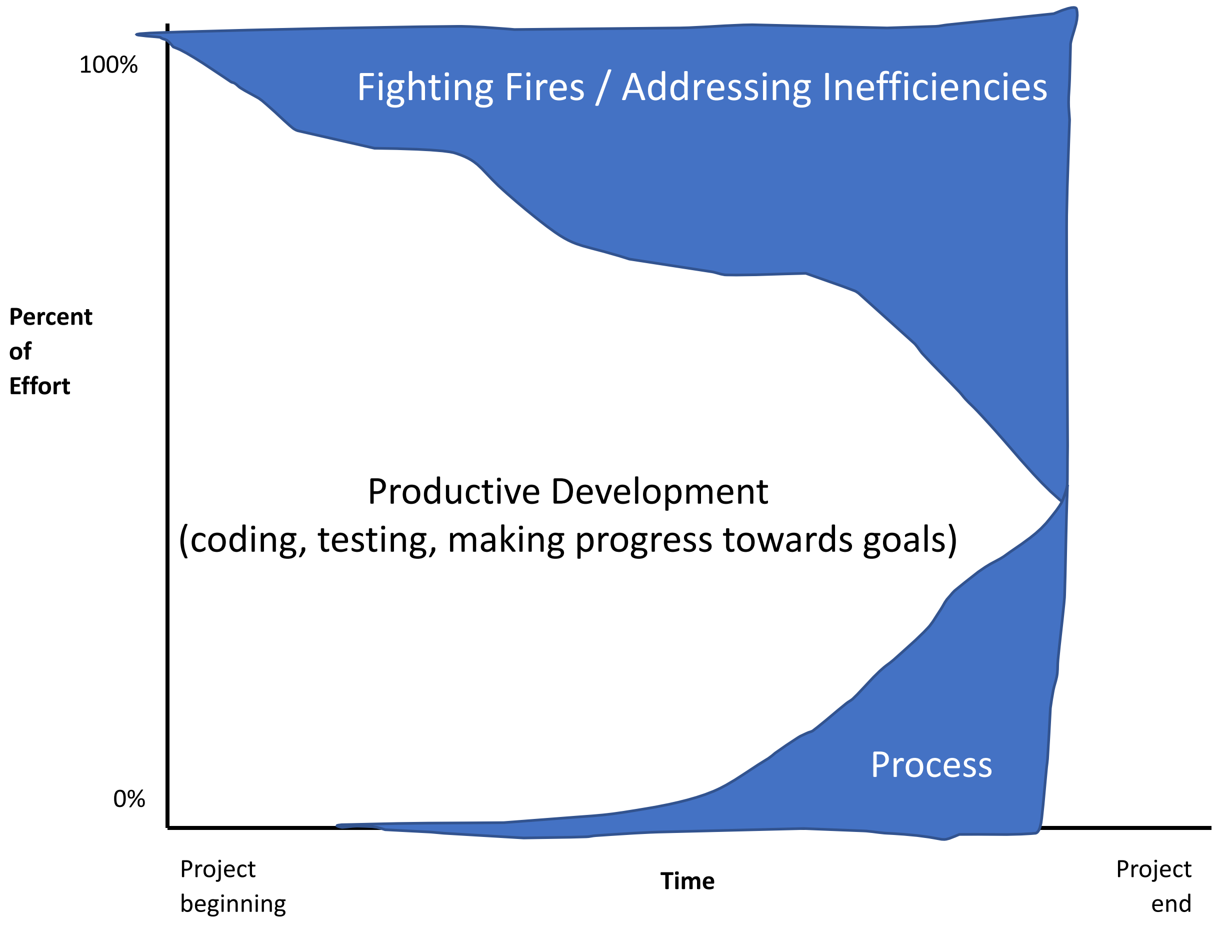

100%
Percent of Effort
0%
Productive Development
(coding, testing, making progress towards goals)
Project beginning
Time
Project end


