

CEN 5016: Software Engineering

Fall 2024



University of
Central Florida

Dr. Kevin Moran

Week 3 - Class 11: Software Teams & Communication





- *Team-forming this week - Due Thursday, Sept 12th EoD!!*
 - Teams of 3 students
 - See Ed Discussions Post
- *Assignment 2 Coming Soon!*
 - Probably End of the Day today/tomorrow morning
 - May adjust due date to give more time

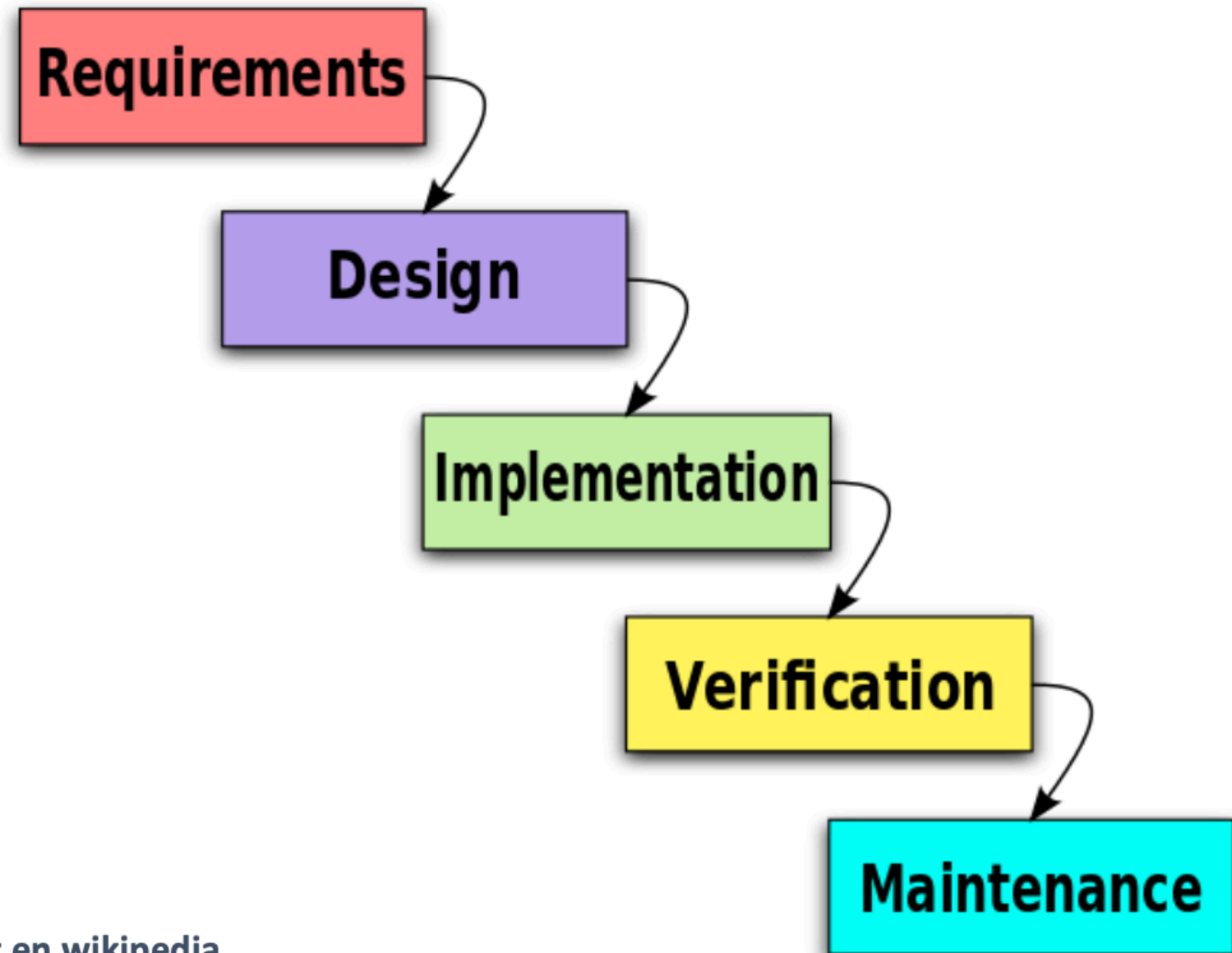
Project Planning & Agile Development



Processes

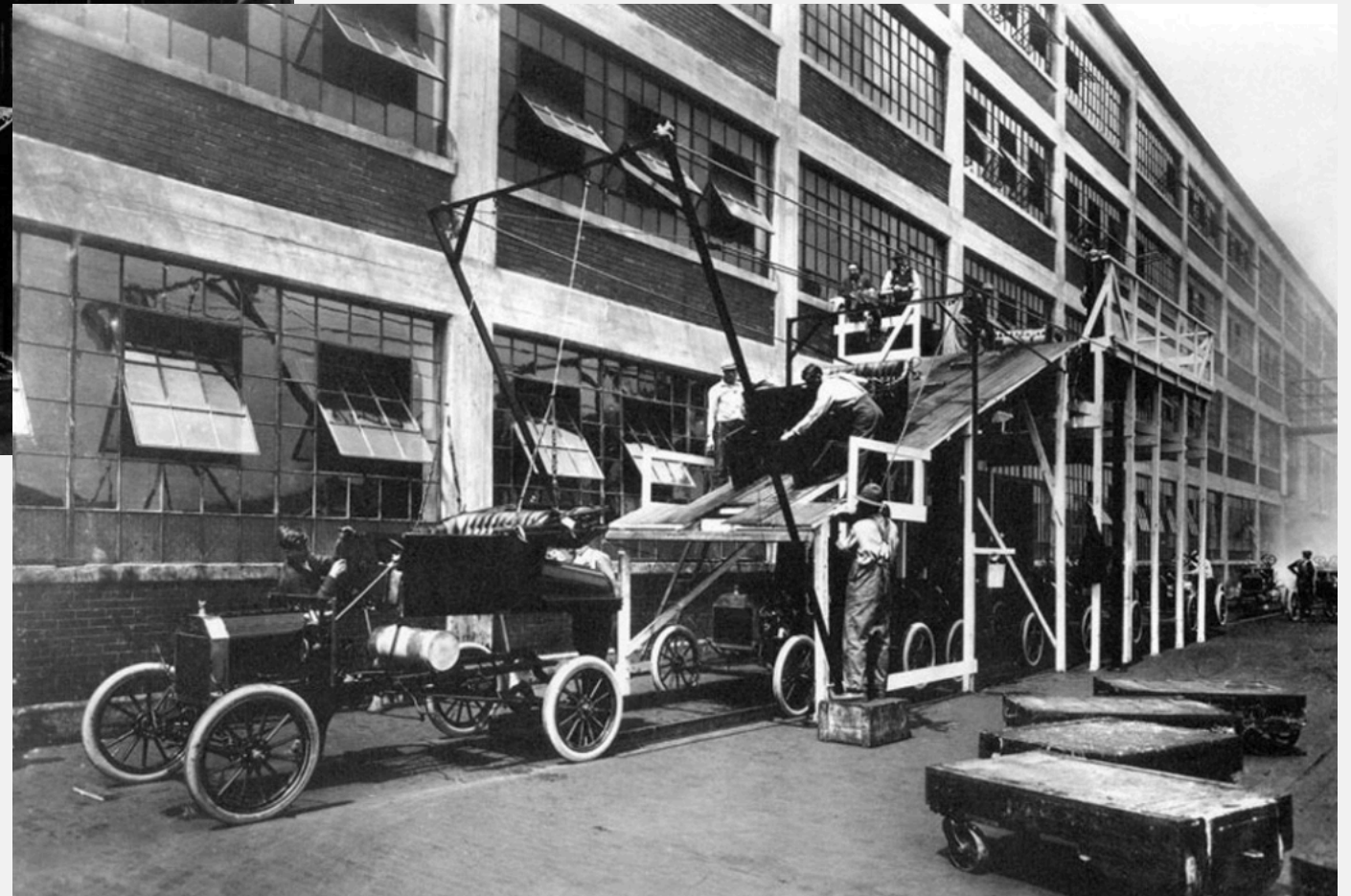


Waterfall was the OG Software Process



Waterfall diagram CC-BY 3.0 [Paulsmith99](#) at [en.wikipedia](#)

