ECE444: Software Engineering

Introduction to Process

Shurui Zhou

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

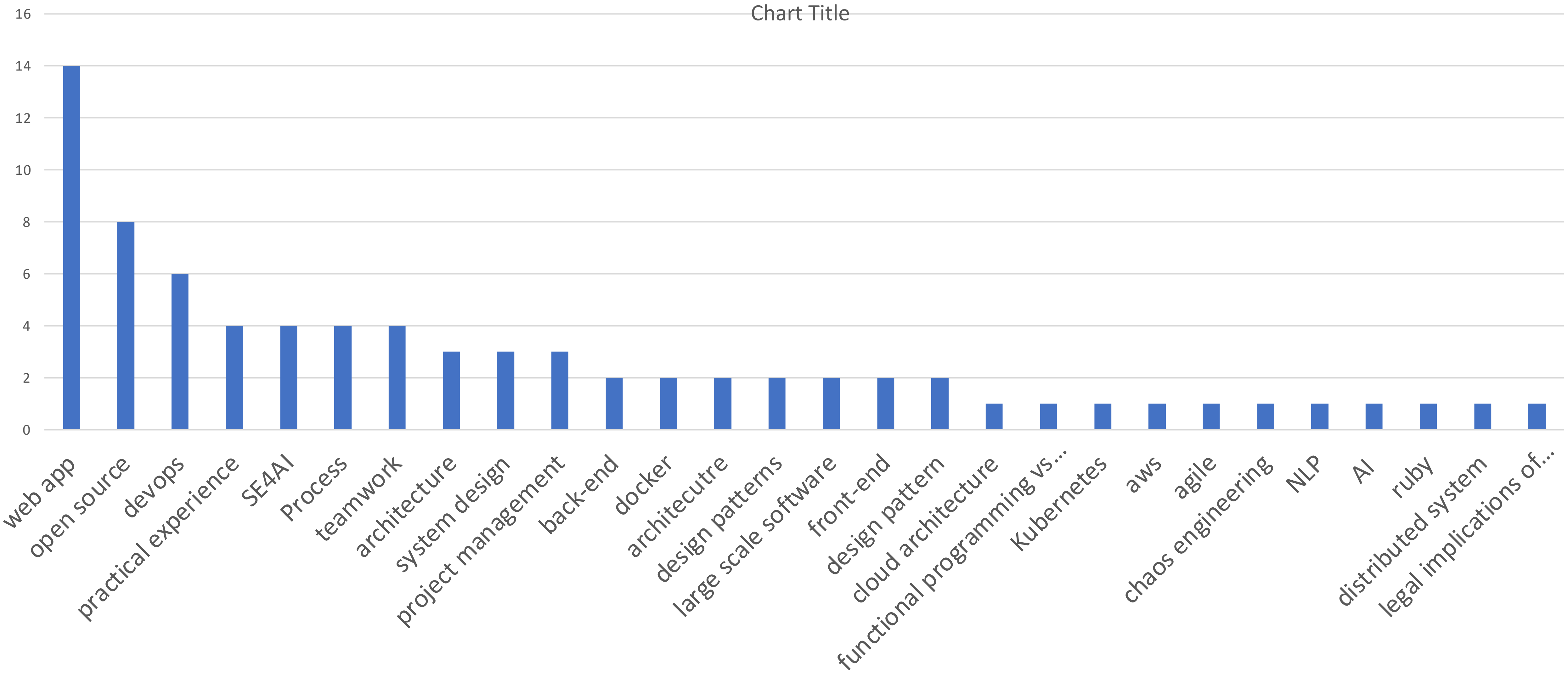
Administrivia

- **Proj1_Milestone0** due **9/16** 11:59pm EST
    - team name
    - project proposal
- **Vote for idea** due **9/17** 11:59pm EST
- About Lab
    - tutorial by TA
    - lab task, submitting by Friday (participation)
    - Q&A, group meeting (not required to stay for the whole session)

Why did you pick this class?

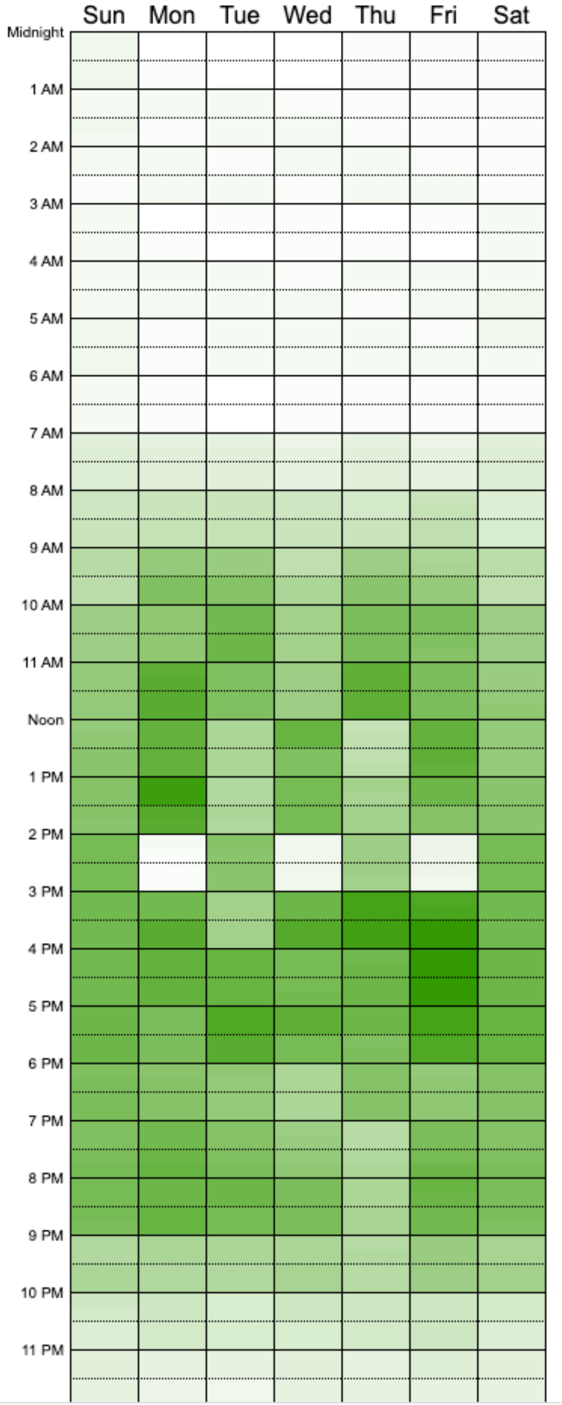
- I'm planning to work in industry after graduation
- To get strong first-hand experience
- improve my project management and programming skills
- To find a way to better understanding and editing the code written by others
- I am interested in web page design.
- It is a hot topic!

What do you want to learn?



Office Hour

- 68 replied
- Friday 4-5pm EST
- By appointment



Learning Goals

- Recognize the Importance of process
- Understand the difficulty of measuring progress
- Use milestones for planning and progress measurement
- Understand backlogs and user stories

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]

2013

- 2M people working on 300K software projects in the US
- 1/3 - 2/3 exceed schedule and budget targets before delivery
- Of the most expensive software projects, about half will eventually be canceled for being out of control.

https://ptgmedia.pearsoncmg.com/images/9781572316218/samplepages/9781572316218.pdf

Software projects succeed or fail based on how carefully they are planned and how deliberately they are executed

Process

How to develop software?

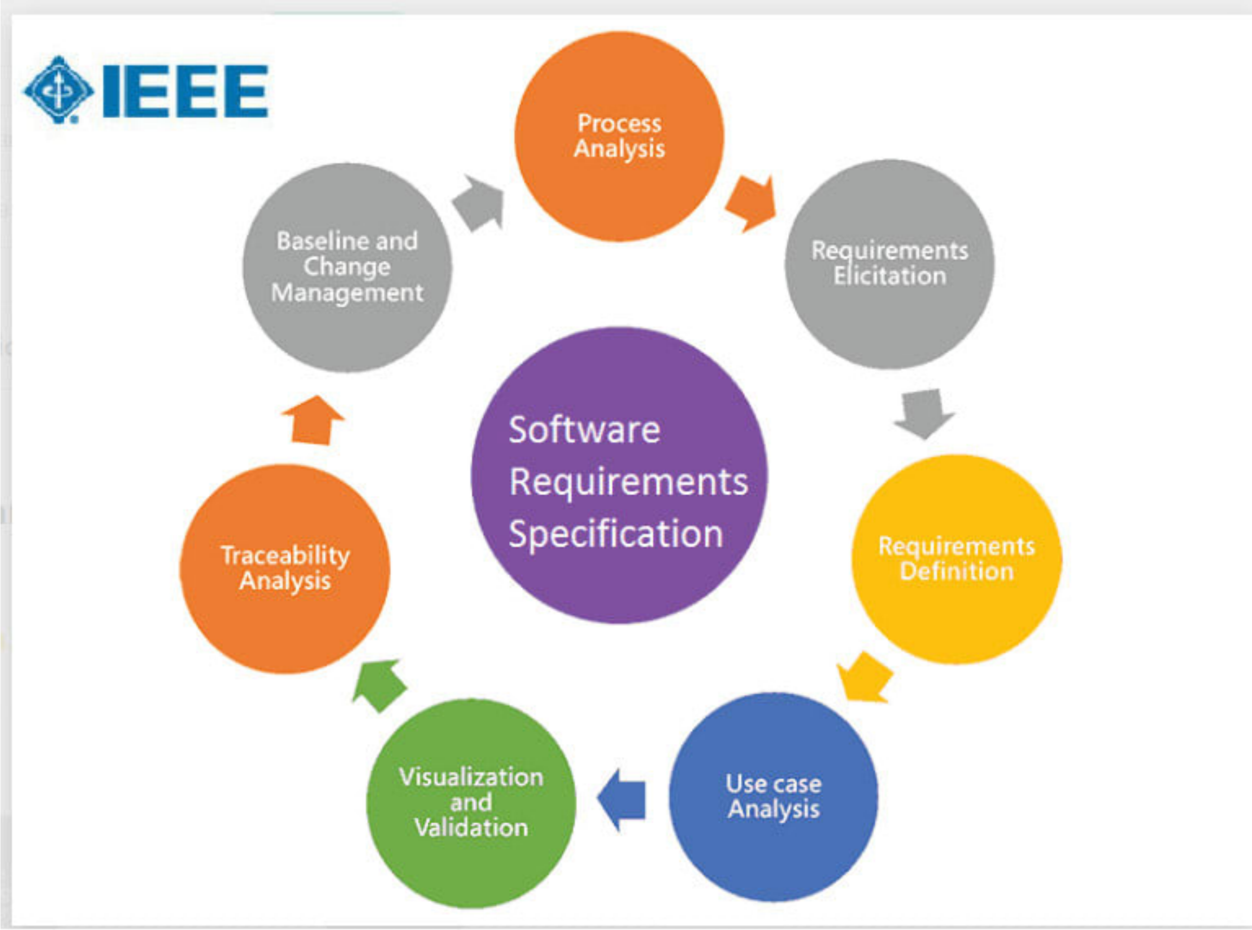
1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1

Software Process

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

Example of Process Decisions

- Writing down all requirements

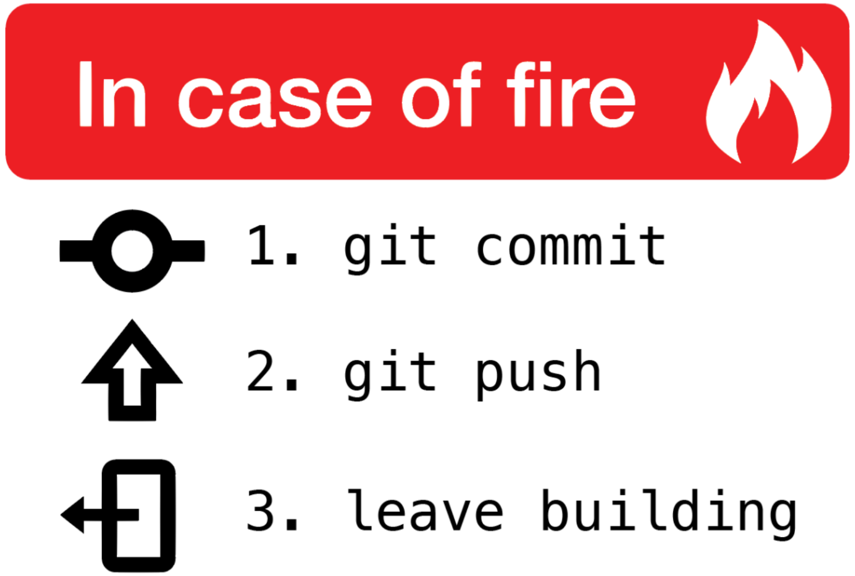


Example of Process Decisions

- Writing down all requirements
- Require approval for all changes to requirements

Example of Process Decisions

- Writing down all requirements
- Require approval for all changes to requirements
- Use version control for all changes