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

Your manager asks you to follow a process

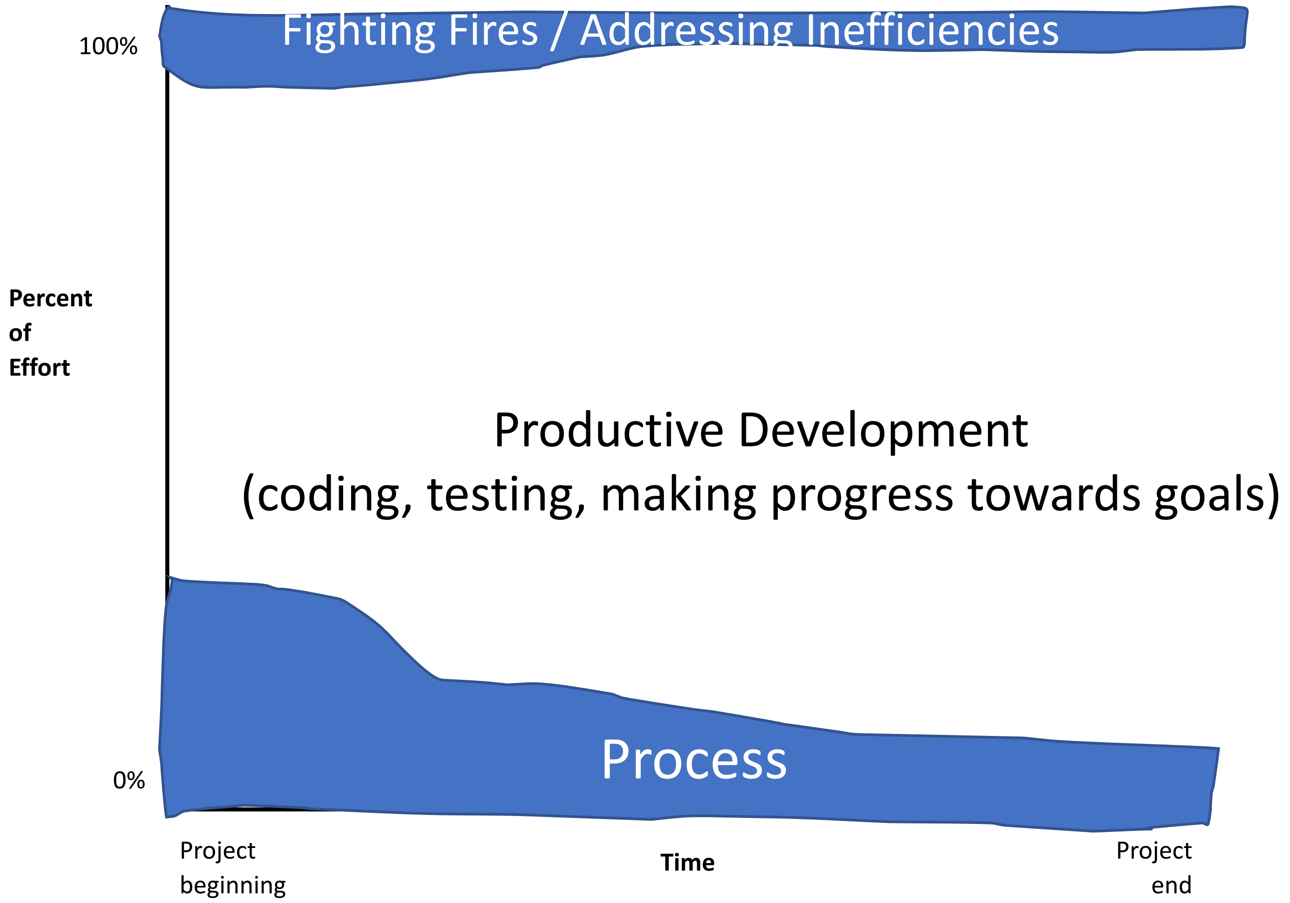
- Writing down all requirements
- Require approval for all changes to requirements
- Use version control for all changes
- Track all reported bugs
- Review requirements and code
- Break down development into smaller tasks and schedule and monitor them
- Planning and conducting quality assurance
- Have daily status meetings
- Use Docker containers to push code between developers and operation