Akin to Processes Pioneered in Auto Manufacturing by Ford



LEAN Production Adapts to Variable Demand

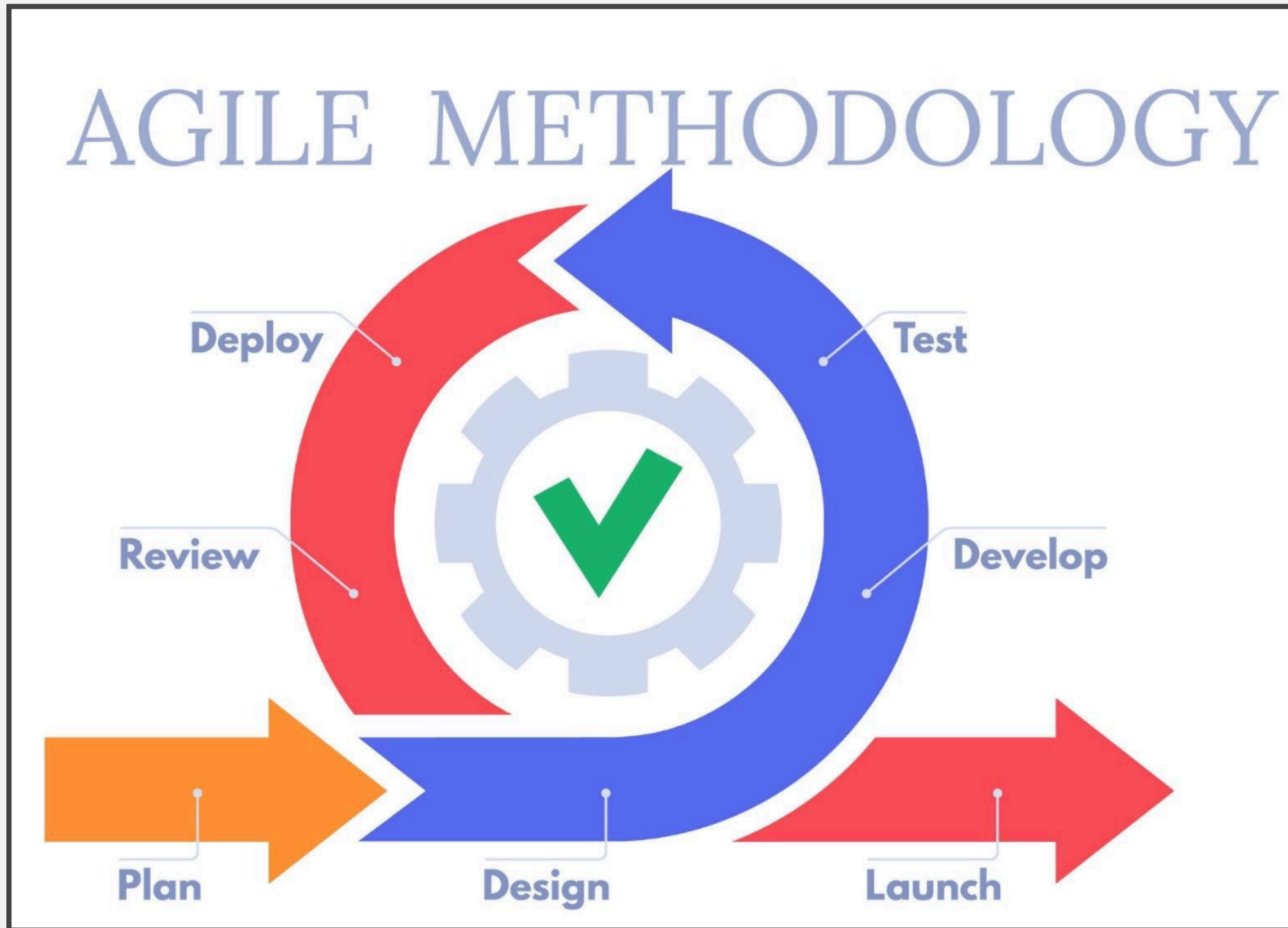


- Toyota Production System (TPS)
 - Build only what is needed, only when it is needed.
 - Use the “pull” system to avoid overproduction (Kanban)
 - Stop to fix problems, to get quality right from the start (Jidoka)
 - Workers are multi-skilled and understand the whole process; take ownership
- Lots of recent software buzzwords build on these ideas
 - Just-in-time, DevOps, Shift-Left



Taiichi Ohno

Now, Most Teams use some form of Agile Methods





Agile software development

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

Manifesto for Agile Software Development (2001)



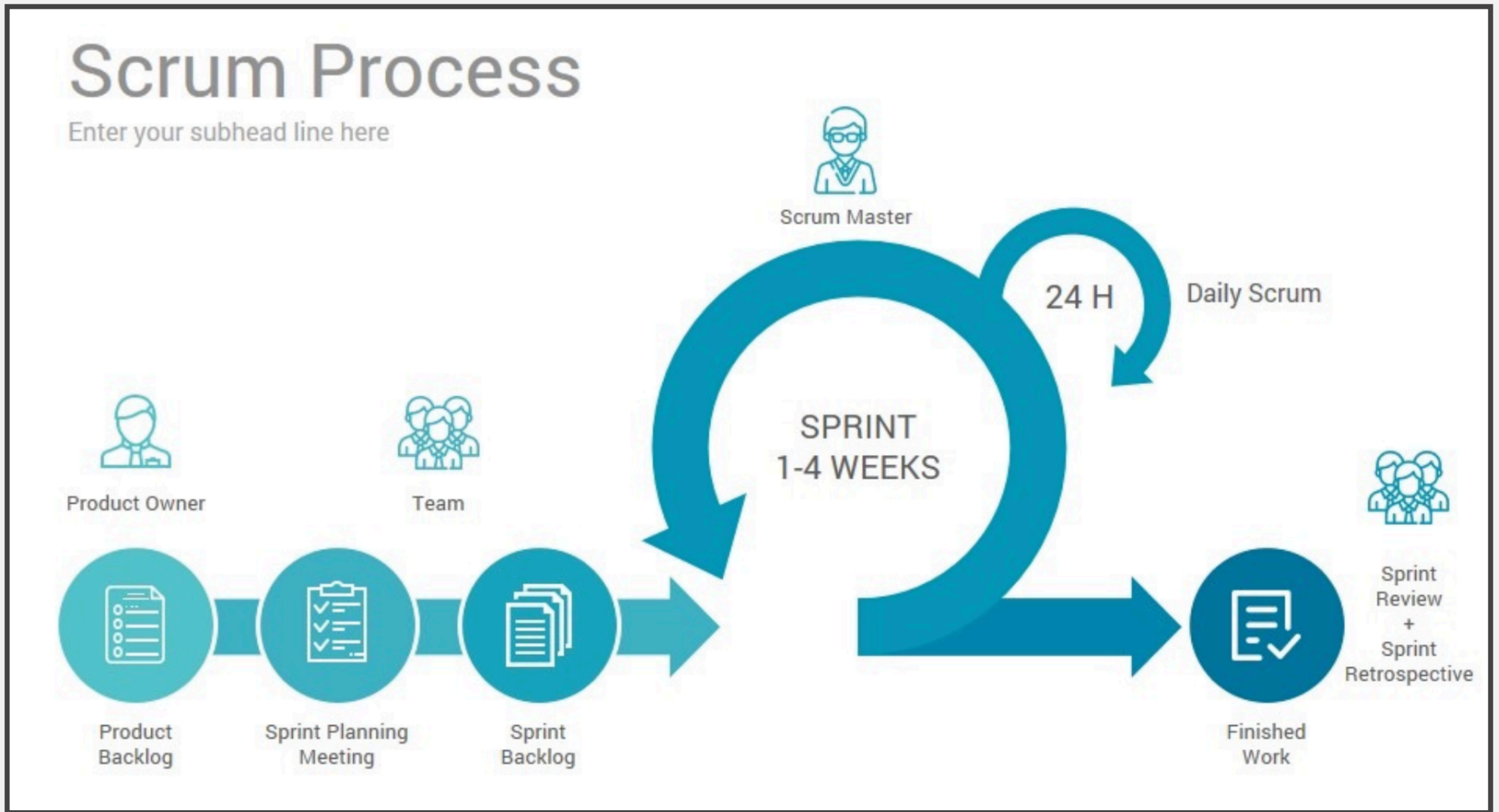
Core Concepts in Agile



Core concepts	Facets of agility in the literature
(1) Incremental design and iterative development	<i><u>Anticipating change by working iteratively</u> – in short, delivery cycles – and thereby reducing the scope of the product to small increments to create opportunities for inspection; <u>Creating change through incremental software design in response to change from what has been learned</u></i>
(2) Inspect and adapt cycles	<i><u>Anticipating change by instituting ceremonies for inspecting and adapting</u> (i.e., <u>learning from and creating change in response to discovered changes</u>) the product increment (e.g., simplifying – “just enough” – design, testing software frequently) and the development process (e.g., updating work statuses, reevaluating team processes, reprioritizing requirements)</i>
(3) Working cooperatively/ Collaboratively/In close communication	<i><u>Anticipating change through recognising and predicting changes in one's environment</u>; <u>Creating change as a team by working together to respond to change from what has been learned collectively</u></i>
(4) Continuous customer involvement	<i>In addition to the cell above, centralising user requirements changes by working together with the customer to collectively identify and <u>respond to change early through close customer involvement</u></i>

Scrum







- 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



Upstream issues to track 4

<https://github.com/git-lfs/git-lfs/issues/2627>

Add card Cancel

- Git LFS 2.3.1 seems to break Windows**
#2627 opened by larsxschneider
- docker build limit io disk**
#35012 opened by sztwiorok
area/builder kind/feature
Reference to moby/moby
- repl: allow `await` in REPL**
#13209 opened by benjaminjr
cli feature request promises repl
Reference to nodejs/node

New things to check out 4

- Implement split diffs**
1 of 6
#866 opened by BinaryMuse
work-in-progress
Reference to atom/github
- Change license and remove references to PATENTS**
#10804 opened by sophiebits
CLA Signed
Reference to facebook/react
- "Clone in Desktop" flow now recognizes gists**
#2939 opened by shiftkey
ready-for-review
Reference to desktop/desktop