In case of fire

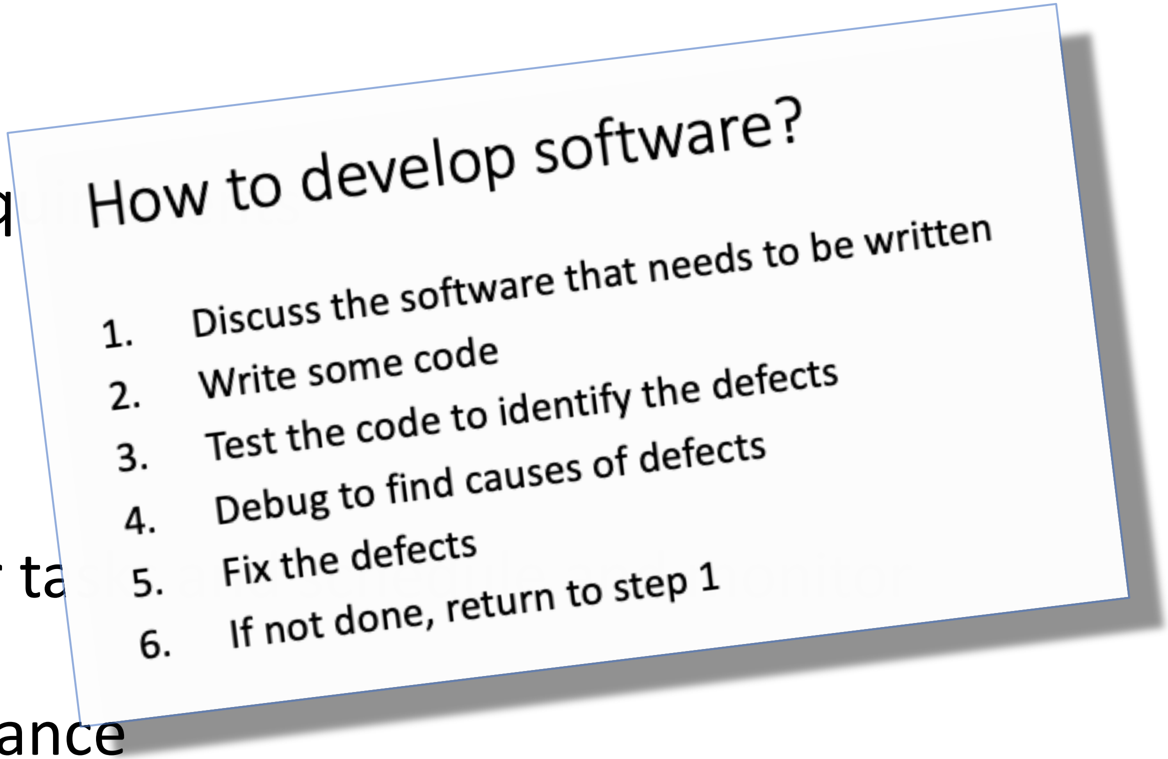
1. git commit
2. git push
3. leave building

Example of Process Decisions

- 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

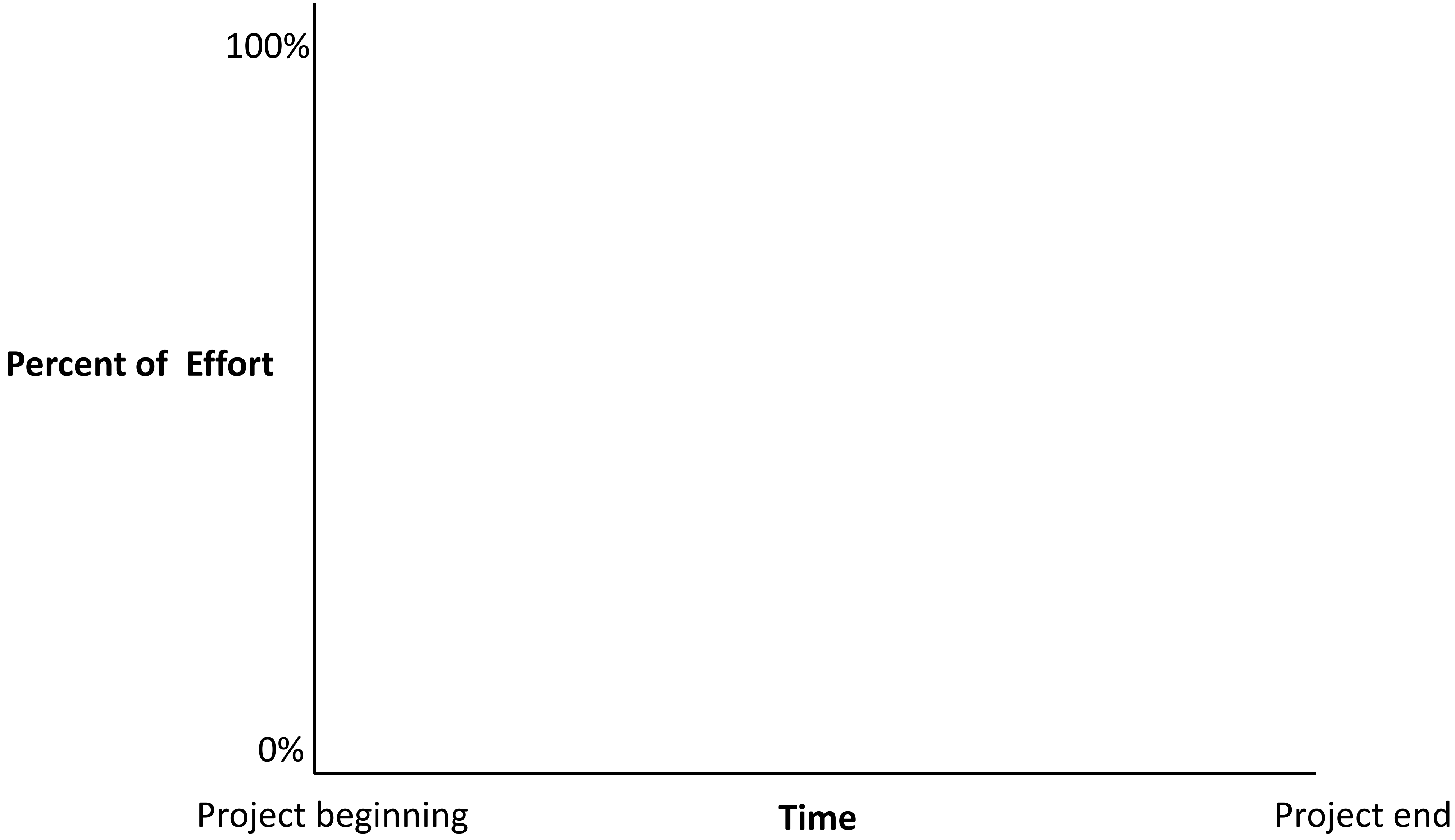
Example of Process Decisions

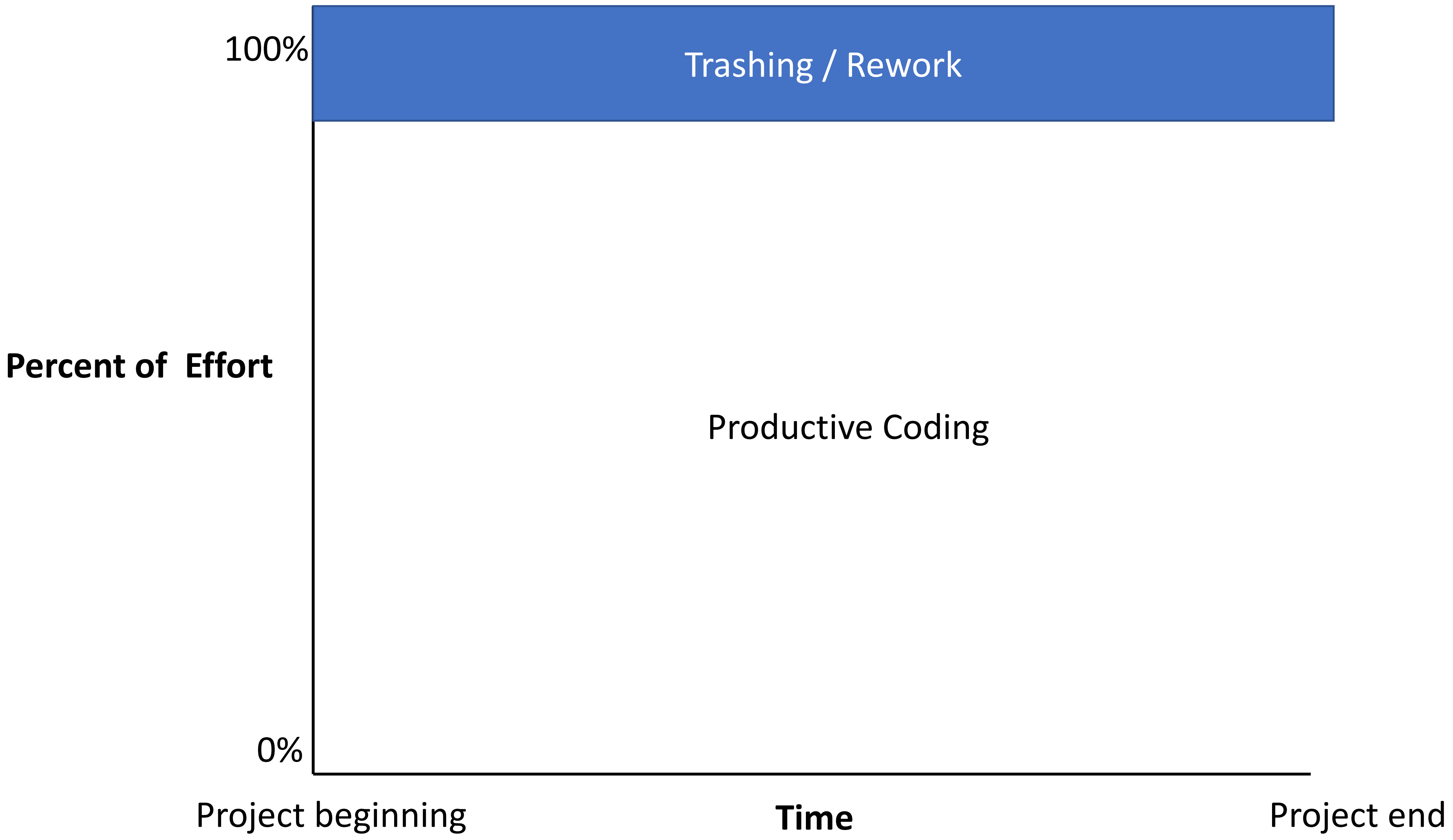
- Writing down all requirements
- Require approval for all changes to req
- Use version control for all changes
- Track all reported bugs
- Review requirements and code
- Break down development into smaller ta
  them
- Planning and conducting quality assurance
- Have daily status meetings
- Use Docker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

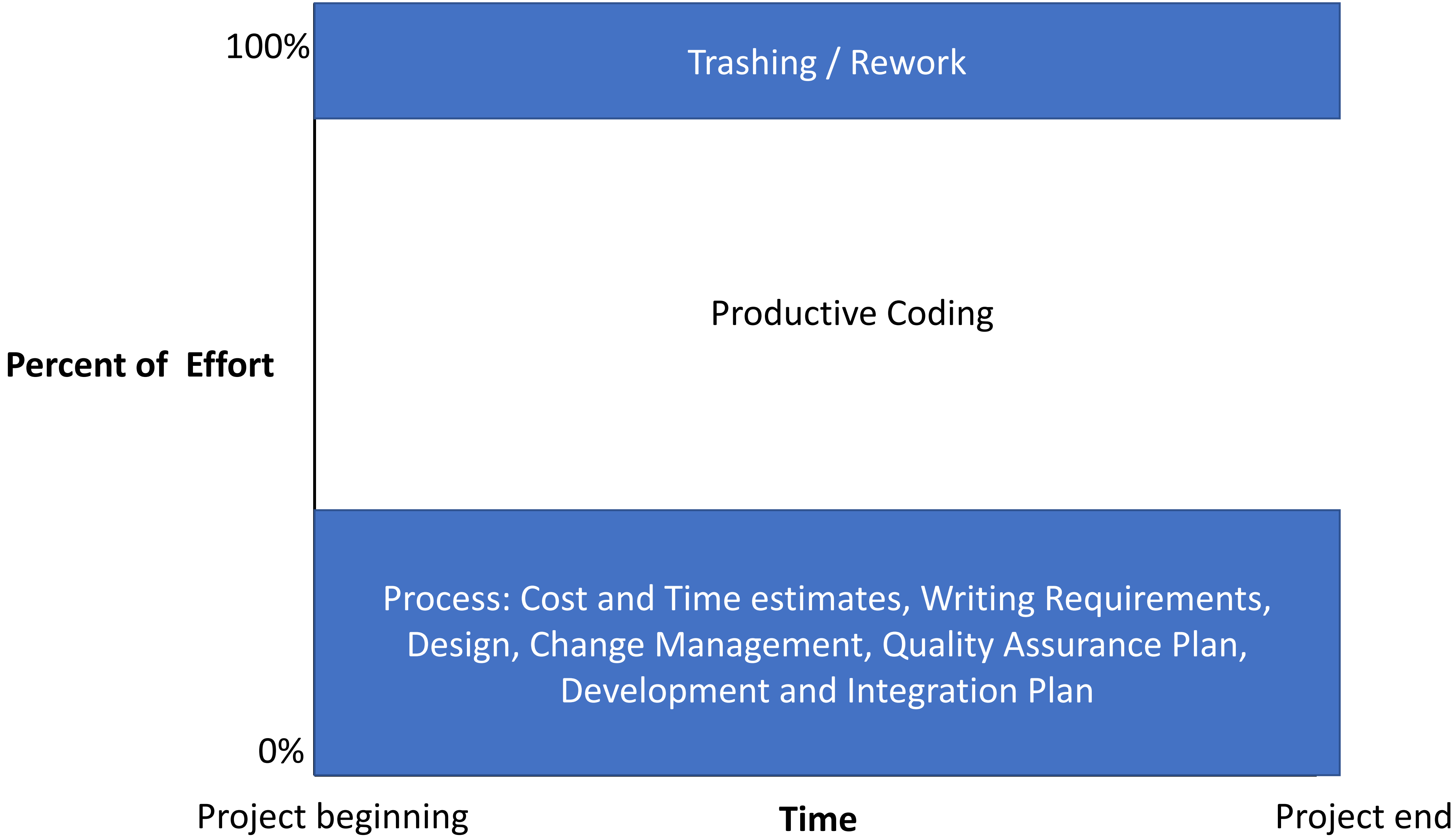


How to develop software?