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


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

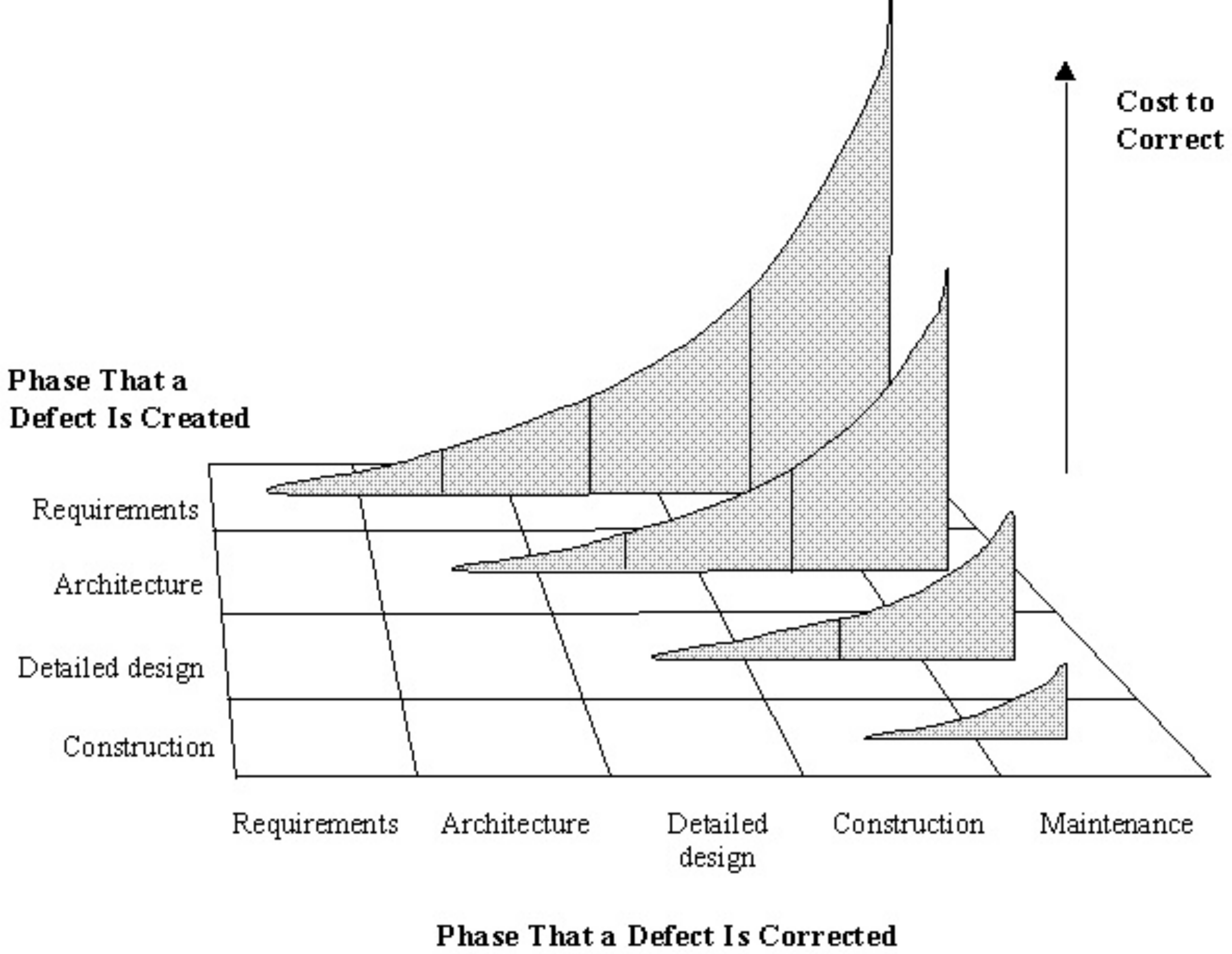
Example process issues

- Change Control: Mid-project informal agreement to changes suggested by customer or manager. Project scope expands 25-50%
- Quality Assurance: Late detection of requirements and design issues. Test-debug-reimplement cycle limits development of new features. Release with known defects.
- Defect Tracking: Bug reports collected informally, forgotten
- System Integration: Integration of independently developed components at the very end of the project. Interfaces out of sync.
- Source Code Control: Accidentally overwritten changes, lost work.
- Scheduling: When project is behind, developers are asked weekly for new estimates.



Hypothesis: Process increases flexibility and efficiency

Ideal Curve: Upfront investment for later greater returns


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

Questions/Discussion

Planning

Time estimation

I'M JUST OUTSIDE TOWN, SO I SHOULD BE THERE IN FIFTEEN MINUTES.
ACTUALLY, IT'S LOOKING MORE LIKE SIX DAYS.
NO, WAIT, THIRTY SECONDS.
THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

https://xkcd.com/612/

Activity: Estimate Time

Task A: Simple web version of the Monopoly board game with San Diego street names

Team: just you

Task B: Bank smartphone app

Team: you with team of 4 developers, one experienced with iPhone apps, one with background in security

* Estimate in 8h days (20 work days in a month, 220 per year)

```
My Task A estimate: ___
My Task B estimate: ___

Other Task A estimate: __
Other Task B estimate: __

Other Task A estimate: __
Other Task B estimate: __
```

Revise Time Estimate

- Do you have comparable experience to base an estimate off of?
- How much design do you need for each task?
- Break down the task into ~5 smaller tasks and estimate them.
- Revise your overall estimate if necessary

π

Measuring Progress?