Fixes to upgrade for 4

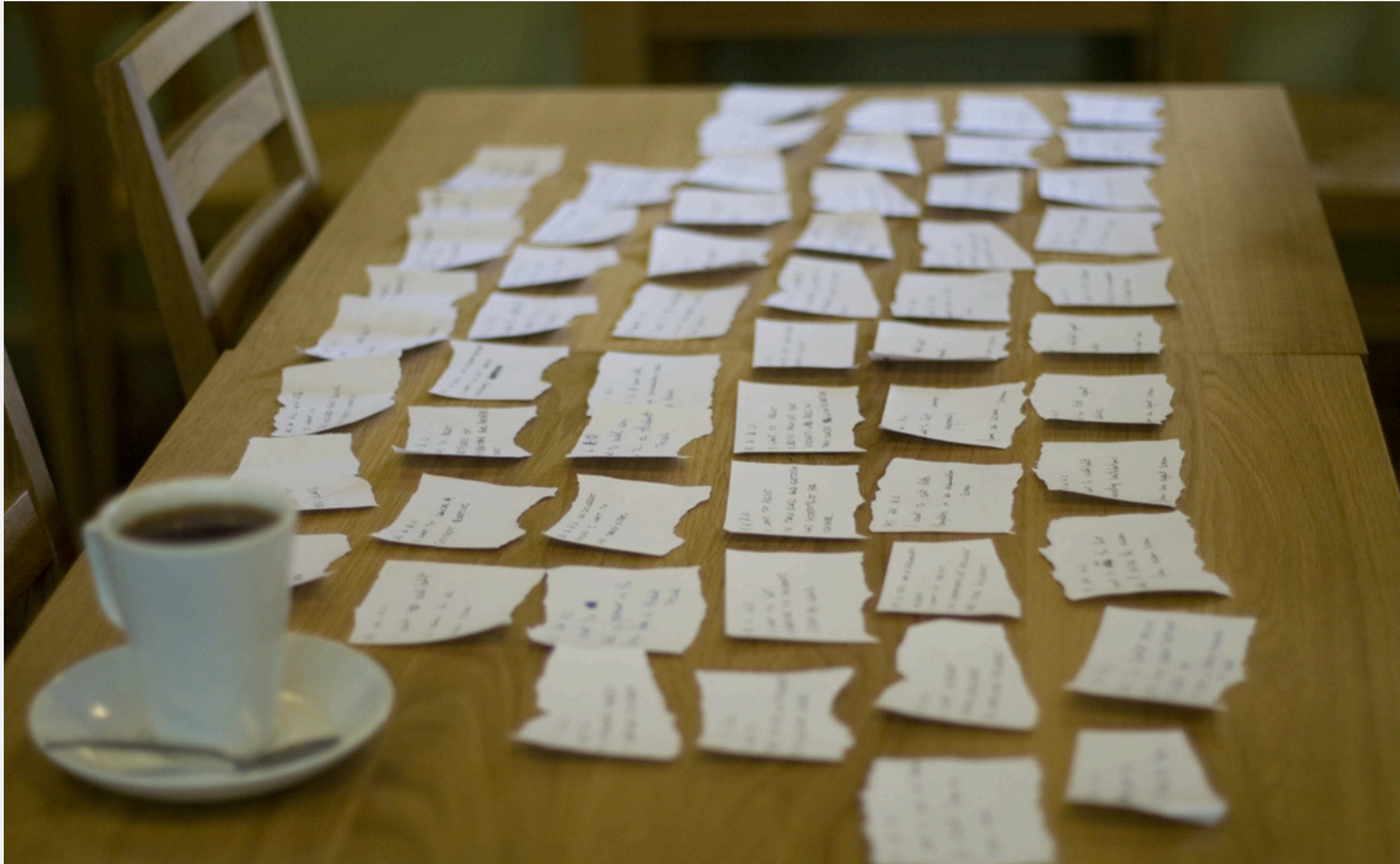
- #3311 opened by kdzwinel
audit
Reference to GoogleChrome/lighthouse
- Error: Undefined variable: "\$h1-size-mobile"**
#229 opened by kaelig
Reference to primer/primer-css
- util: use faster -0 check**
3 of 3
#15726 opened by mscdex
performance util
Reference to nodejs/node
- Git LFS 2.3.1 seems to break Windows**
#2627 opened by larsxschneider

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

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



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



- What must a developer do to implement this user story?



- 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?

How to Evaluate a User Story



Follow the INVEST guidelines for good user stories!

→

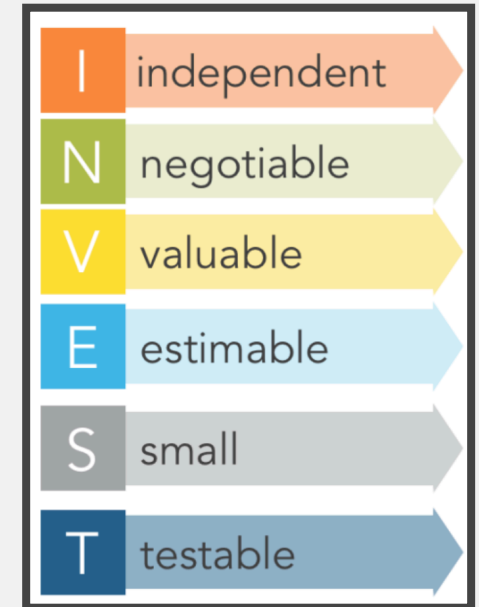
- I independent
- N negotiable
- V valuable
- E estimable
- S small
- T testable

one | 80
SERVICES

Independent



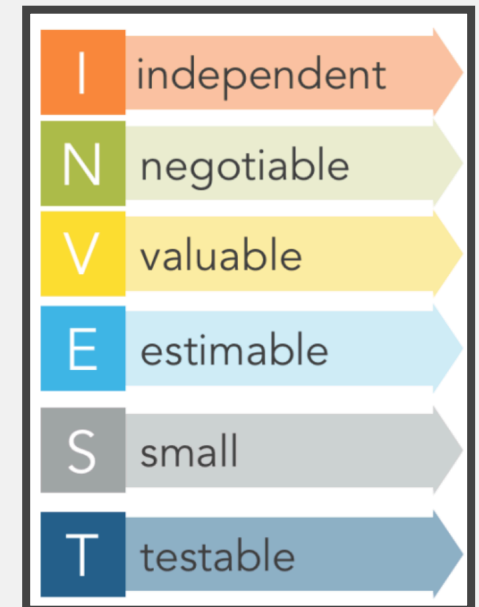
- Schedule in any order.
- Not overlapping in concept.
- Not always possible.



Negotiable

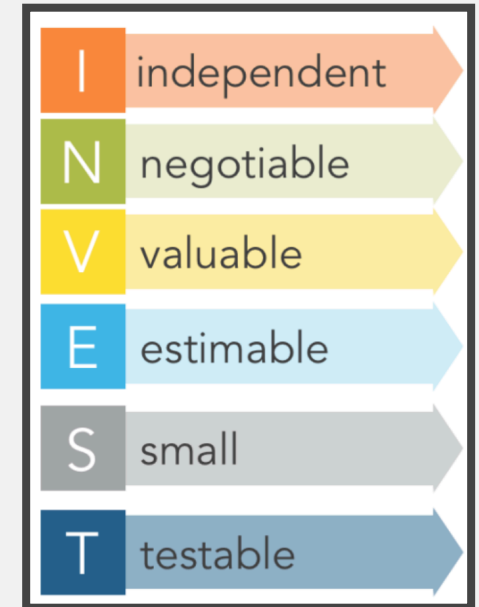


- Details to be negotiated during development.
- A good story captures the essence, not the details.





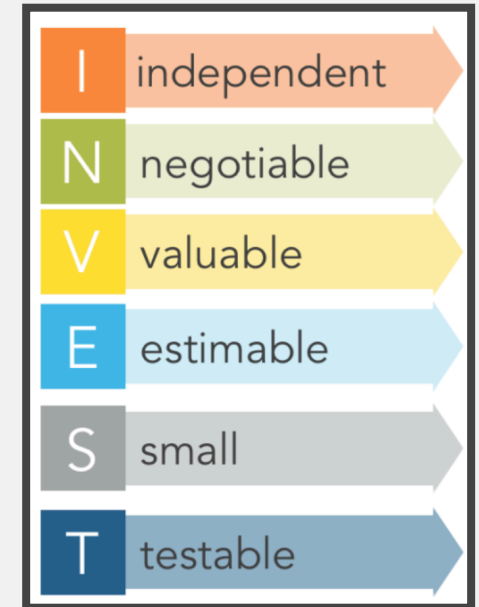
- This story needs to have value to someone (hopefully the customer).
- Especially relevant to splitting up issues.



Estimable



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



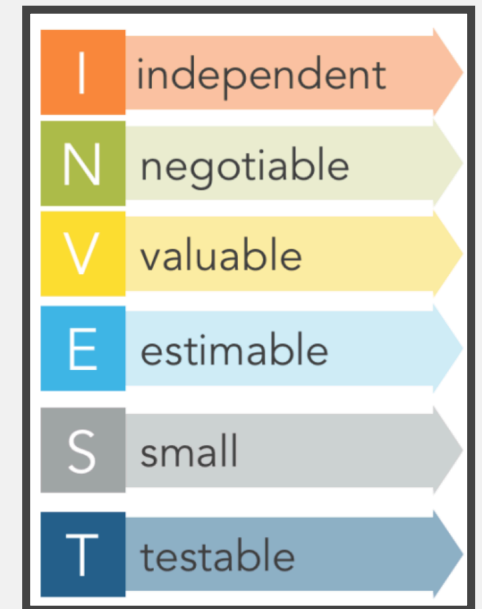


- Can be written on a 3x5 card.
- At most two person-weeks of work.
- Too big === unable to estimate

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



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



Software Teams & Communication



Learning Goals



- Describe the pros and cons of working as a team
- Recognize the importance of communication in collaboration
- Recognize the need of having multiple communication channels
- Select an appropriate communication tool for a given communication goal
- Ask technical questions effectively
- Write clear and specific Github issues, pull requests, and comments