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%
Project beginning
Time
Project end


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



100%

Trashing / Rework

Productive Coding

Percent of Effort

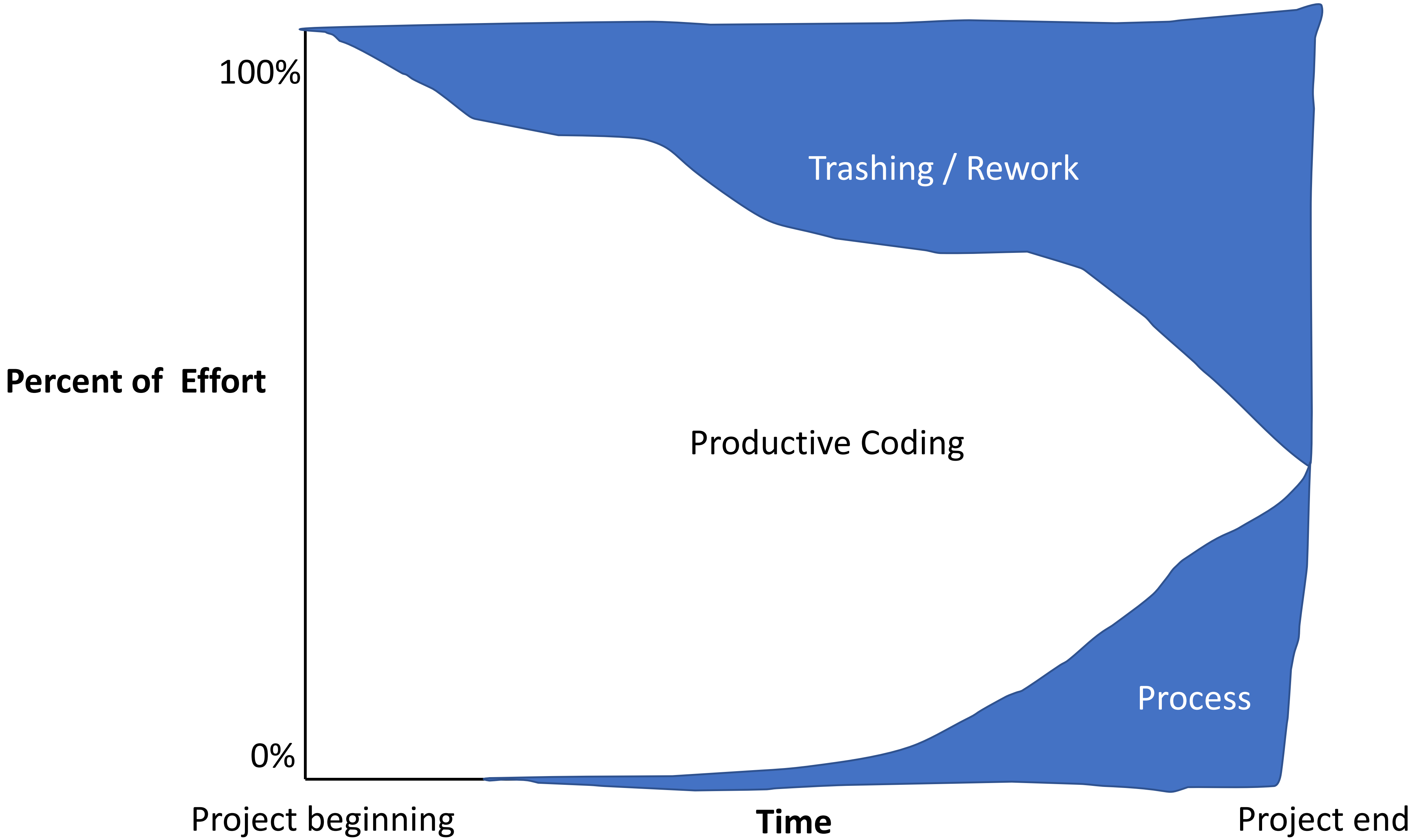
Process: Cost and Time estimates, Writing Requirements, Design, Change Management, Quality Assurance Plan, Development and Integration Plan

0%

Project beginning

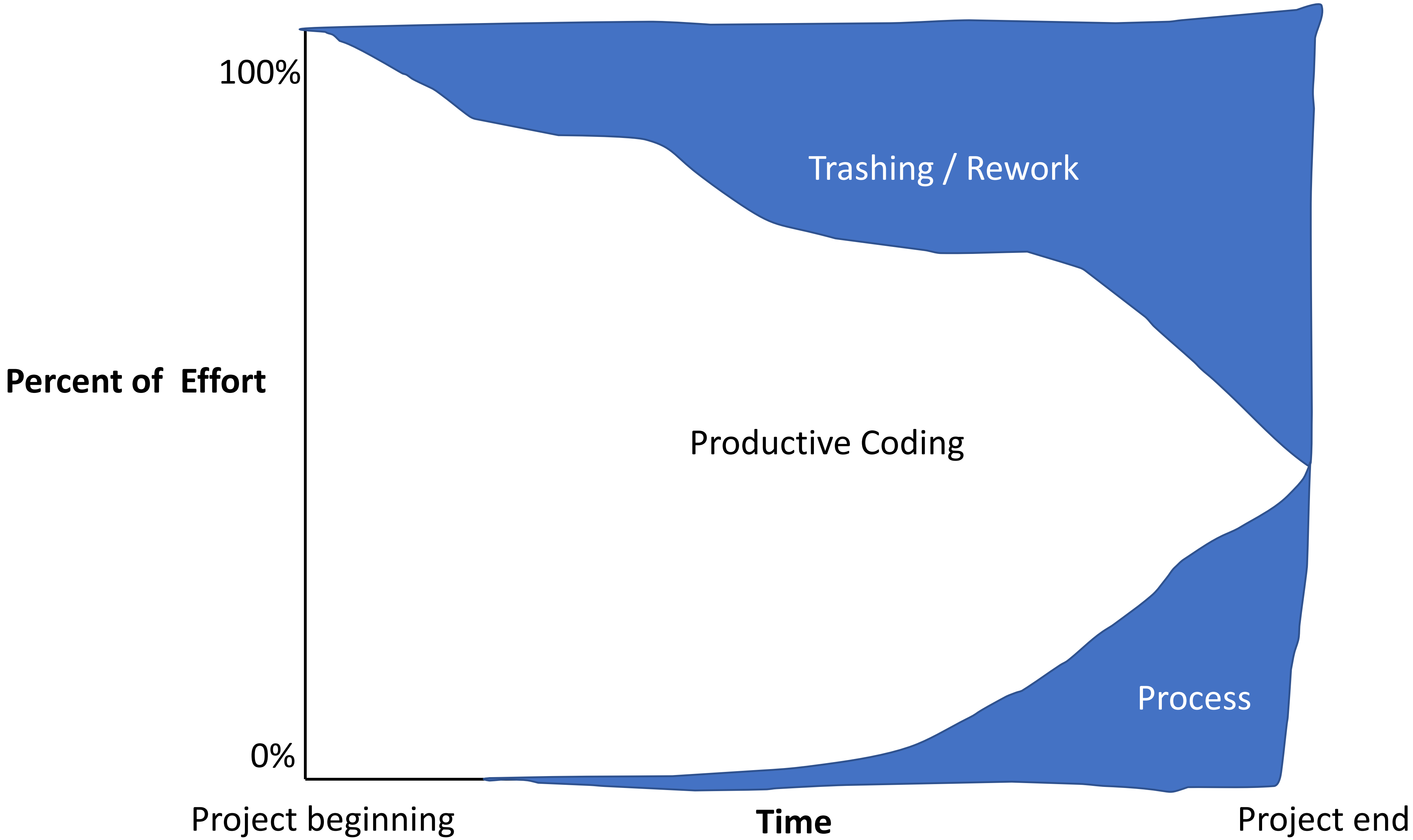
Time

Project end


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

Survival Mode

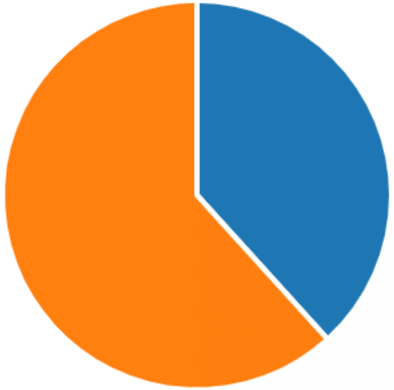
- Missed deadlines -> "solo development mode" to meet own deadlines
- Ignore integration work
- Stop interacting with testers, technical writers, managers, …

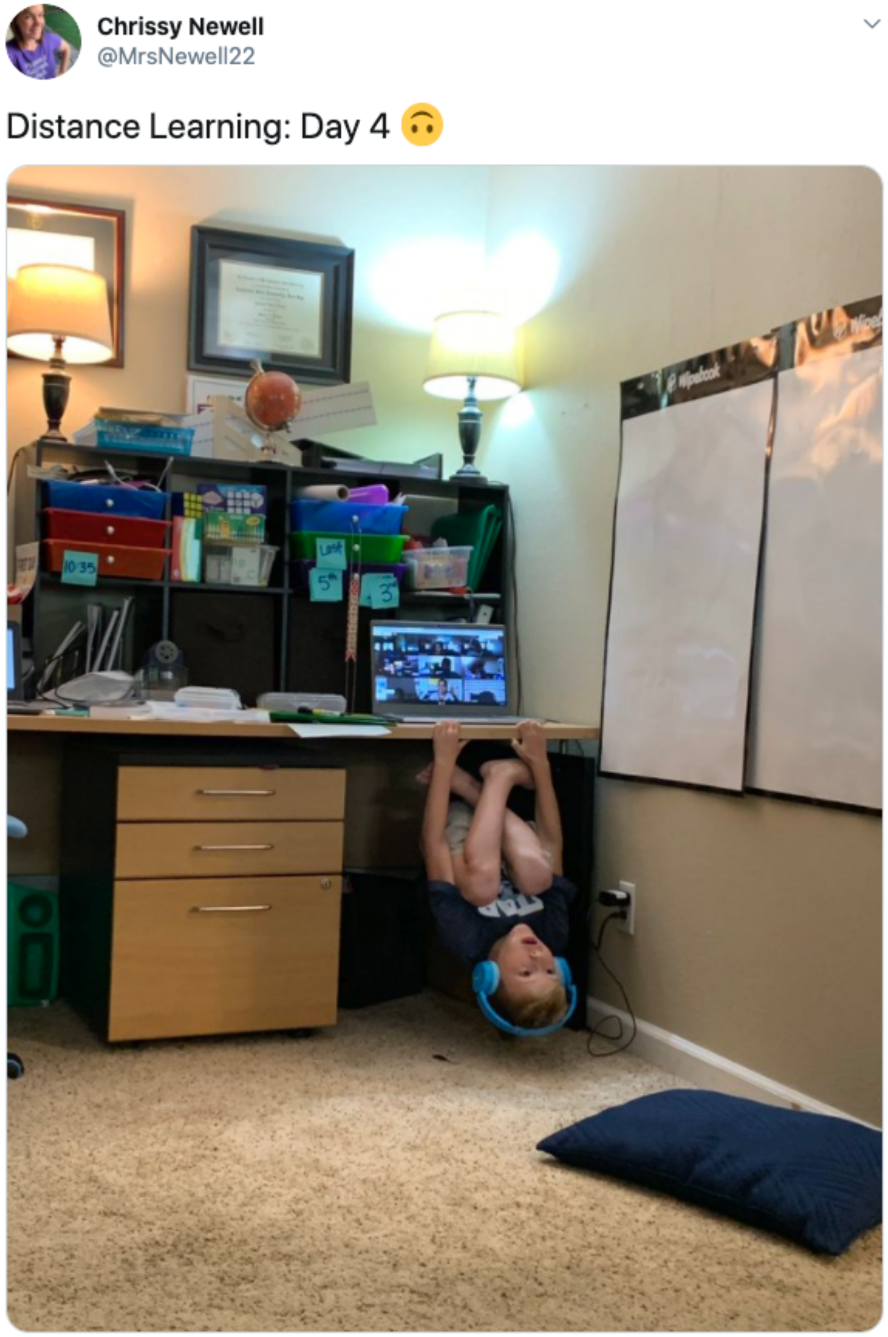

100%
Percent of Effort
Trashing / Rework
Productive Coding
Process
0%
Project beginning
Time
Project end

26. I saw at least one software project fail and (believe to) know the main reason

More Details

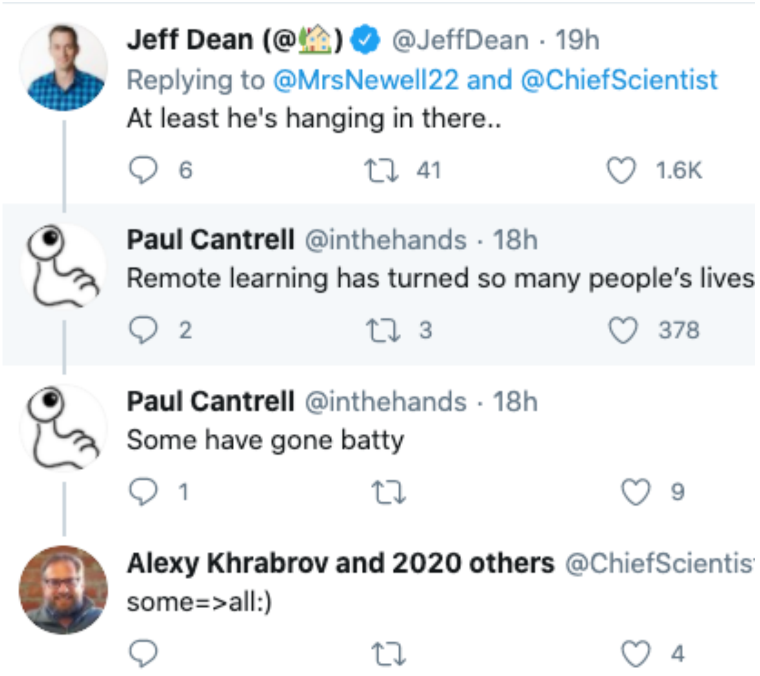
●	yes	31
●	No	50





Chrissy Newell
@MrsNewell22

Distance Learning: Day 4 🙃



Jeff Dean (@🏡) ✓ @JeffDean · 19h
Replying to @MrsNewell22 and @ChiefScientist
At least he's hanging in there..

