CEN 5016: Software Engineering

Spring 2024

University of Central Florida

Dr. Kevin Moran

Week 3 - Class 1: Software Teams & Communication

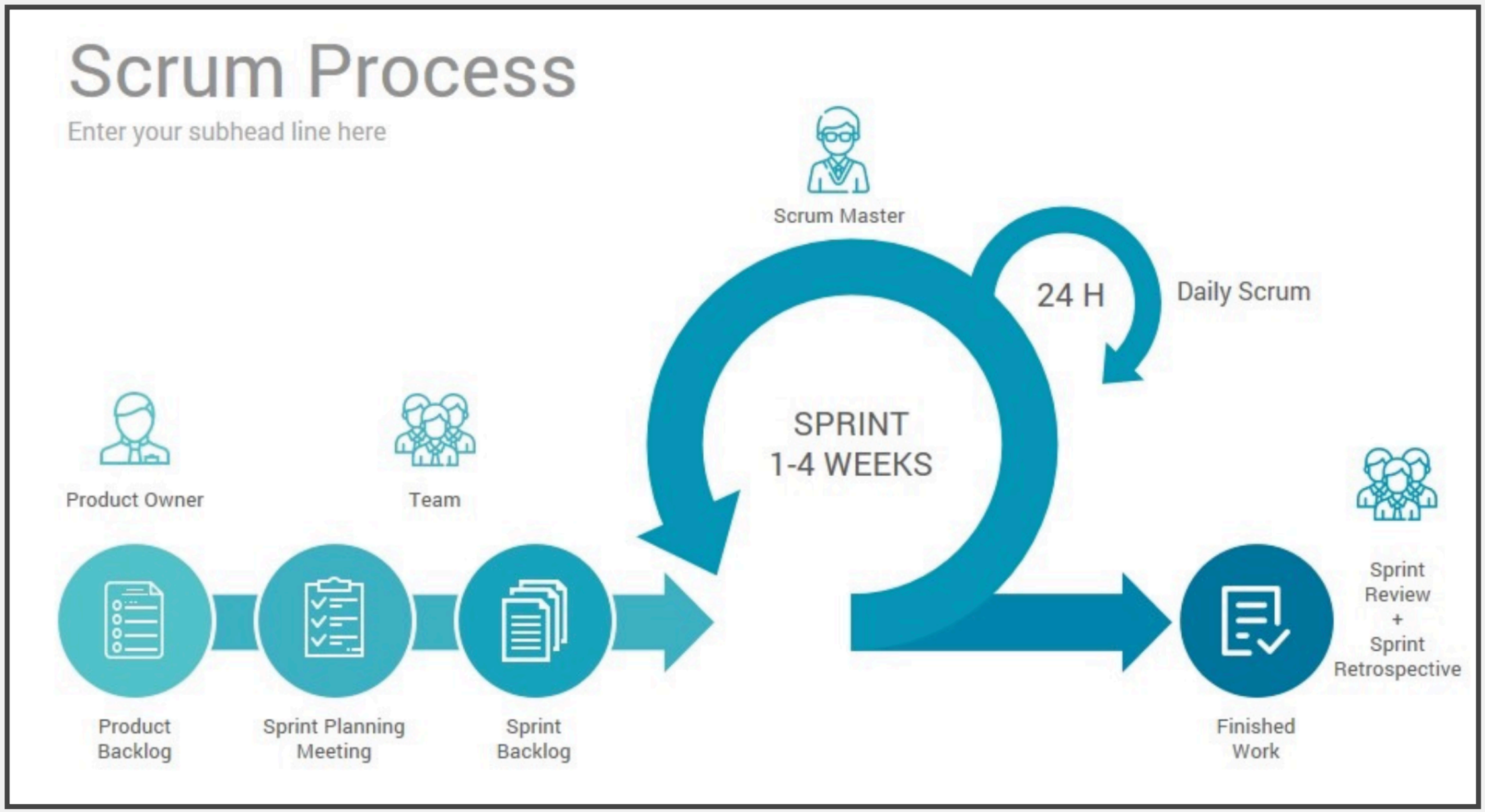
Administrivia

- *Team-forming this week - Due Today EoD!!*
  - Teams of 3 students
  - See Ed Discussions Post
- *Assignment 2 posted*
  - Getting familiar with NodeBB, the subject of our SDE project
  - Due in one week on Tuesday, January 30th, 11:59pm