We all Work in a Team



Bubble Sort

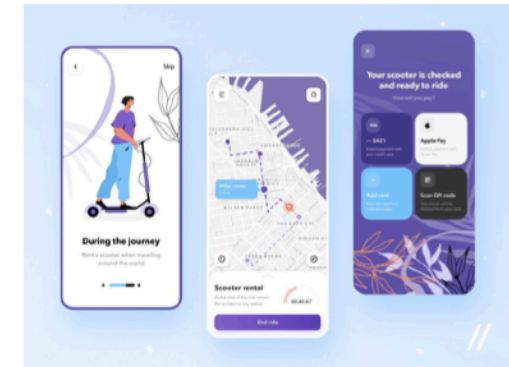
First pass	6	2	8	4	10
Next pass	2	6	8	4	10
Next pass	2	6	4	8	10
	2	4	6	8	10

Review complete

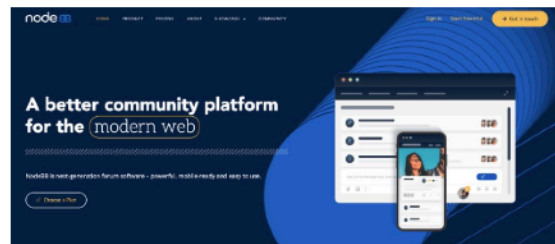
Bubble Sort



Monopoly Game



Scooter App

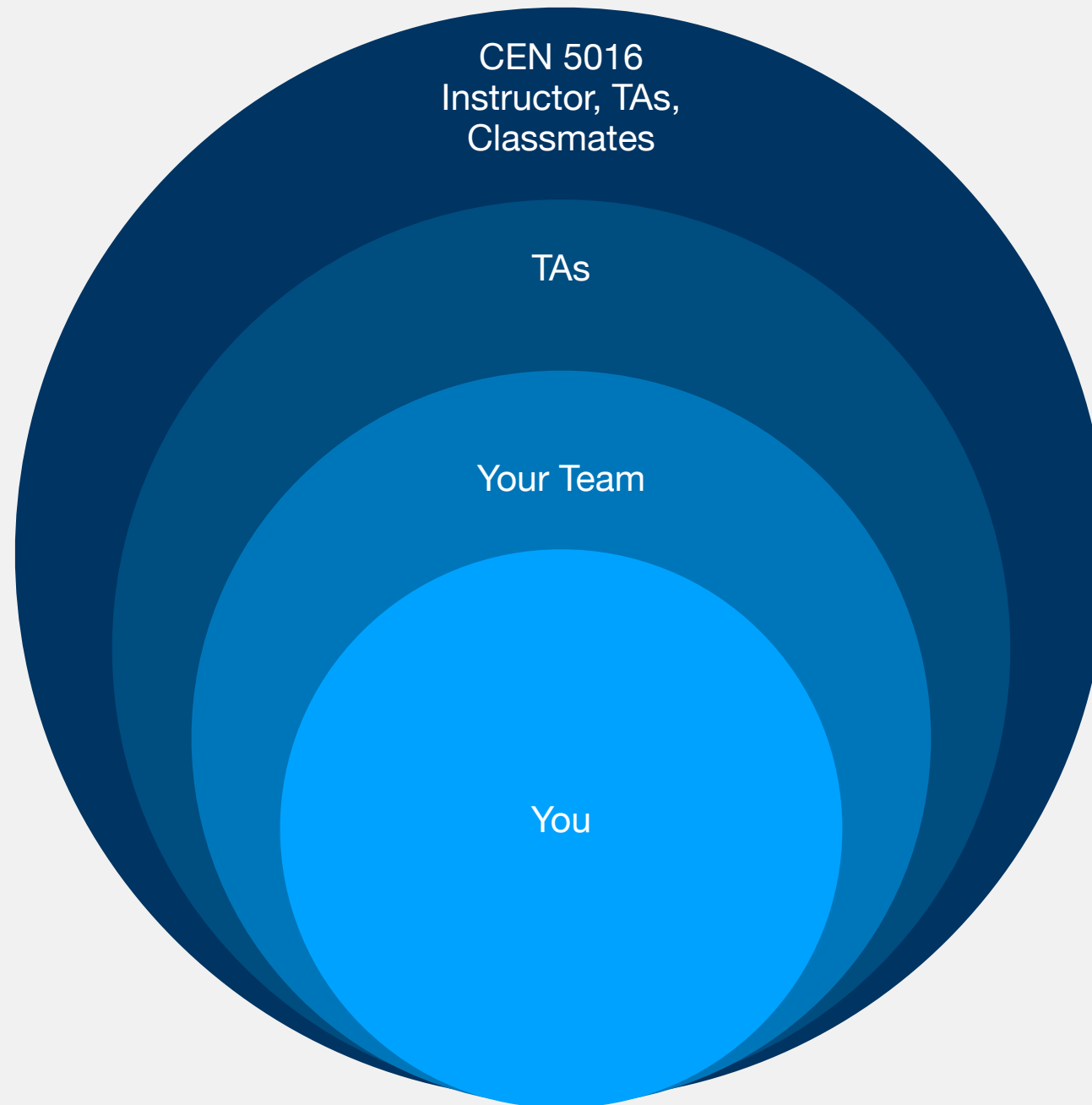


NodeBB



Autonomous Vehicle

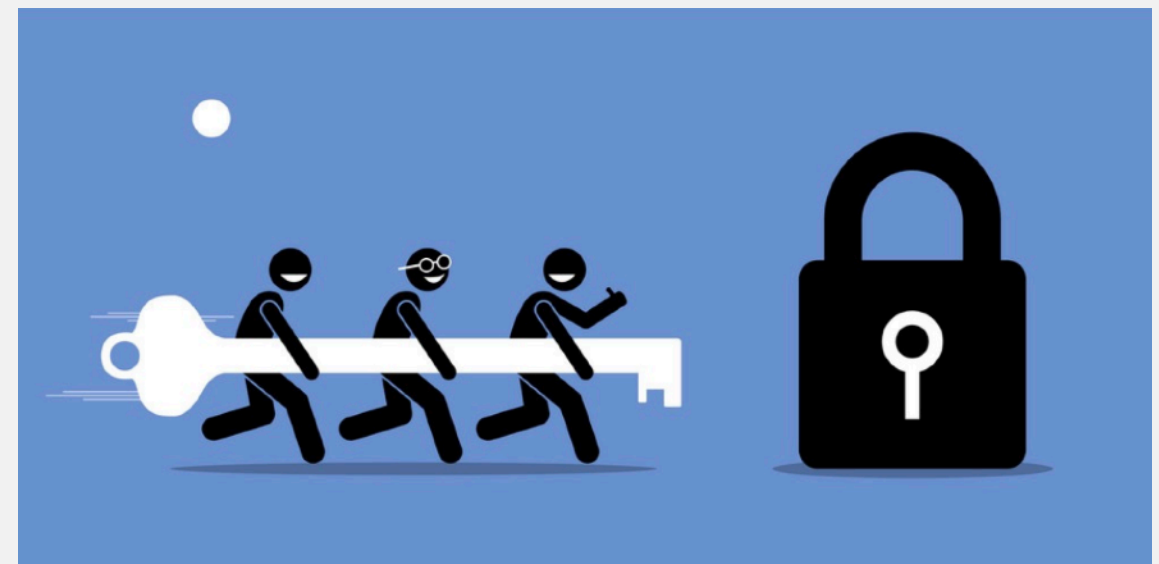
We all Work in a Team



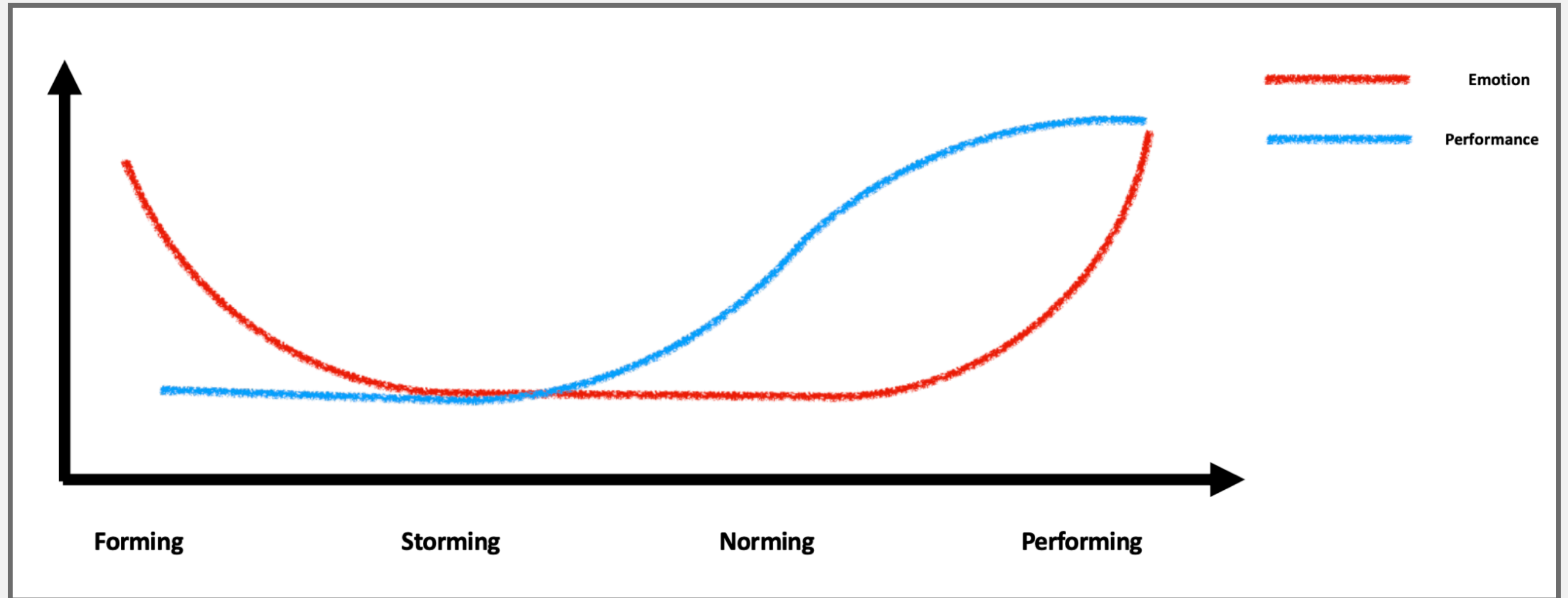
Working as a Team



- Design & implement software
- Establish a collaboration process
- Meet with the team
- Choose a leader
- Divide work and integrate
- Share knowledge
- Resolve conflicts