6 41 1.6K

Paul Cantrell @inthehands · 18h
Remote learning has turned so many people's lives

2 3 378

Paul Cantrell @inthehands · 18h
Some have gone batty

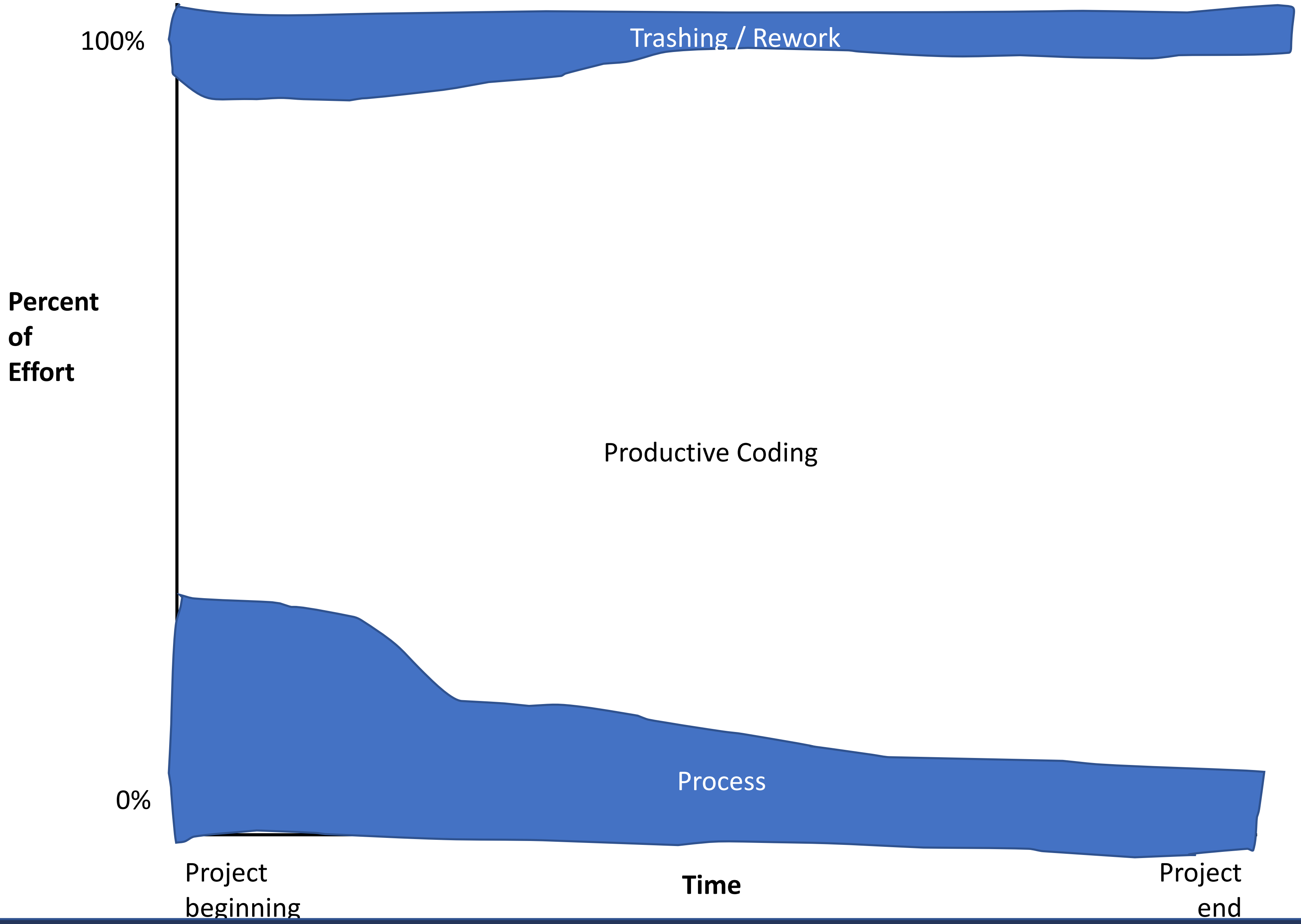
1 9

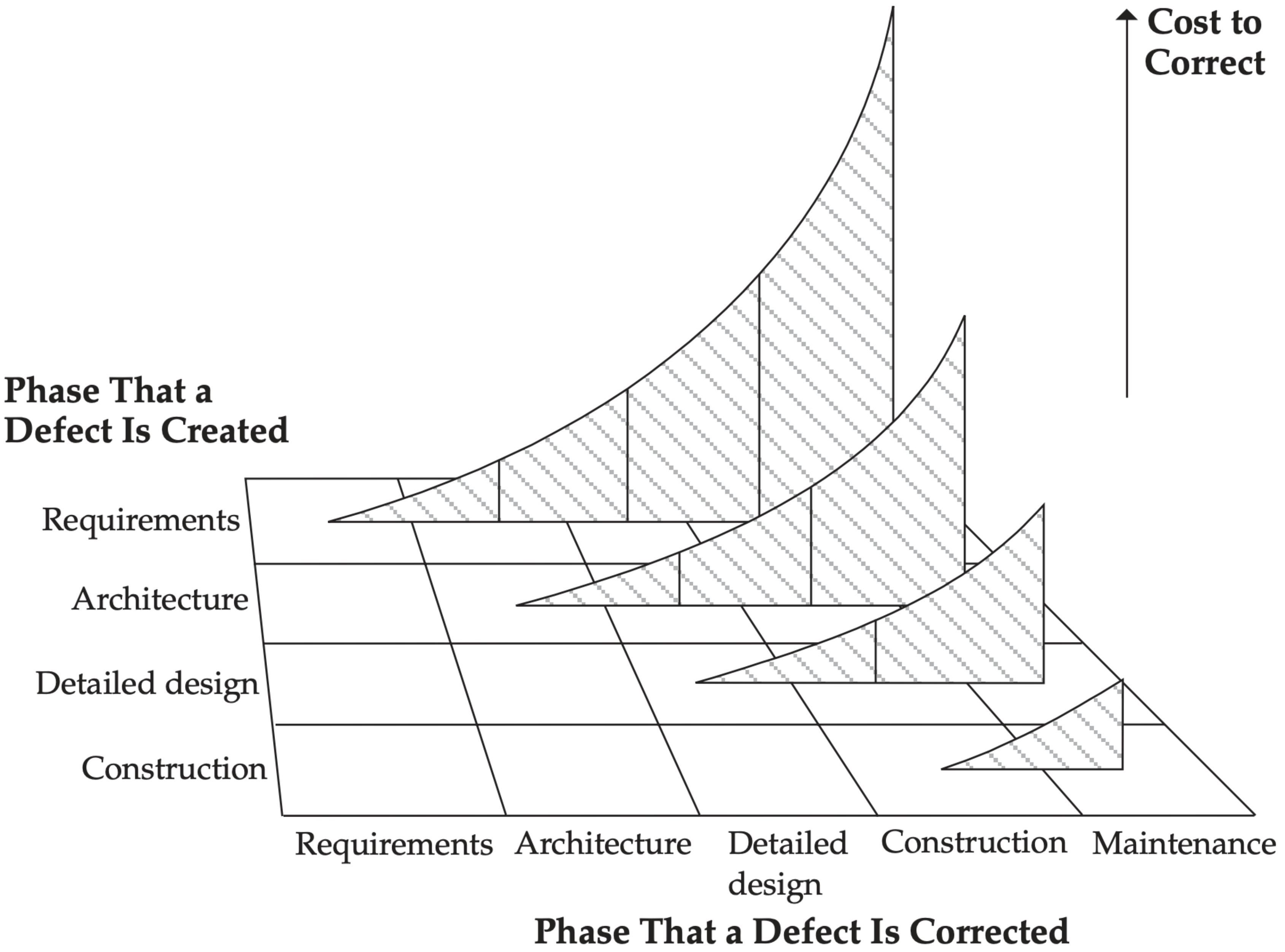
Alexy Khrabrov and 2020 others @ChiefScientis
some=>all:)

4

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.


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


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

Real world cases

Organizations that have explicitly focused on improving their development processes have, over several years, cut their time-to-market by about one-half and reduced their costs and defects by factors of 3 to 10.

LOCKHEED MARTIN

5 yr, cost -75%, time - 40%, defects - 90%

NASA

8 yr, cost -50%, defects - 75%

MOTOROLA

xerox™

Planning

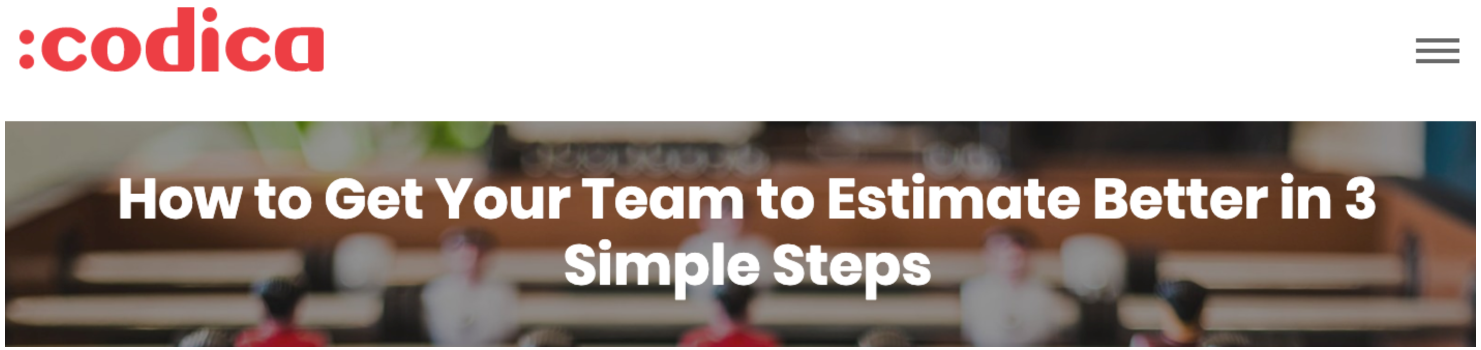
Task: Estimate Time

- a web application of Trip guide (booking, scheduling, route planning...)

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

Revise Time Estimate

- Remember the GIS system experience?
- Is GIS similar/different/easier/more challenging/reusable?
- How much design did you do?
- Break down the task into ~5 smaller tasks and estimate them.
- Revise your overall estimate if necessary

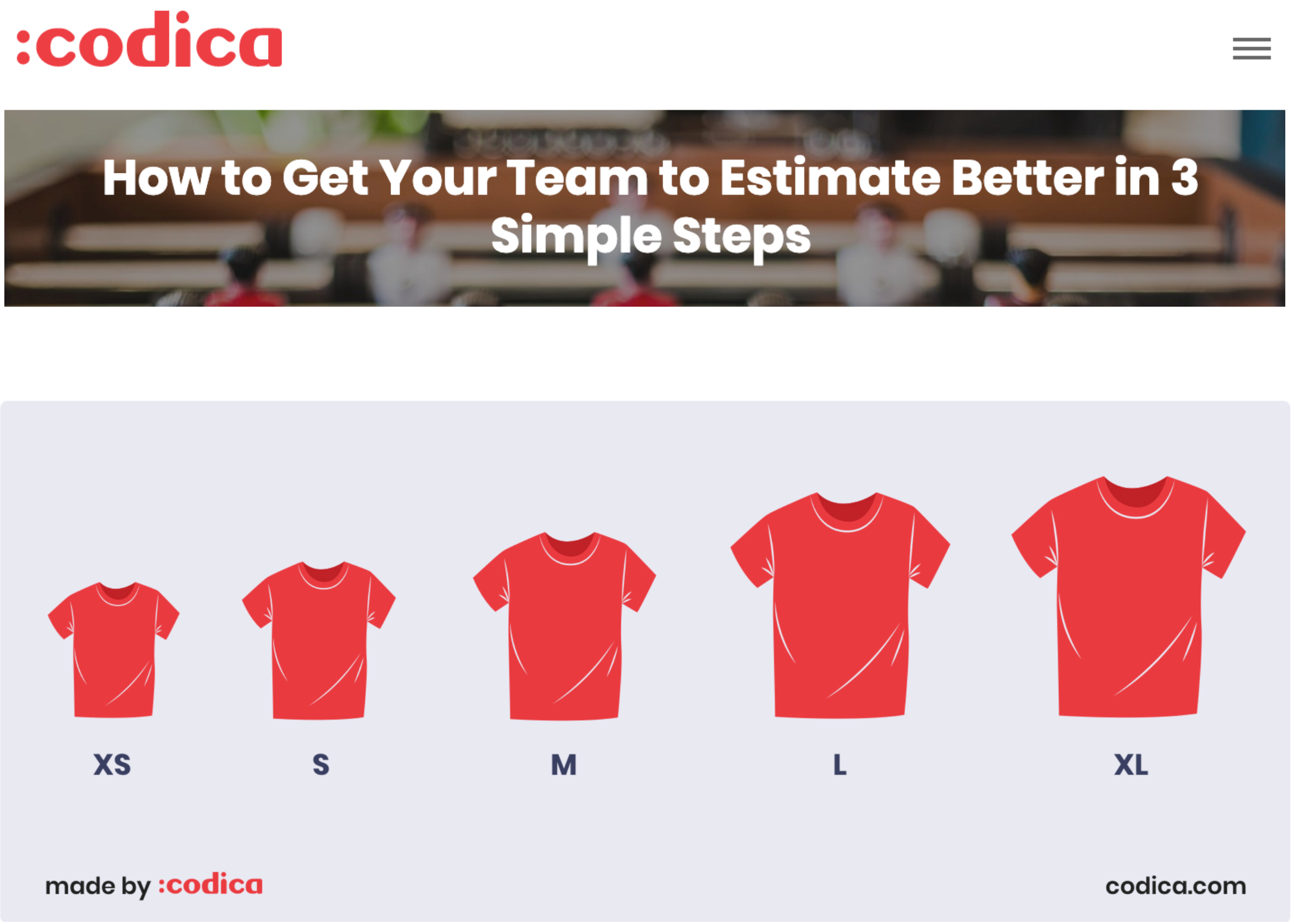


:codica

How to Get Your Team to Estimate Better in 3 Simple Steps

- 2 Types of Projects
  - Projects having an accurate target, technical inquiry and deadlines.
  - Projects having a general idea and no accurate visualization of further development, like products for startups or Time & Material projects.

https://www.codica.com/blog/how-to-get-better-estimates/



How to Get Your Team to Estimate Better in 3 Simple Steps

"It is important to concentrate on the scale of complexity, not the amount of further work."

https://www.codica.com/blog/how-to-get-better-estimates/

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?