Project Planning & Agile Development

Scrum

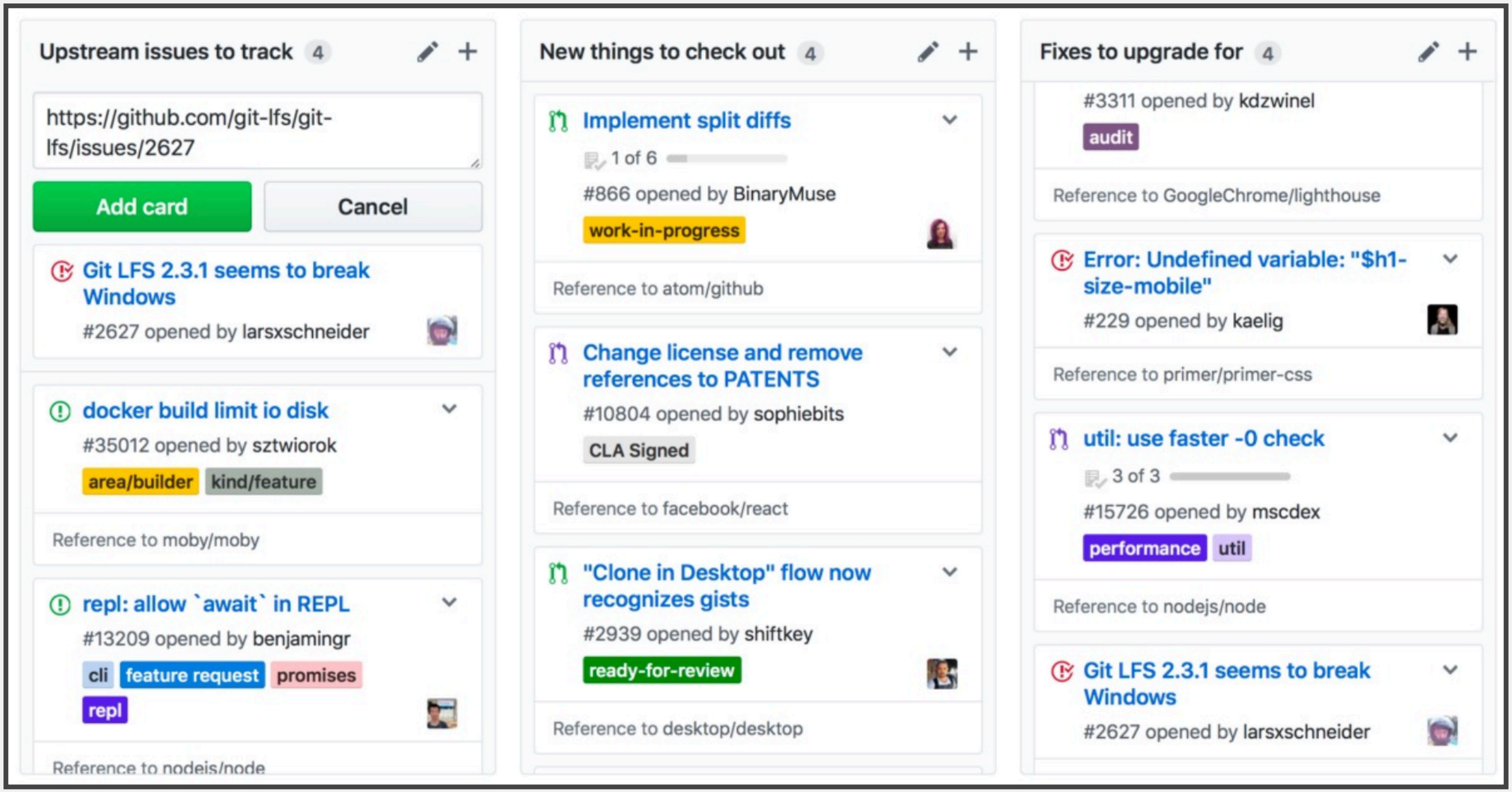
Elements of Scrum



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