Stages of Team Formation





- When working with someone who is remote, how do you like to work together?
- How do you manage your time when you get busy with a lot of tasks?
- How do you feel about chatting by text message, audio call, video call?
 - Exchange phone numbers with your project partner(s) in case your Internet goes out and you still want to work on the project together.
- Negotiate when you can work on the project together outside of class.
- Have you had a positive prior teaming experience?
 - How often did your team meet?
 - Did your team have a leader? If yes, what did that leader do?
 - What was your role on the team?
 - How well did you get along with your teammates related to work, or related to non-work?



SIMPLE SABOTAGE
FIELD MANUAL
Strategic Services
(Provisional)

What Not to Do 🤔



(11) General Interference in Production

(a) Organizations

(1) Insist on "channels." Never in order to expedite.

(2) Make "spare points" by long personal experiences. appropriate "patience."

(3) When possible, for "committees, for 'tion.'" Attempt to as possible — never.

(4) Bring up as possible.

(5) Haggle communications, minutes.

(6) Refer back the last meeting question of the afternoon.

(7) Advocate and urge your "able" and avoid embarrassments.

(8) Be worried decision — raise action as is contradiction of the ground with the policy of

(b) Managers and Supervisors

(1) Demand written orders.
(2) "Misunderstand" orders. questions or engage in long about such orders. Quibble over can.

(3) Do everything possible delivery of orders. Even though may be ready beforehand, don't it is completely ready.

(4) Don't order new work until your current stocks have been exhausted, so that the slightest your order will mean a shutdown.

(5) Order high-quality materials hard to get. If you don't get them it. Warn that inferior materials inferior work.

(6) In making work assignments sign out the unimportant jobs the important jobs are assigned workers of poor machines.

(7) Insist on perfect work important products; send back those which have the least flaws defective parts whose flaws are the naked eye.

(8) Make mistakes in routing and materials will be sent to the

(12) Multiply paper work in plausible ways. Start duplicate files.

(13) Multiply the procedures involved in issuing instructions so on. See that three people everything where one would do

(14) Apply all regulations

(c) Office Workers

(1) Make mistakes in quality when you are copying orders names. Use wrong addresses.

(2) Prolong correspondence ment bureaus.

(3) Misfile essential documents.

(4) In making carbon copies few, so that an extra copying be done.

(5) Tell important callers or talking on another telephone.

(6) Hold up mail until the

(7) Spread disturbing rumors like inside dope.

(d) Employees

(1) Work slowly. Think increase the number of movements your job: use a light hammer one, try to make a small wrench one is necessary, use little force able force is needed, and so on.

(2) Contrive as many interruptions work as you can: when changing on which you are working, a lathe or punch, take needles

(3) Even if you understand the language pretend not to understand instructions in a foreign tongue.

(4) Pretend that instructions are hard to understand, and ask to have them repeated more than once. Or pretend that you are particularly anxious to do your work, and pester the foreman with unnecessary questions.

(5) Do your work poorly and blame it on bad tools, machinery, or equipment. Complain that these things are preventing you from doing your job right.

(6) Never pass on your skill and experience to a new or less skillful worker.

(7) Snarl up administration in every possible way. Fill out forms illegibly so that they will have to be done over; make mistakes or omit requested information in forms.

(8) If possible, join or help organize a group for presenting employee problems to the management. See that the procedures adopted are as inconvenient as possible for the management, involving the presence of a large number of employees at each presentation, entailing more than one meeting for each grievance, bringing up problems which are largely imaginary, and so on.

(9) Misroute materials.

(10) Mix good parts with unusable scrap and rejected parts.

General Devices for Lowering Morale and Creating Confusion

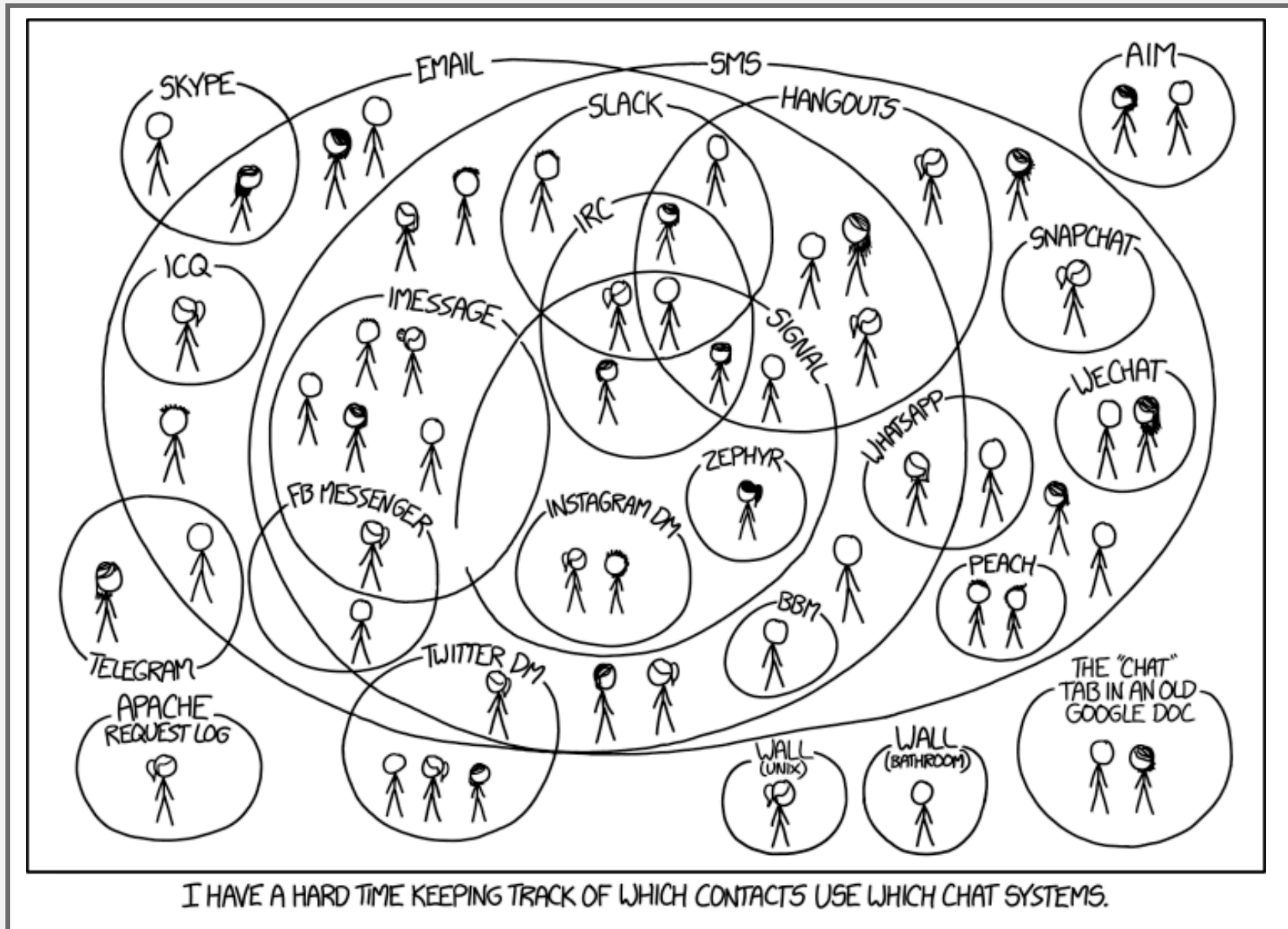
(a) Give lengthy and incomprehensible explanations when questioned.