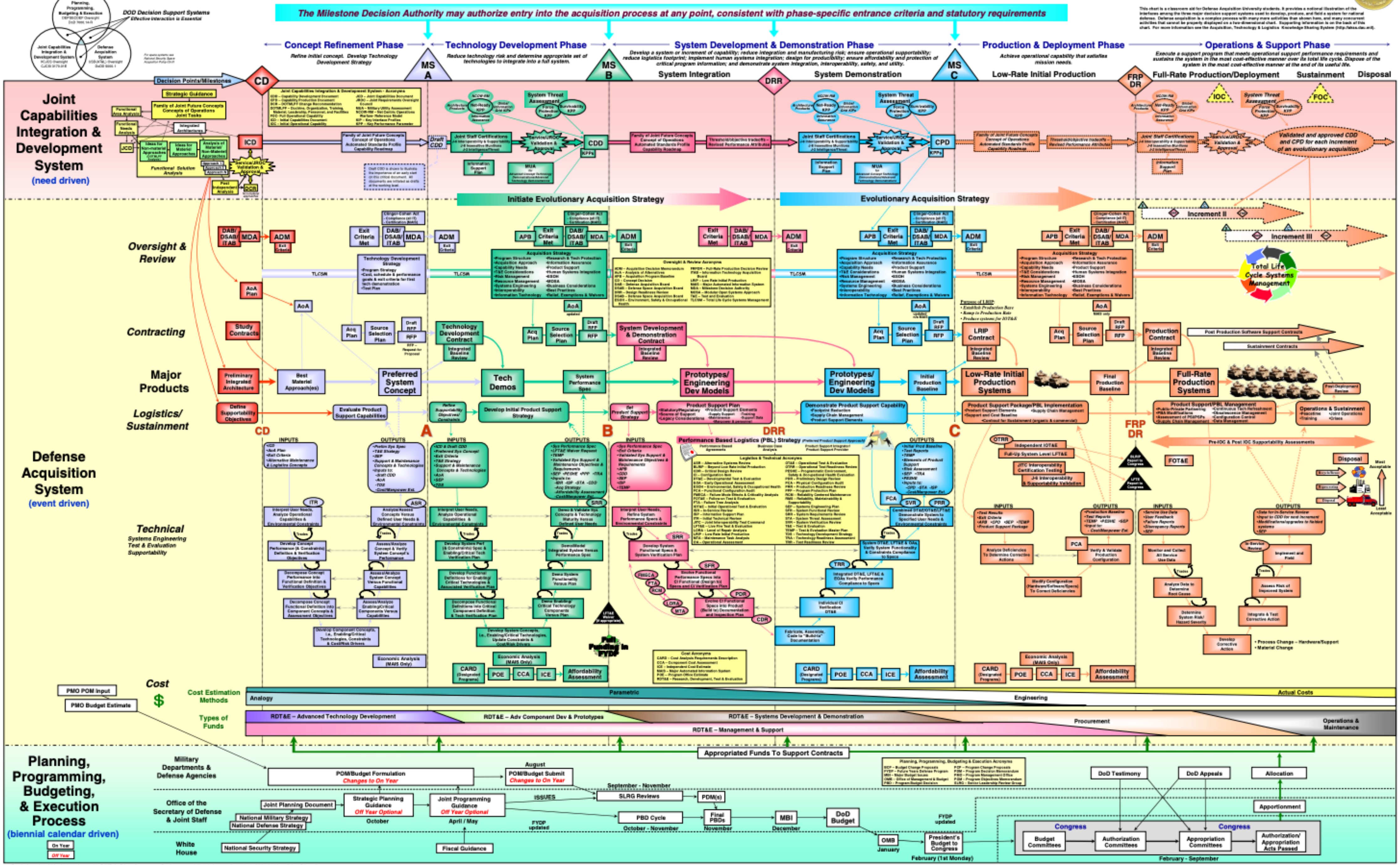
- Developer judgment: x% done
- Lines of code?
- Functionality?
- Quality?

Milestones and Deliverables

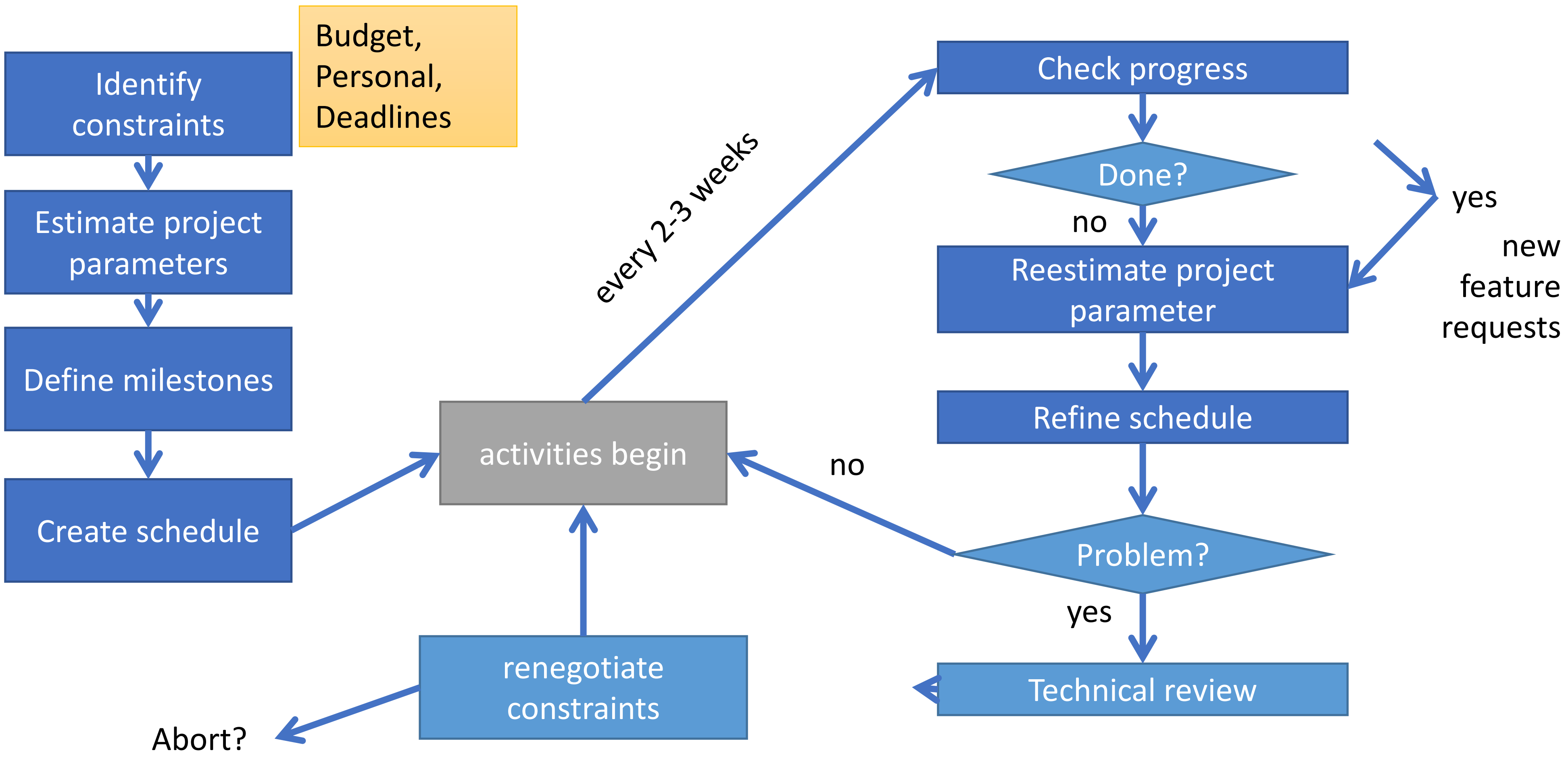
- Making progress observable, especially for software
- Milestone: clear end point of a (sub)tasks
  - For project manager
  - Reports, prototypes, completed subprojects
  - "80% done" not a suitable milestone
- Deliverable: Result for customer
  - Similar to milestone, but for customers
  - Reports, prototypes, completed subsystems

ver. 5.2. August 2005

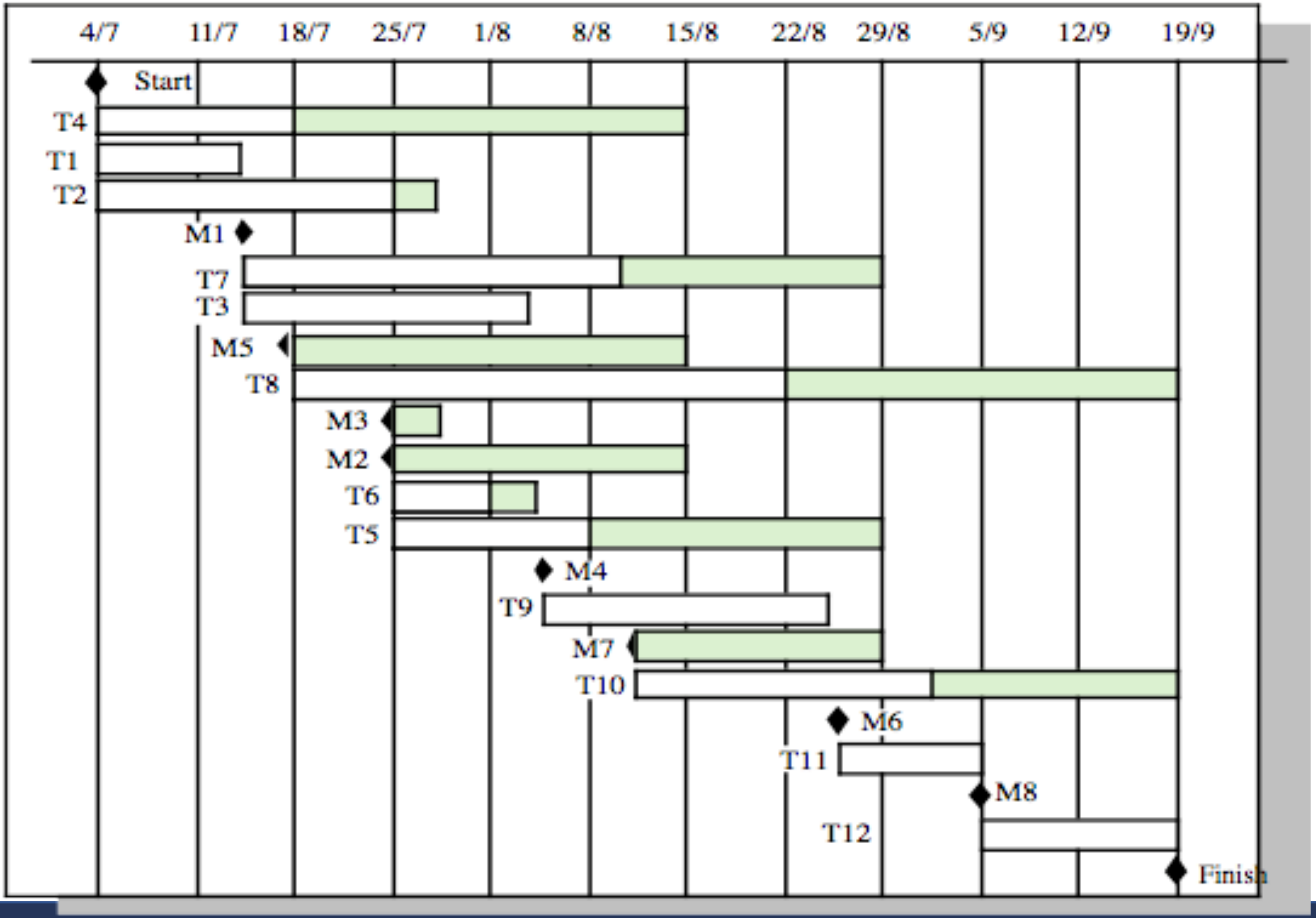
Integrated Defense Acquisition, Technology, & Logistics Life Cycle Management Framework