- “I’m almost done with the app. The frontend is almost fully implemented. The backend is fully finished except for the one stupid bug that keeps crashing the server. I only need to find the one stupid bug, but that can probably be done in an afternoon. We should be ready to release next week.”

Measuring Progress?

- Developer judgment: x% done
- Lines of code?
- Functionality?
- Quality?

Milestones and deliverables make progress observable

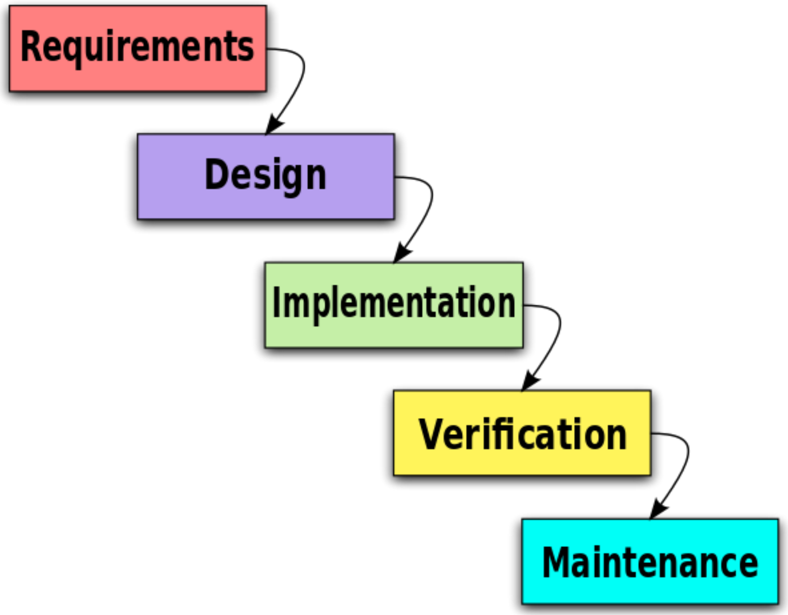
Milestone: clear end point of a (sub)tasks

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

Deliverable: Result for customer

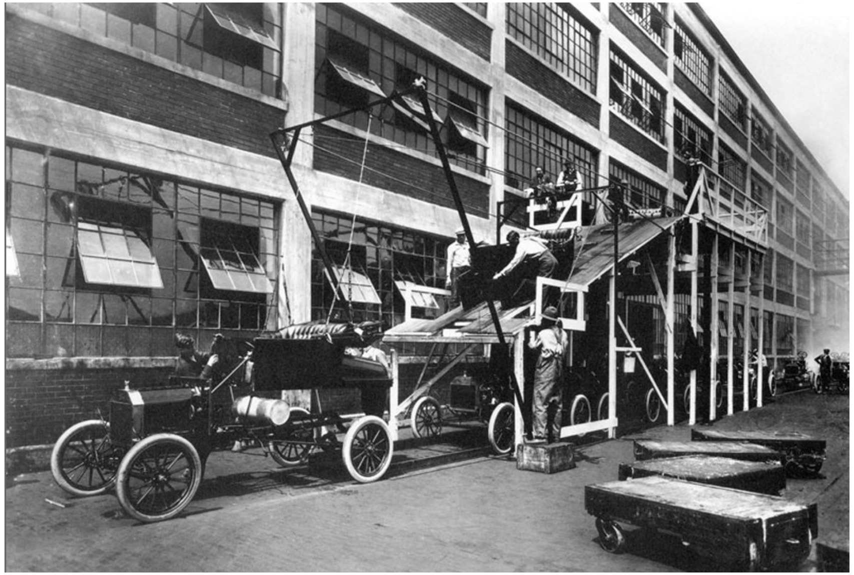
- Similar to milestone, but for customers
- Reports, prototypes, completed subsystems

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)



Lean production adapts to variable demand

Taiichi Ohno

Toyota Production System (TPS)