(b) Report imaginary spies or danger to the

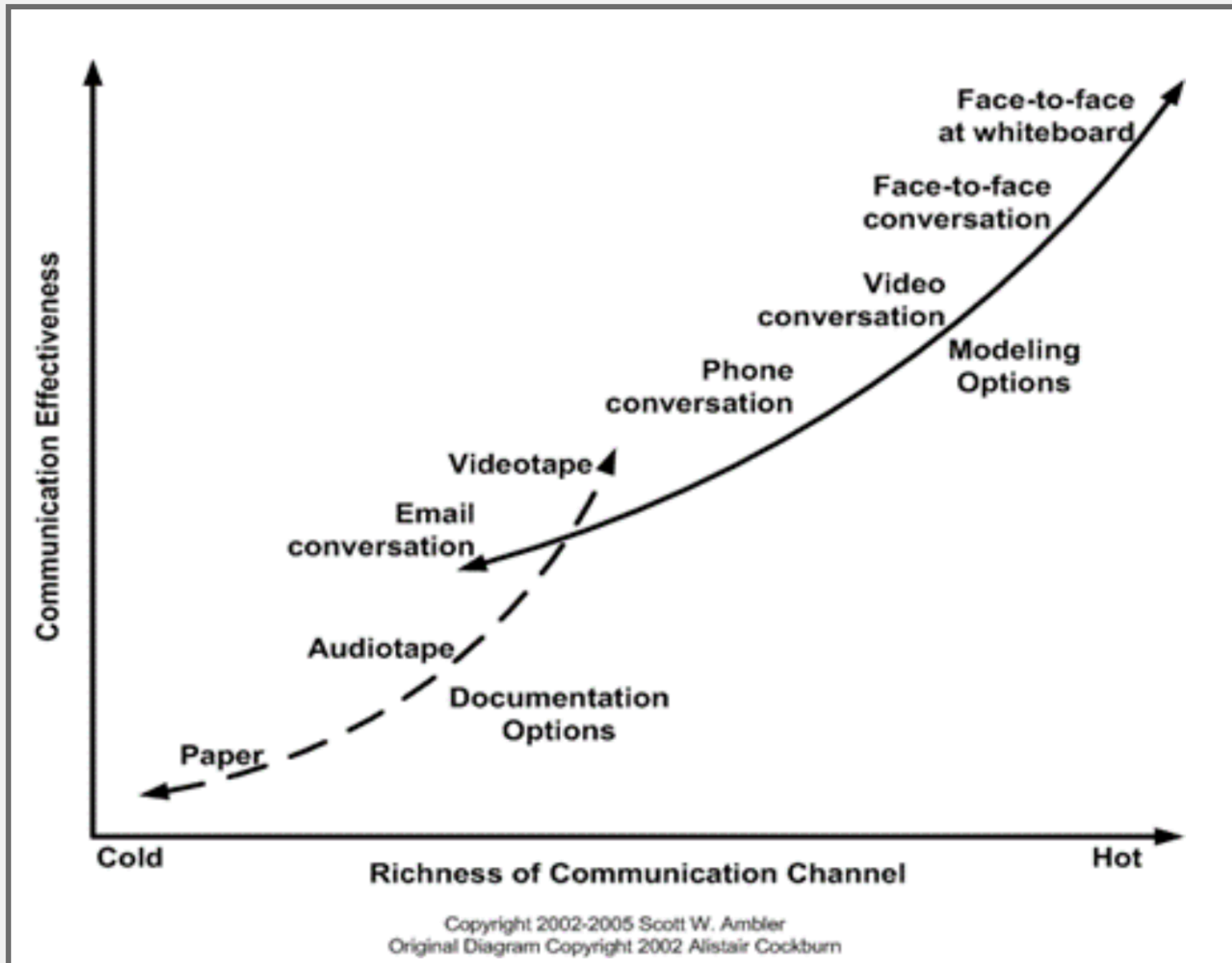
Establish a Collaboration Process



Communication App Confusion



Select the Right Communication Tools



Establish Communication Patterns



- Asana, Trello, Microsoft Projects, ...
- Github Wiki, Google Docs, Notion, ...
- Github Issues, Jira, ...
- Email, Slack, Facebook groups, ...
- Zoom, Microsoft Teams, Skype, Phone call, ...
- Face-to-face meetings



- Ed Discussions
- Regular meeting (Lectures, Recitations)
- Office Hours
- Webcourses
- Course Webpage



Communication

- Forums: Discuss implementations, research, etc. <https://discuss.pytorch.org>
- GitHub Issues: Bug reports, feature requests, install issues, RFCs, thoughts, etc.
- Slack: The [PyTorch Slack](#) hosts a primary audience of moderate to experienced PyTorch users and developers for general chat, online discussions, collaboration, etc. If you are a beginner looking for help, the primary medium is [PyTorch Forums](#). If you need a slack invite, please fill this form: <https://goo.gl/forms/PP1AGvNHpSaJP8to1>
- Newsletter: No-noise, a one-way email newsletter with important announcements about PyTorch. You can sign-up here: <https://eepurl.com/cbG0rv>
- Facebook Page: Important announcements about PyTorch. <https://www.facebook.com/pytorch>
- For brand guidelines, please visit our website at pytorch.org

Communication Expectation



- Quality of service guarantee
 - How soon will you get back to your teammates?
 - Weekend? Evening?
- Emergency
 - Tag w/ 911
 - Notify everyone with @channel

Running a Meeting



How to Run a Meeting



- The Three Rules of Running a Meeting
 - Set the Agenda
 - Start on Time. End on Time.
 - End with Action Items (and share them - Github Issues, Meeting Notes, ...)

How to Run a Meeting



- Set and document clear responsibilities and expectations
- Make everyone contribute
 - Possible Roles: Coordinator, Scribe, Checker
 - Manage Personalities
 - Be Vulnerable

Atlassian Meeting Flowchart



Every Team Needs a Leader & a Manager



- Note: these are not the same thing.
- A leader inspires with their vision of how everyone could work together.
 - They maintain a positive working environment.
 - They actively create their team culture.
 - They promote fair play among team members.
 - They acknowledge their team members' individuality.
 - They are humble and understand that others may know more than they do.

How to be a Great Manager



- Managers handle work assignments and day-to-day scheduling.
- Managers find resources to support their team's tasks.
- Managers continuously improve their team's processes.
- Managers allow team members to work autonomously, without micromanaging them.
- Managers facilitate communication between team members.

Choosing a Team Leader



- Some leaders are respected for technical excellence.
- Some leaders are chosen based on past accomplishments.
- Some leaders have high EQ (emotional quotient) and earn everyone's trust.
- Some leaders *take* the position through force of will and because others acquiesce.

Why do you want to be team leader?

Divide Work and Integrate



Is this Issue Useful?



← Image Slider #2

Open calebsylvest opened this issue just now · 0 comments

Edit **New Issue**

Labels

bug

Milestone

No milestone

Assignee

calebsylvest

Notifications

Unsubscribe

1 participant

calebsylvest commented just now

The image slider is broken

Write **Preview** Comments are parsed with [GitHub Flavored Markdown](#)

Leave a comment

Attach images by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Close **Comment**

Unsubscribe You are receiving notifications because you were assigned.

Writing Useful Github Issues



← Cropping of Image Slider Pics #3

Open calebsylvest opened this issue just now · 0 comments

Edit **New Issue**

Labels

Milestone No milestone

Assignee calebsylvest

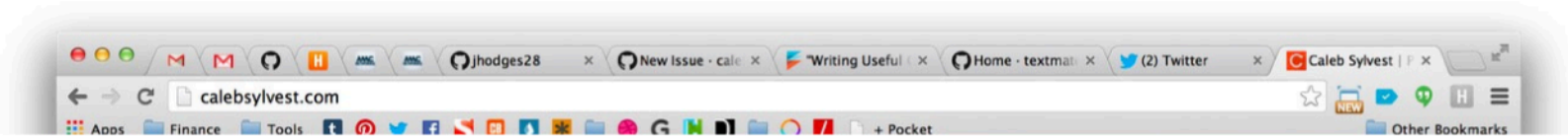
Notifications

calebsylvest commented just now

<http://calebsylvest.com/>

The cropping of the images in the slideshow seem to be off. The text is not visible and partially hidden by content below. The Developer Tools show the full-size un-cropped image is being loaded, but obviously not displaying.

Browser: Google Chrome
OS: Mavericks
Hardware: MacBook Pro Retina



Writing Useful Github Issues



- Issue should include
 - Context: explain the conditions which led you to write the issue
 - Problem or idea: the context should lead to something
 - Previous attempts to solve
 - Solution or next step (if possible)
- Be specific!
 - Include environment settings, versions, error messages, code examples when necessary

@Mention or Assign Appropriate People



Update game to use new rendering engine

Write Preview

H B I @

Now that we've decided on our new rendering engine (see #824), we need to update our collision logic to use the engine, build an engine prototype, and update the game logic.

- [] #740
- [] <https://github.com/octo-org/octo-repo/issues/1752>
- [] Update aliens and cannon game logic

Attach files by dragging & dropping, selecting or pasting them.

Styling with Markdown is supported

Submit new issue

Assignees

octocat

Labels

enhancement space game

Projects

None yet

Milestone

beta release

Linked pull requests

Successfully merging a pull request may close this issue.

None yet



- Break the project down by areas of responsibility
- Mark non-triaged issues
- Isolate issues that await additional information from the reporter
- Example:
 - Bug / Duplicate / Documentation / Help Wanted / Invalid / Enhancement
 - status: wip, status: ready to implement, status: needs discussion

Don't Forget to Follow Up and Close Issues



- closes/resolves #issue_number

Commit changes

Duplicate completion items are no more

Closes #1, resolves #dup|

- ! #1 Duplicate items in code completion
- ! #2 Duplicate items in code completion
- ! #13 Class completion list contains duplicates**

Commit directly to the `main` branch.

Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)


Pull Requests




update stuff #13

Open bunnymatic wants to merge 7 commits into `master` from `chores/fix-all-the-things`

Conversation 0 ← Commits 7 ↘ Checks 0 📄 Files changed 24

 **bunnymatic** commented 3 minutes ago Owner + 👤 ...

No description provided.

 **bunnymatic** added 7 commits 3 minutes ago

How to Write Good Pull Requests



```
## What?  
## Why?  
## How?  
## Testing?  
## Screenshots (optional)  
## Anything Else?
```


How to Write Good Pull Requests



What?

I've added support for authentication to implement Key Result 2 of OKR1. It includes model, table, controller and test. For more background, see ticket

#JIRA-123.

Why?

These changes complete the user login and account creation experience. See #JIRA-123 for more information.

How?