The Milestone Decision Authority may authorize entry into the acquisition process at any point, consistent with phase-specific entrance criteria and statutory requirements
DOD Decision Support Systems
Effective Interaction is Essential
Planning, Programming, Budgeting & Execution
Joint Capabilities Integration & Development System
Defense Acquisition System
Concept Refinement Phase
Refine initial concept. Develop Technology Development Strategy
Technology Development Phase
Reduce technology risk and determine appropriate set of technologies to integrate into a full system.
System Development & Demonstration Phase
System Integration
System Demonstration
Production & Deployment Phase
Achieve operational capability that satisfies mission needs.
Low-Rate Initial Production
Operations & Support Phase
Full-Rate Production/Deployment
Sustainment
Disposal
Decision Points/Milestones
CD
MS A
MS B
DRR
MS C
FRP DR
IOC
FOC
Joint Capabilities Integration & Development System
(need driven)
ICD
CDD
CPD
Validated and approved CDD and CPD for each increment of an evolutionary acquisition
Initiate Evolutionary Acquisition Strategy
Evolutionary Acquisition Strategy
Increment II
Increment III
Oversight & Review
DAB/DSAB/ITAB
MDA
ADM
AoA
Exit Criteria
APB
TLCSM
Total Life Cycle Systems Management
Contracting
Study Contracts
Technology Development Contract
System Development & Demonstration Contract
LRIP Contract
Production Contract
Post Production Software Support Contracts
Sustainment Contracts
Major Products
Preliminary Integrated Architecture
Best Material Approach(es)
Preferred System Concept
Tech Demos
System Performance Spec
Prototypes/ Engineering Dev Models
Initial Production Baseline
Low-Rate Initial Production Systems
Final Production Baseline
Full-Rate Production Systems
Logistics/ Sustainment
Define Supportability Objectives
Evaluate Product Support Capabilities
Develop Initial Product Support Strategy
Product Support Plan
Demonstrate Product Support Capability
Product Support Package/PBL Implementation
Product Support/PBL Management
Operations & Sustainment
Performance Based Logistics (PBL) Strategy
Defense Acquisition System
(event driven)
Technical
Systems Engineering
Test & Evaluation
Supportability
INPUTS
OUTPUTS
Cost
Cost Estimation Methods
Analogy
Parametric
Engineering
Actual Costs
Types of Funds
RDT&E – Advanced Technology Development
RDT&E – Adv Component Dev & Prototypes
RDT&E – Systems Development & Demonstration
RDT&E – Management & Support
Procurement
Operations & Maintenance
PMO POM Input
PMO Budget Estimate
Appropriated Funds To Support Contracts
Planning, Programming, Budgeting & Execution Process
(biennial calendar driven)
Military Departments & Defense Agencies
Office of the Secretary of Defense & Joint Staff
White House
POM/Budget Formulation
POM/Budget Submit
Strategic Planning Guidance
Joint Programming Guidance
Joint Planning Document
National Military Strategy
National Defense Strategy
National Security Strategy
Fiscal Guidance
SLRG Reviews
PBD Cycle
MBI
DoD Budget
OMB
President's Budget to Congress
Congress
Budget Committees
Authorization Committees
Appropriation Committees
Authorization/ Appropriation Acts Passed
DoD Testimony
DoD Appeals
Allocation
Apportionment

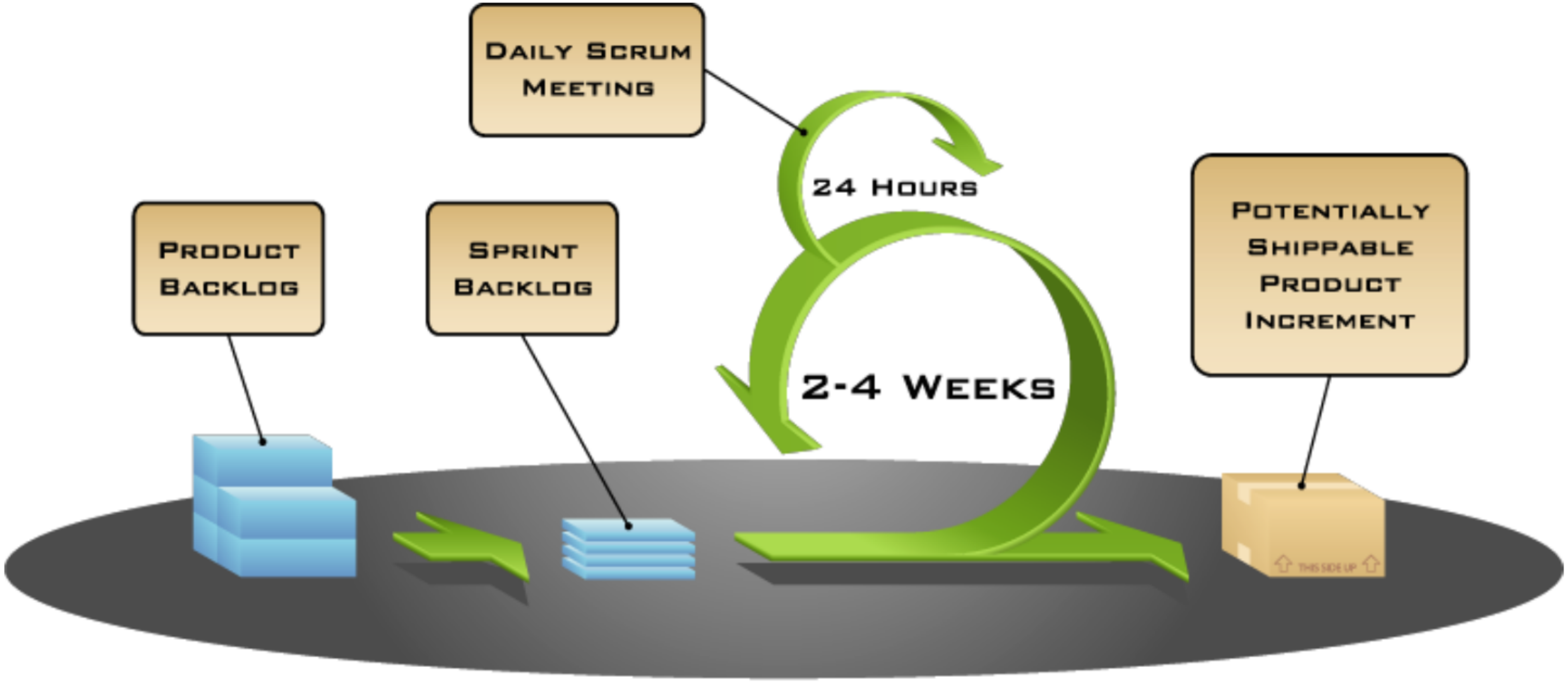
Project Planning



Gantt Diagrams



Brief intro to Scrum



Elements of Scrum

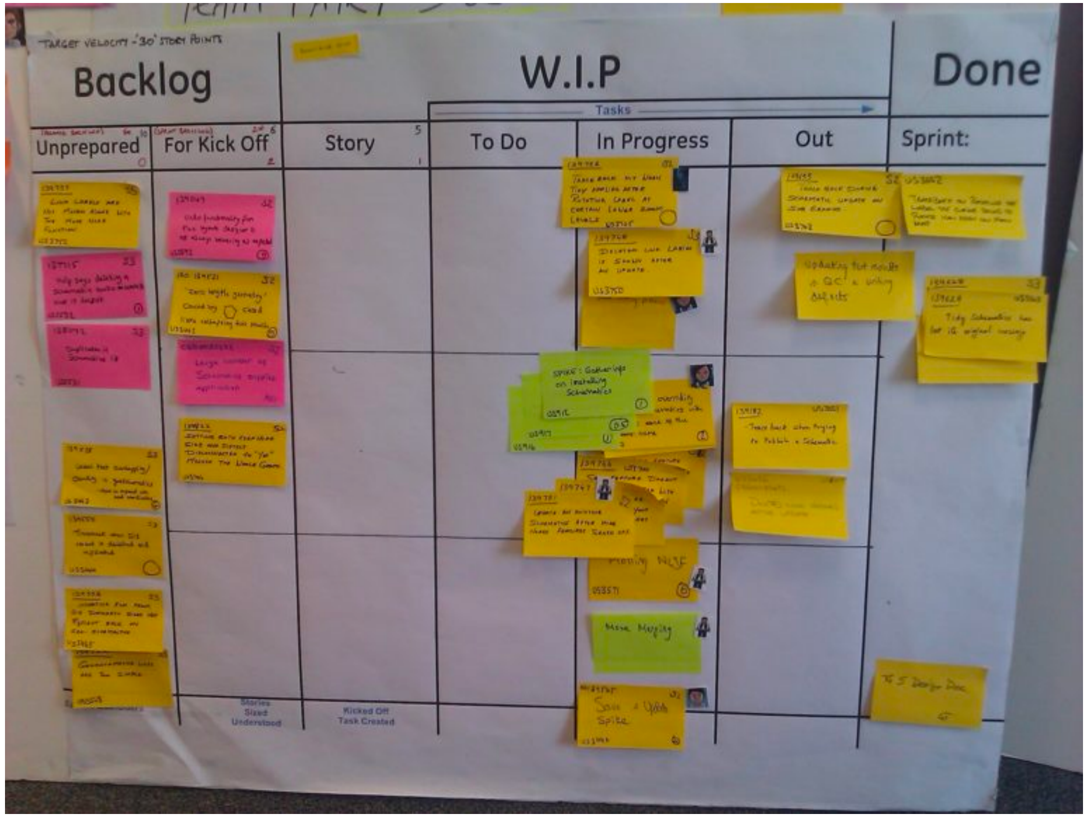
- Products:
  - Product Backlog
  - Sprint Backlog

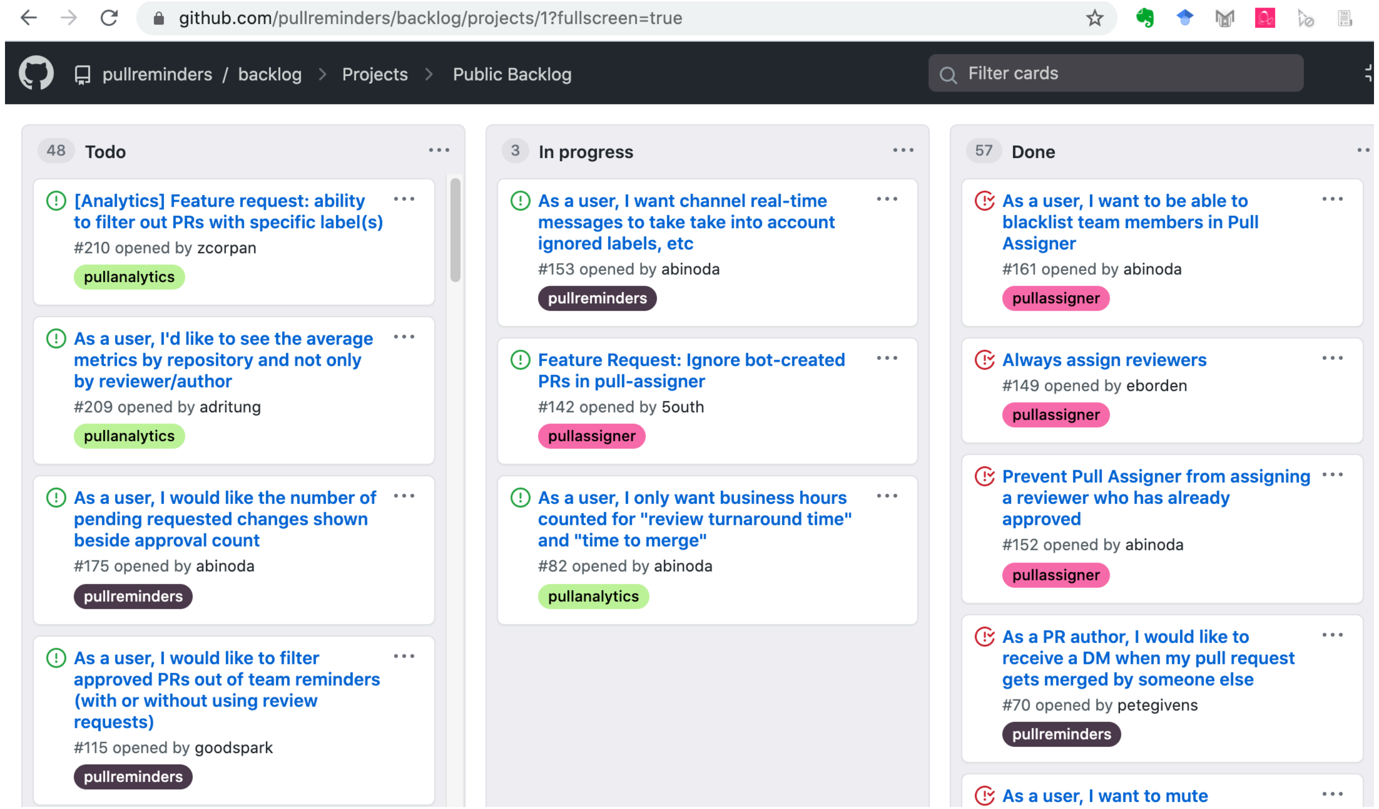
- Process:
  - Sprint Planning Meeting
  - Daily Scrum Meeting
  - Sprint Retrospective
  - Sprint Review Meeting

Product Backlog/Sprint Backlog

- 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

Backlog – information radiators




github.com/pullreminders/backlog/projects/1?fullscreen=true
pullreminders / backlog > Projects > Public Backlog
Filter cards

48 Todo

- [Analytics] Feature request: ability to filter out PRs with specific label(s)
  #210 opened by zcorpan
  pullanalytics
- As a user, I'd like to see the average metrics by repository and not only by reviewer/author
  #209 opened by adritung
  pullanalytics
- As a user, I would like the number of pending requested changes shown beside approval count
  #175 opened by abinoda
  pullreminders
- As a user, I would like to filter approved PRs out of team reminders (with or without using review requests)
  #115 opened by goodspark
  pullreminders