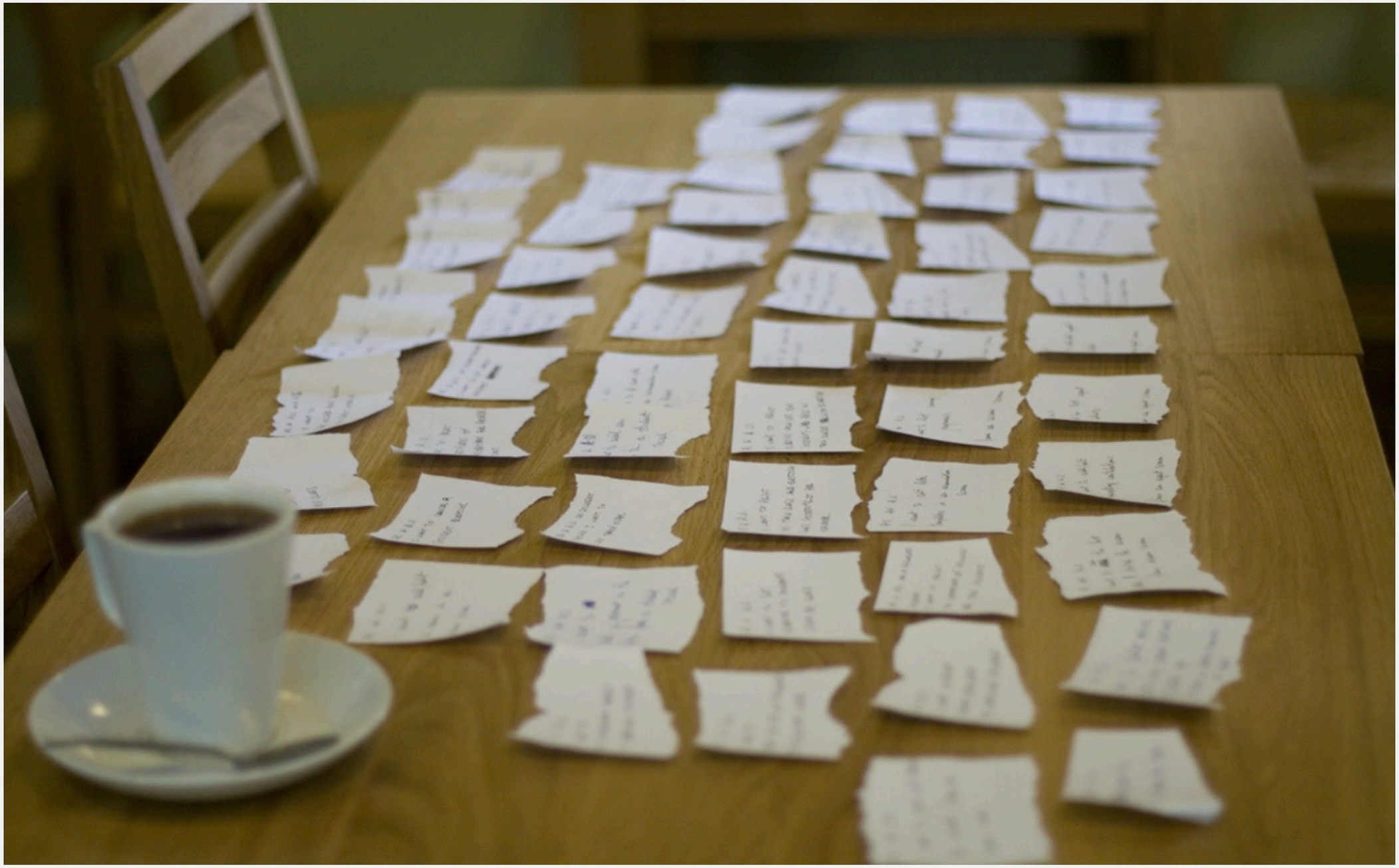
- Sprint Planning Meeting
  - Entire Team decides together what to tackle for that sprint

- • 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?