This includes a migration, model and controller for user authentication. I'm using Devise to do the heavy lifting. I ran Devise migrations and those are included here.

Testing?

I've added coverage for testing all new methods. I used Faker for a few random user emails and names.

Screenshots (optional)

0

Anything Else?

Let's consider using a 3rd party authentication provider for this, to offload MFA and other considerations as they arise and as the privacy landscape evolves. AWS Cognito is a good option, so is Firebase. I'm happy to start researching this path. Let's also consider breaking this out into its own service. We can then re-use it or share the accounts with other apps in the future.

How to Write Good Pull Requests



- Remember that anyone (in the company) could be reading your PR
- Be explicit about what/when feedback you want
- @mention individuals that you specifically want to involve in the discussion, and mention why.
 - “/cc @jesseplusplus for clarification on this logic”

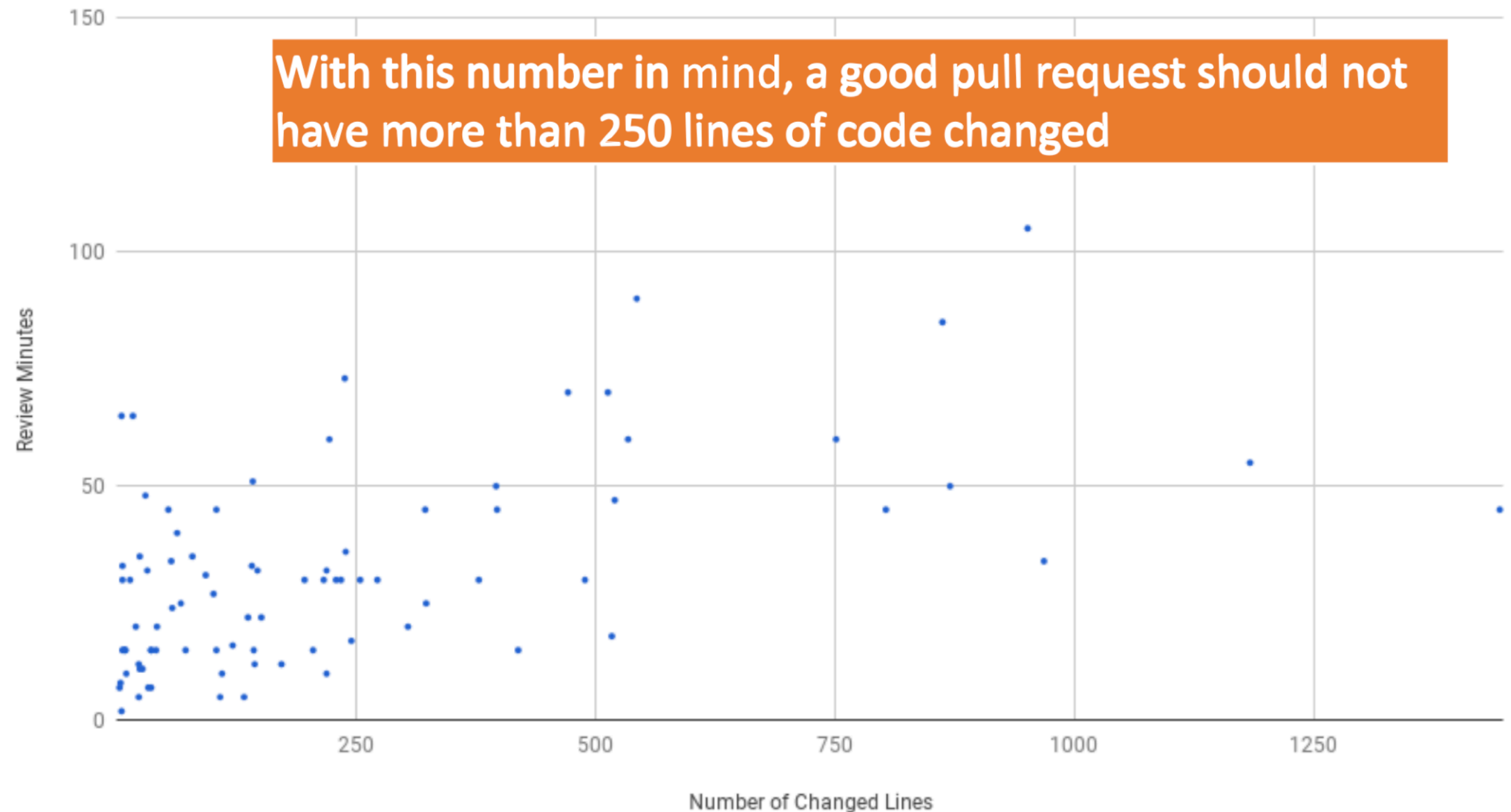
Keep your PRs Small



Keep your PRs Small



Relationship between Pull Request Size and Review Time



Offer Useful Feedback



- If you disagree strongly, consider giving it a few minutes before responding; think before you react.
- Ask, don't tell. ("What do you think about trying...?" rather than "Don't do...")
- Explain your reasons why code should be changed. (Not in line with the style guide? A personal preference?)
- Be humble. ("I'm not sure, let's try...")
- Avoid hyperbole. ("NEVER do...")
- Be aware of negative bias with online communication.

Avoid Duplicates



- “Duplicate of” issue/pull request number

The screenshot shows a GitHub comment thread. The first comment is from 'octocat' (the GitHub mascot) and says 'We should update our README.md file to include new team members.' The second comment is from 'megbird' and says 'Duplicate of #4'. Below the second comment, there is a notification that 'megbird marked this as a duplicate of #4'. The interface includes user avatars, comment text, and action buttons like 'Owner', '+', and 'Undo'.

octocat commented 4 minutes ago

We should update our README.md file to include new team members.

megbird commented 4 minutes ago

Duplicate of #4

megbird marked this as a duplicate of #4 4 minutes ago

Be a Nice Person



Date Sat, 13 Jul 2013 15:40:24 -0700
Subject Re: [GIT pull] x86 updates for 3.11
From Linus Torvalds <>

 share

638

On Sat, Jul 13, 2013 at 4:21 AM, Thomas Gleixner <tglx@linutronix.de> wrote:

```
>  
> * Guarantee IDT page alignment
```

What the F*CK, guys?

This piece-of-shit commit is marked for stable, but you clearly never even test-compiled it, did you?

Because on x86-64 (the which is the only place where the patch matters), I don't see how you could have avoided this honking huge warning otherwise:

```
arch/x86/kernel/traps.c:74:1: warning: braces around scalar  
initializer [enabled by default]  
  gate_desc idt_table[NR_VECTORS] __page_aligned_data = { { { { 0, 0 } } }, };  
  ^
```


Knowledge Sharing





No matter the format, documentation is important

Building on top of others' work in a community-like way can be an accelerator, both in open source and in companies. Documentation often signals if a repository is reliable to reuse code from, or if it's an active project to contribute to. What signs do developers look for?

In both open source projects and enterprises, developers see about

50%

productivity boost with easy-to-source documentation

What the data shows: At work, developers consider documentation trustworthy when it is up-to-date (e.g., looking at time-stamps) and has a high number of upvotes from others. Open source projects use READMEs, contribution guidelines, and GitHub Issues, to elevate the quality of any project, and to share information that makes them more attractive to new contributors. Enterprises can adopt the same best practices to achieve similar success.

In both environments, developers see about a 50% productivity boost when documentation is up-to-date, detailed, reliable, and comes in different formats (e.g. articles, videos, forums).

Using the data: Review the documentation your team consumes: When was the last time it was updated? Can everyone on your team improve the documentation? Check this frequently to stay on track.