3 In progress

- As a user, I want channel real-time messages to take take into account ignored labels, etc
  #153 opened by abinoda
  pullreminders
- Feature Request: Ignore bot-created PRs in pull-assigner
  #142 opened by 5outh
  pullassigner
- As a user, I only want business hours counted for "review turnaround time" and "time to merge"
  #82 opened by abinoda
  pullanalytics

57 Done

- As a user, I want to be able to blacklist team members in Pull Assigner
  #161 opened by abinoda
  pullassigner
- Always assign reviewers
  #149 opened by eborden
  pullassigner
- Prevent Pull Assigner from assigning a reviewer who has already approved
  #152 opened by abinoda
  pullassigner
- As a PR author, I would like to receive a DM when my pull request gets merged by someone else
  #70 opened by petegivens
  pullreminders
- As a user, I want to mute

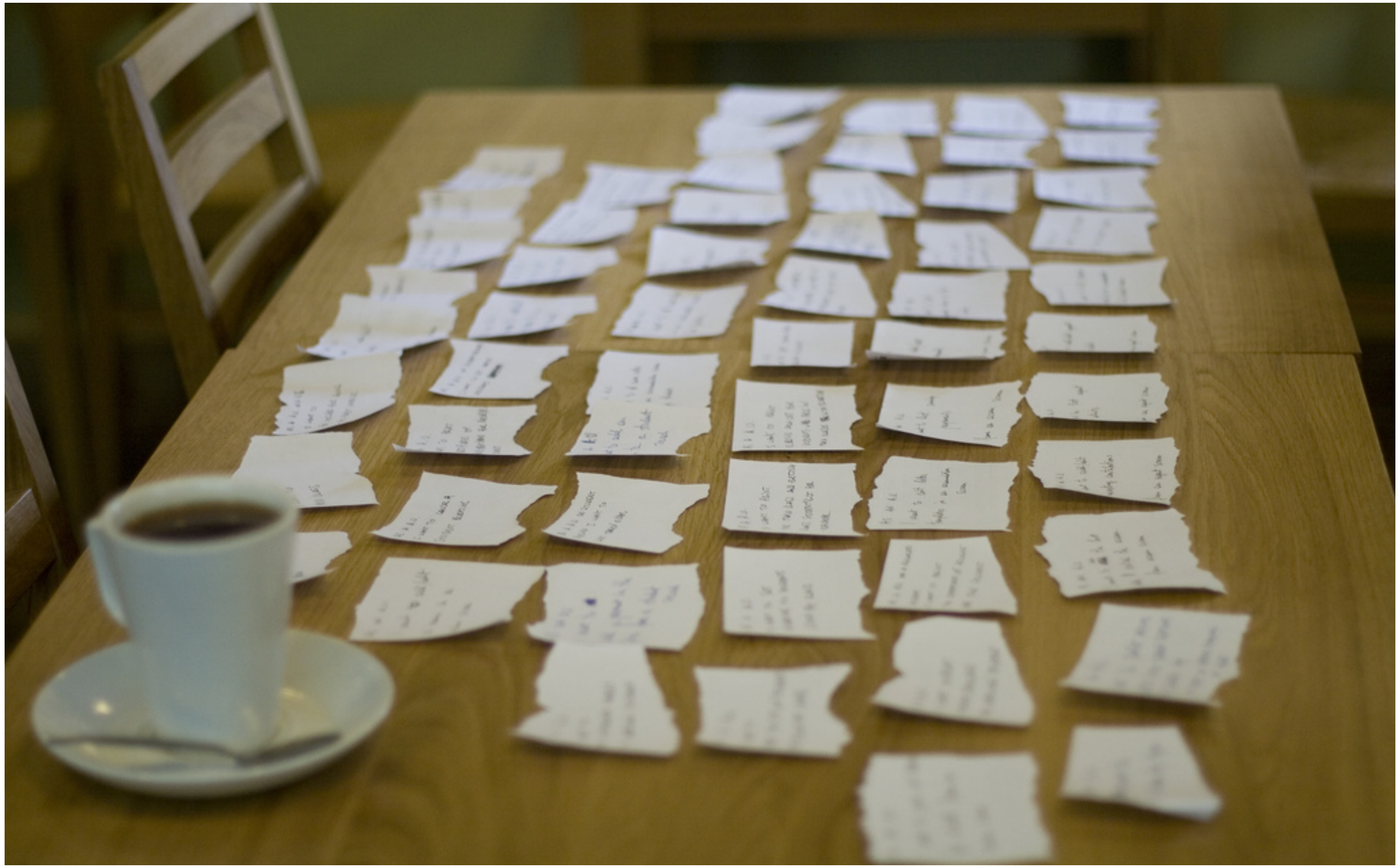
Scrum meetings

- 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

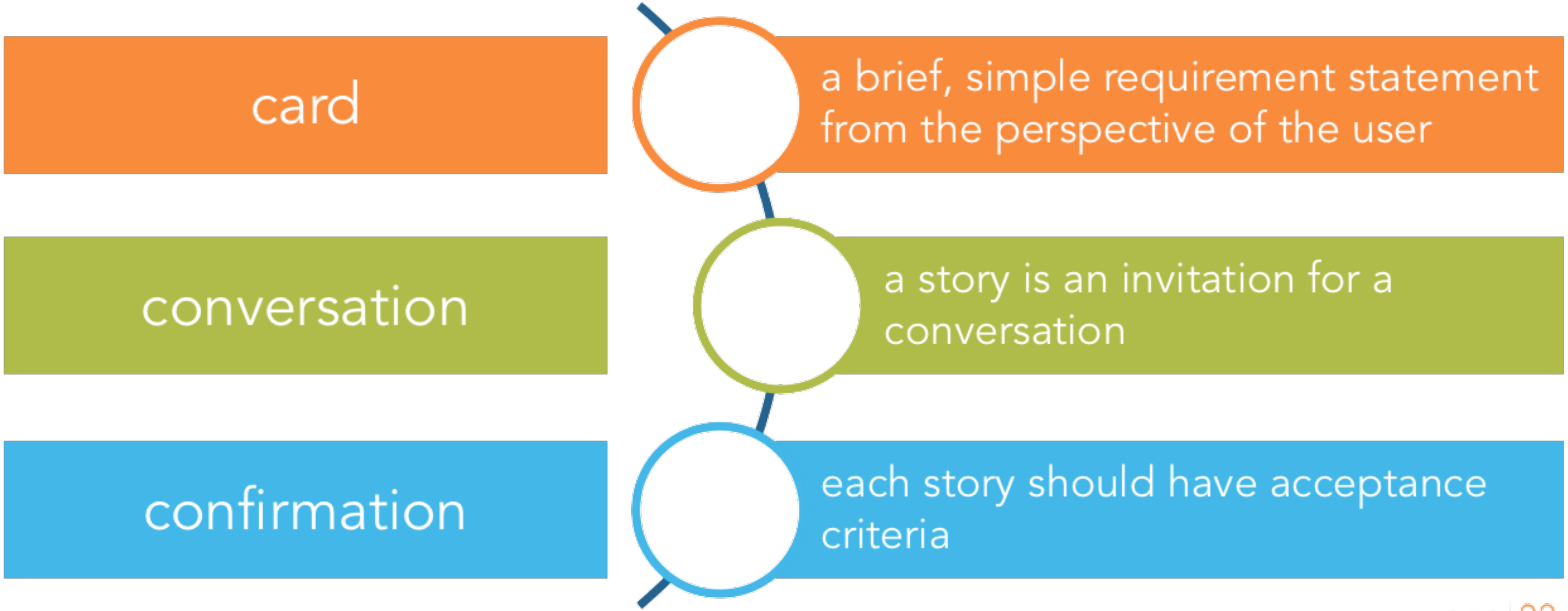
Planning

- Time estimation
- Tools
- Agile
- User stories

User Stories



User Stories



card

a brief, simple requirement statement from the perspective of the user

conversation

a story is an invitation for a conversation

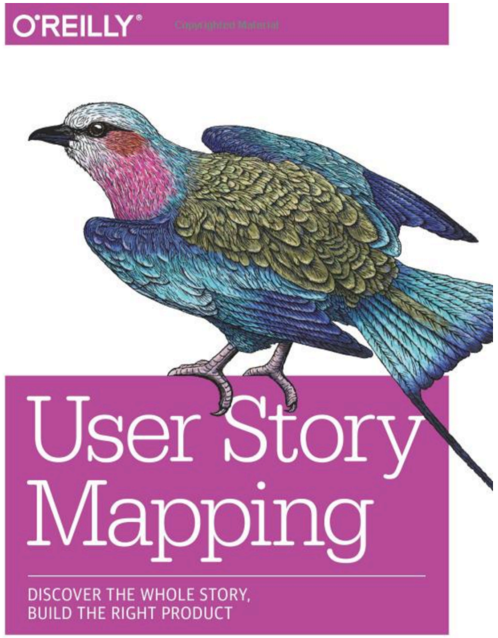
confirmation

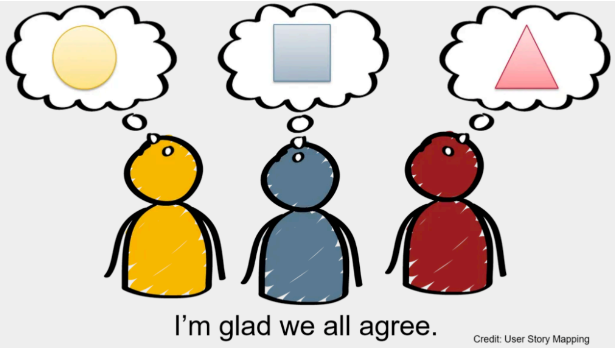
each story should have acceptance criteria

one 80

The conversation

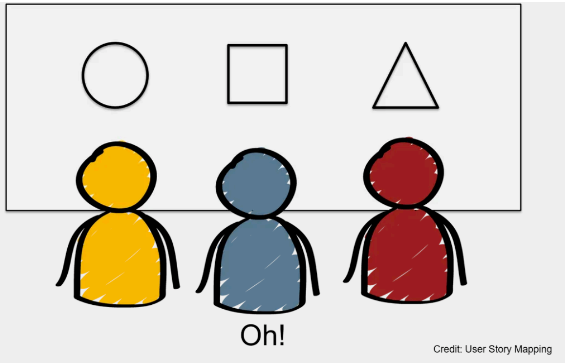
- An open dialog between everyone working on the project and the client
- Split up Epic Stories if needed





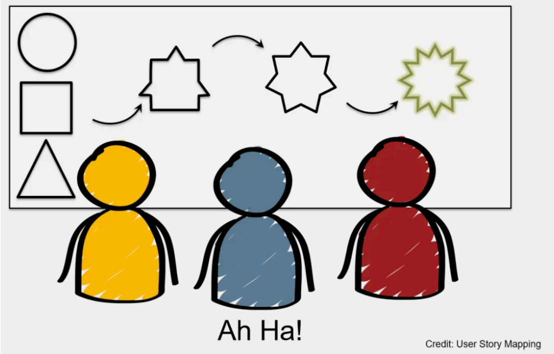
Credit: User Story Mapping

User Story (Not Shared Understanding)



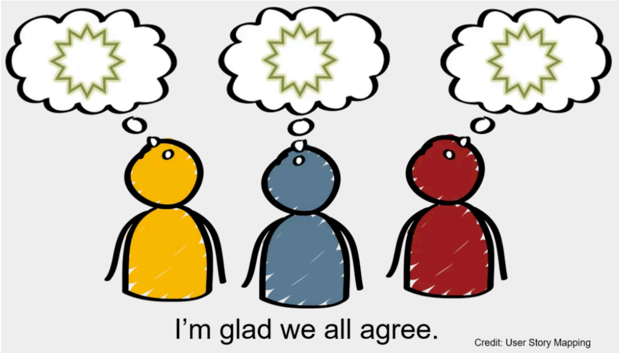
Credit: User Story Mapping

User Story (Explain Understanding)



Credit: User Story Mapping

User Story (Conversation towards understanding)



Credit: User Story Mapping

User Story (Shared Understanding)

The Card

As a < type of user >,
I want < some goal >
so that < some reason >.

Who (User)

This should describe a fairly detailed user. It is not sufficient to just say “user.” Strive towards something like “broke college student on a mobile device user.” When we express the **who** with more detail we are able to better empathize with that particular user, determine the best solution and uncover implicit needs.

What (Goal)

The goal or action the user intends to take.