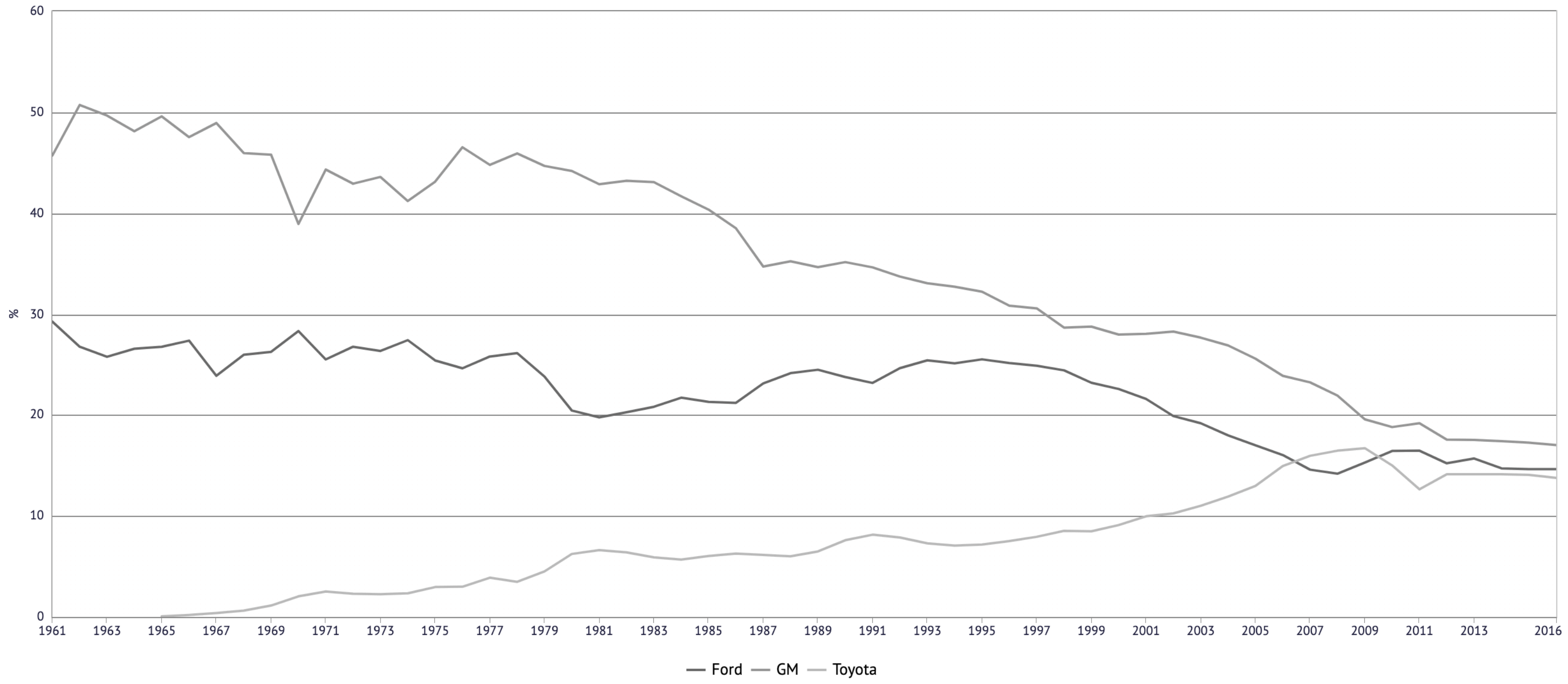
- Build only what is needed, only when it is needed.
- Use the “pull” system to avoid overproduction. (Kanban)
- Stop to fix problems, to get quality right from the start (Jidoka)
- Workers are multi-skilled and understand the whole process; take ownership

Lots of software buzzwords invented recently build on these ideas

- Just-in-time, DevOps, Shift-Left

See also: “The machine that changed the world” by James P Womack et al. The Free Press, 2007.

US vehicle sales market share; 1961—2016 (source: knoema.com)



Agile

Agile Overview

- Keep a *prioritized* list of user stories in a **backlog**
- The **product owner** sets priorities of backlog items
- Divide work into **sprints** (often, two weeks long)
- Conceptually: at end of each sprint, you could ship
- The **scrum master** keeps the process on track
  - Removes barriers to success

Sprint Structure

- Start with a **planning meeting**
  - First, **estimate** user stories
  - Then, **commit** to user stories individually
- Every day: **standup meeting**
  - What did I do yesterday?
  - What will I do today?
  - Am I stuck?
- Then: **sprint review** and **sprint retrospective**

Sprint review

- For each user story: demo!
- If acceptance criteria achieved, great.
    - Otherwise, user story goes back on the backlog.

Sprint retrospective

- Discuss how the sprint went
- Refine interactions, processes, tools
- Identify and solve problems
- Decide on changes to improve effectiveness

Expectations

- Monday: populate backlog; start sprint 1
    - It's okay if you take Monday to get the backlog ready
    - At least start writing code after the exam
    - Hold a sprint planning meeting
- Use branches to work on features; then merge & delete branches
- Only merge after a successful peer code review
    - Every code review should consider correctness, architecture, style, and readability
- Sprint review at end of Week 6