Types of Documentation

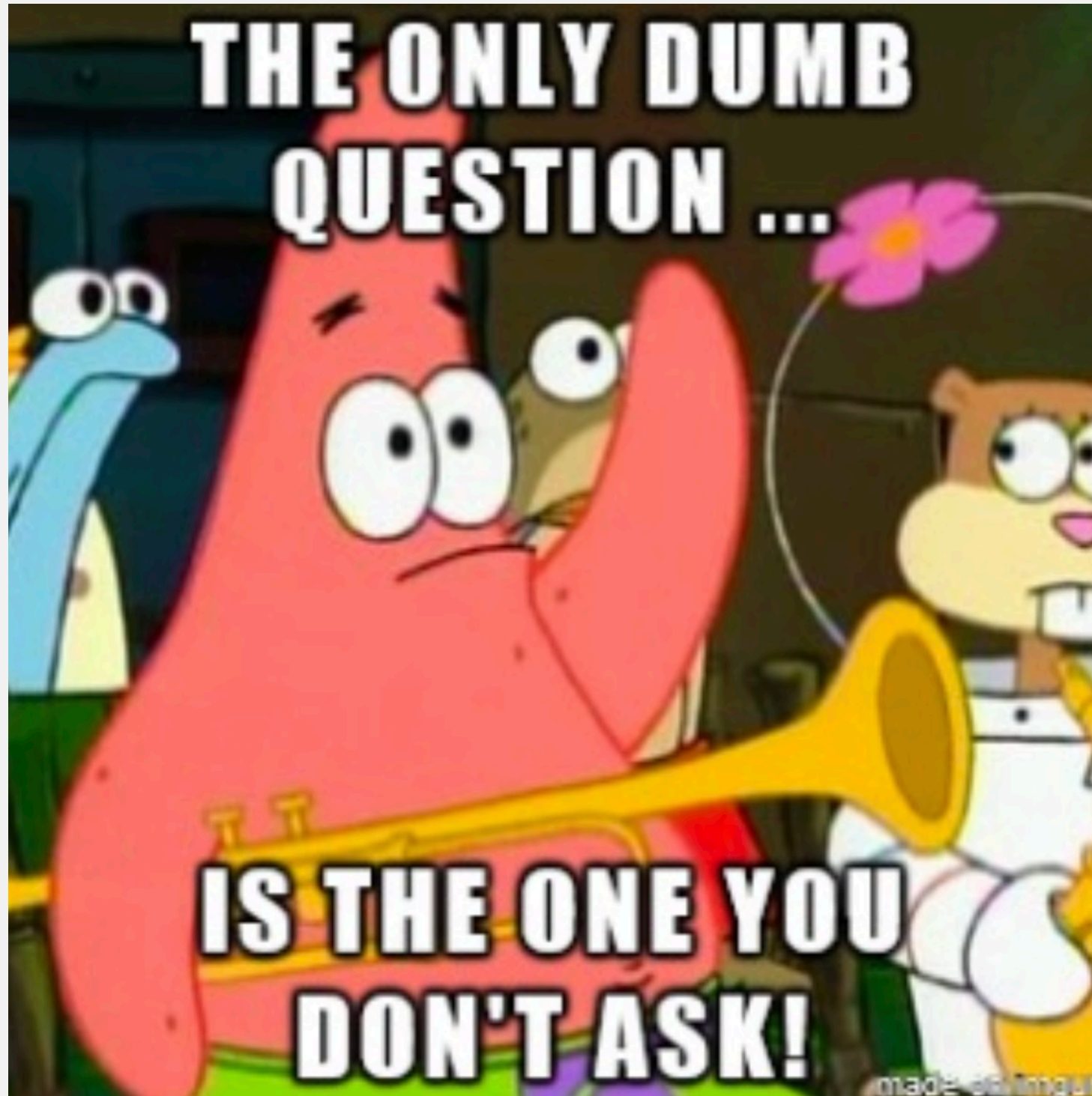


Knowledge Type	Description (Excerpt)
Functionality and Behavior	Describes what the API does (or does not do) in terms of functionality or features. Describes what happens when the API is used (a field value is set, or a method is called).
Concepts	Explains the meaning of terms used to name or describe an API element, or describes design or domain concepts used or implemented by the API.
Directives	Specifies what users are allowed / not allowed to do with the API element. Directives are clear contracts.
Purpose and Rationale	Explains the purpose of providing an element or the rationale of a certain design decision. Typically, this is information that answers a "why" question: Why is this element provided by the API? Why is this designed this way? Why would we want to use this?
Quality Attributes and Internal Aspects	Describes quality attributes of the API, also known as non-functional requirements, for example, the performance implications. Also applies to information about the API's internal implementation that is only indirectly related to its observable behavior.
Control-Flow	Describes how the API (or the framework) manages the flow of control, for example by stating what events cause a certain callback to be triggered, or by listing the order in which API methods will be automatically called by the framework itself.
Structure	Describes the internal organization of a compound element (e.g. important classes, fields, or methods), information about type hierarchies, or how elements are related to each other.
Patterns	Describes how to accomplish specific outcomes with the API, for example, how to implement a certain scenario, how the behavior of an element can be customized, etc.
Code Examples	Provides code examples of how to use and combine elements to implement certain functionality or design outcomes.
Environment	Describes aspects related to the environment in which the API is used, but not the API directly, e.g., compatibility issues, differences between versions, or licensing information.
References	Includes any pointer to external documents, either in the form of hyperlinks, tagged "see also" reference, or mentions of other documents (such as standards or manuals).
Non-information	A section of documentation containing any complete sentence or self-contained fragment of text that provides only uninformative boilerplate text.



- Internal document for your team (e.g., meeting note)
- Documentation for project contributors
- Documentation for non-developer collaborators (e.g., UX researchers)
- Documentation for developer users
- Documentation for clients with no software knowledge
- User manual for end users

Importance of Asking Questions



How to Ask Questions



New To Coding. Can anyone assist me?

Asked 7 years, 1 month ago Modified 7 years, 1 month ago Viewed 47 times

I am trying to make a word counter and I just cant seem to get it. Can anyone help?

-4

```
import re
print("Welcome To This Software Made By Aaron!")
word = raw_input("Enter Your Words: ")
Check = 0
Right = 0
Length = len(word)
while True:
    if Right == 1:
        if Length < Check:
            Check = Check + 1
            print(Check)
    if Length == Check:
        Right = 1

print("Your Word Count Is " +Check)
```



Make it Easy for People to Help You



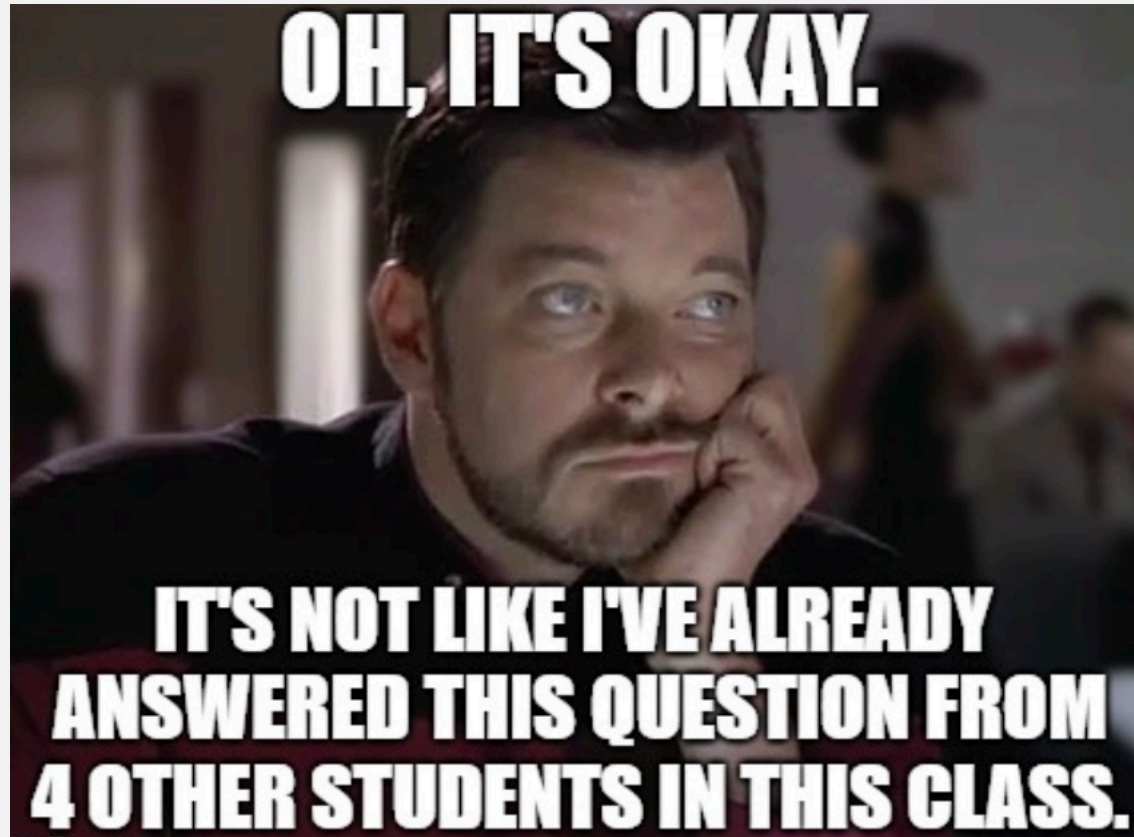
- I am trying to ____, so that I can ____. I am running into ____.
I have looked at ____ and tried ____.
- + I'm using this tech stack: ____.
- + I'm getting this error/result: ____.
- + I think the problem could be ____.



RESEARCH-ARTICLE 🐦 in 🍷 f ✉

Mining duplicate questions in stack overflow

Authors: Muhammad Ahasanuzzaman, Muhammad Asaduzzaman, Chanchal K. Roy, Kevin A. Schneider
[Authors Info & Claims](#)



[Published: 04 November 2015](#)

Studying the needed effort for identifying duplicate issues

[Mohamed Sami Rakha](#) ✉, [Weiyi Shang](#) & [Ahmed E. Hassan](#)

[Empirical Software Engineering](#) 21, 1960–1989 (2016) | [Cite this article](#)

748 Accesses | 19 Citations | 1 Altmetric | [Metrics](#)

Abstract

Many recent software engineering papers have examined duplicate issue reports. Thus far, duplicate reports have been considered a hindrance to developers and a drain on their resources. As a result, prior research in this area focuses on proposing automated approaches to accurately identify duplicate reports. However, there exists no studies that attempt to

Resolving Conflicts





Updated with New Approaches for Today's Communication Challenges

OVER 5 MILLION COPIES SOLD

crucial conversations

— THIRD EDITION —



**TOOLS FOR TALKING WHEN
STAKES ARE HIGH**

JOSEPH GRENNY • KERRY PATTERSON • RON McMILLAN
AL SWITZLER • EMILY GREGORY



Communication

Communication

**You can't solve any Problem
without Communication!**

Communication

Communication

Conflict Resolution



- Your goal: Find a solution to the problem and move forward.
 - As a smart person on "TedLasso" once said, "Fight forward, not back."
- Make sure that everybody works from the same set of facts.
- Establish ground rules for your team's discussion.
 - Talk about how the situation made you feel. Never presume anything about anyone else.
- Remain calm and rational. If you feel triggered or threatened, extract yourself from the situation, wait an hour to chill out, and then try again.
- If you reach an impasse, talk to your team leader.
- If your team remains in conflict, escalate to Dr. Moran.
 - I can help to mediate