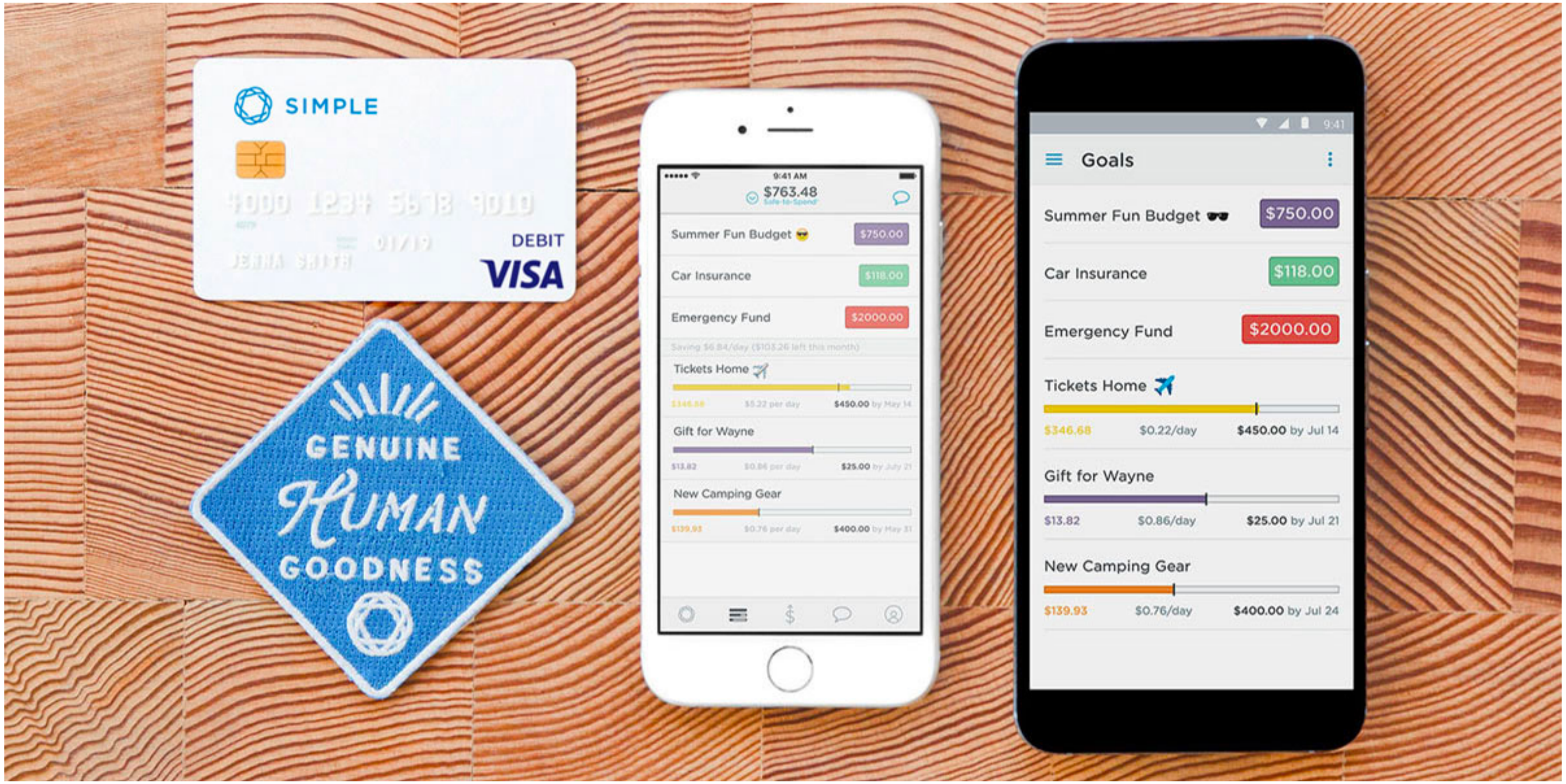
Why (Benefit)

Expressing the benefit to the user is by far the most important in my experience. Some of the most creative and inexpensive solutions come from the developers and users understanding why they are building something.

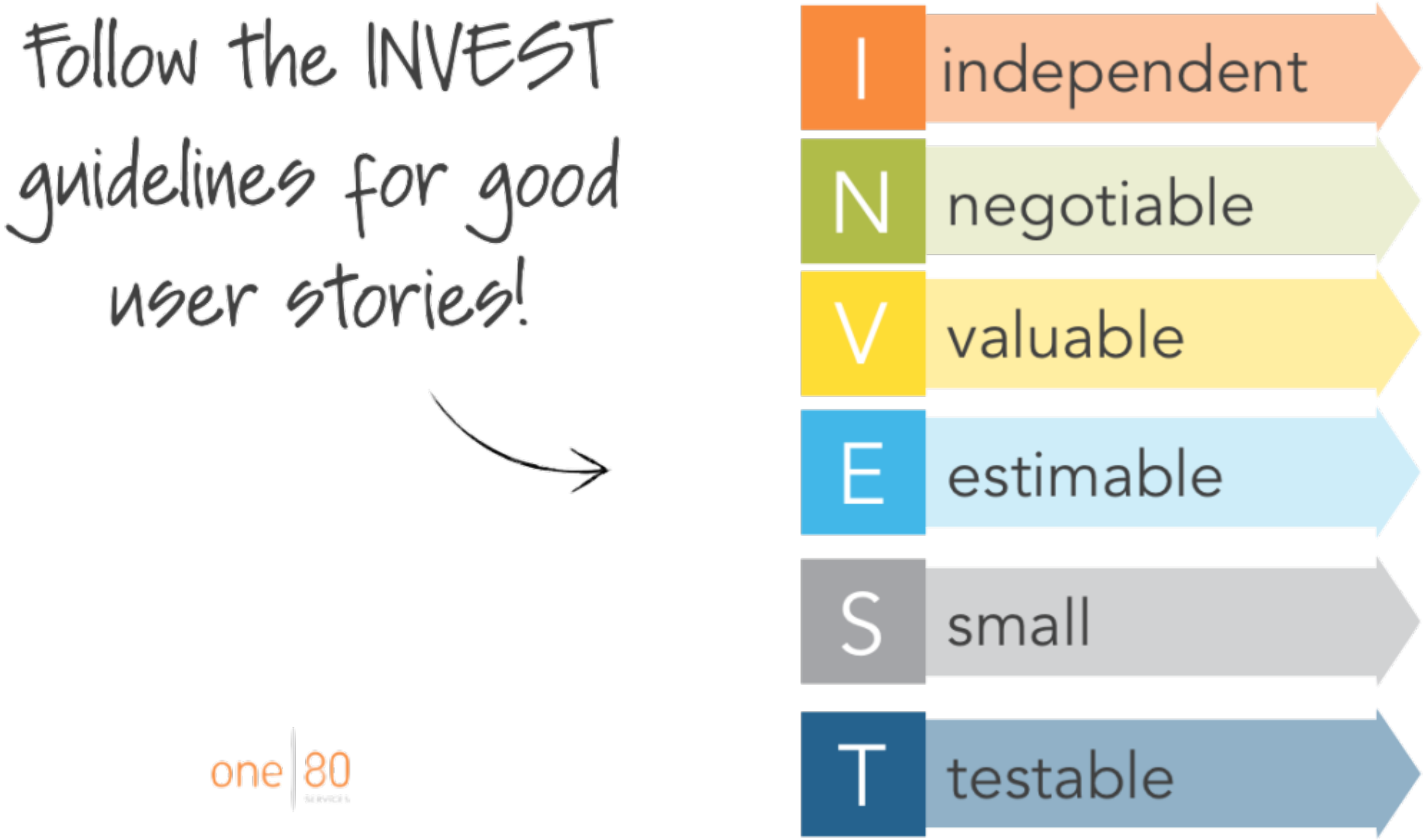
The Confirmation

- A confirmation criteria that will show when the task is completed
- Could be automated or manual

Exercise


SIMPLE
DEBIT
VISA
GENUINE
HUMAN
GOODNESS
$763.48
Goals
Summer Fun Budget
$750.00
Car Insurance
$118.00
Emergency Fund
$2000.00
Tickets Home
$346.68
$0.22/day
$450.00 by Jul 14
Gift for Wayne
$13.82
$0.86/day
$25.00 by Jul 21
New Camping Gear
$139.93
$0.76/day
$400.00 by Jul 24

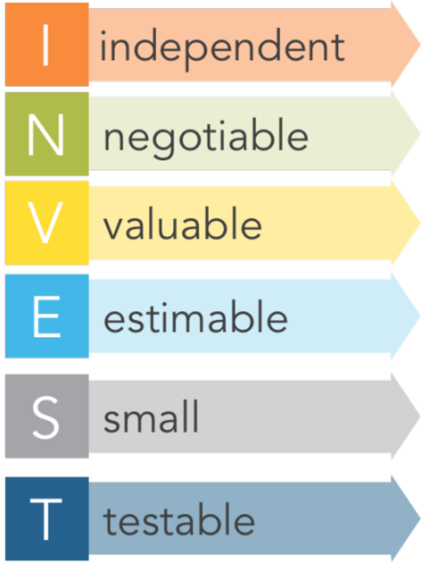
How to evaluate user study?



Follow the INVEST guidelines for good user stories!

- I independent
- N negotiable
- V valuable
- E estimable
- S small
- T testable

I independent



I independent

N negotiable

V valuable

E estimable

S small

T testable

- Schedule in any order.
- Not overlapping in concept
- Not always possible

N negotiable

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

I independent

N negotiable

V valuable

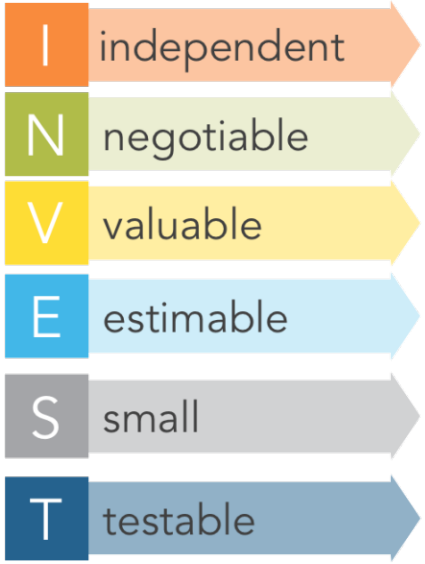
E estimable

S small

T testable

V valuable

- This story needs to have value to someone (hopefully the customer)
- Especially relevant to splitting up issues



I independent

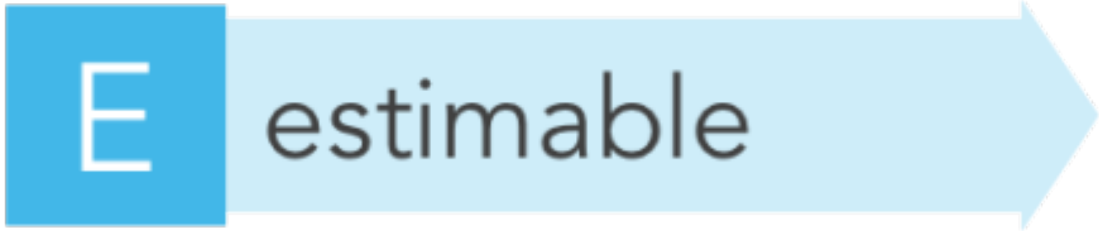
N negotiable

V valuable

E estimable

S small

T testable



I
independent
N
negotiable
V
valuable
E
estimable
S
small
T
testable

- Helps keep the size small
- Ensure we negotiated correctly
- “Plans are nothing, planning is everything” -Dwight D. Eisenhower

S small

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

I independent
N negotiable
V valuable
E estimable
S small
T testable

T testable

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

I independent

N negotiable

V valuable

E estimable

S small

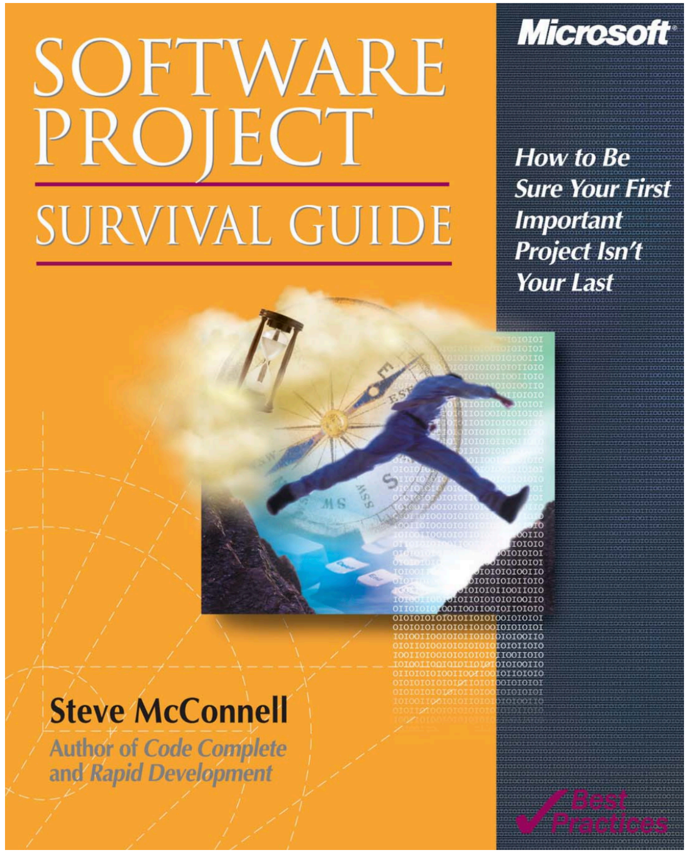
T testable

Planning

- Time estimation
- Tools
- Agile
- User stories

Further Reading

- McConnell. Software Project Survival Guide. Microsoft Press 1998, Chapter 3 (link)
- Sommerville. Software Engineering. 8th Edition. Addison-Wesley 2007. Chapters 5 "Project Planning" and 26 "Software Cost Estimation"


SOFTWARE PROJECT
SURVIVAL GUIDE
Microsoft
How to Be Sure Your First Important Project Isn't Your Last
Steve McConnell
Author of Code Complete and Rapid Development
Best Practices

Teamwork (Student Teams)

More on teams in real projects in the course

Expectation

- Meet initially and then regularly
- Review team policy
- Divide work and integrate
- Establish a process
- **Set and document clear responsibilities and expectations**
  - Possible Roles: Coordinator, Scribe, Checker, Monitor
  - Rotate roles every assignment
- Every team member should understand the entire solution

Dealing with problems

- Openly report even minor team issues in individual part of the milestone report
- In-class discussions and case studies
- Additional material throughout semester
- We will attend one team meeting

Planning and In-Team Communication

- Asana, Trello, Microsoft Project, …
- Github Wiki, Google docs, …
- Email, Slack, Facebook groups, …

Meet your teammates!