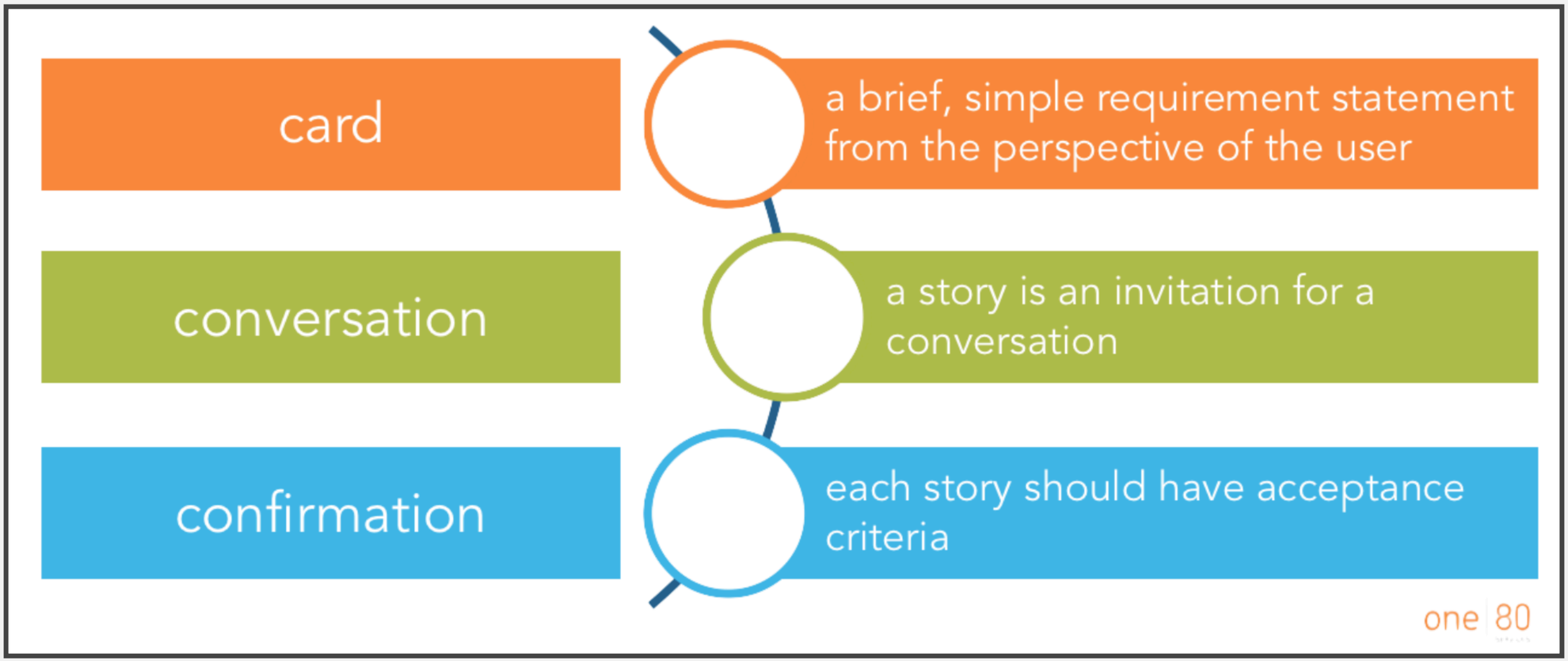
- Sprint Retrospective
  - Review sprint process

- Sprint Review Meeting
  - Review Product

User Stories



User Stories



Card

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

Card

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

Confirmation

- 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

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.

Valuable

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

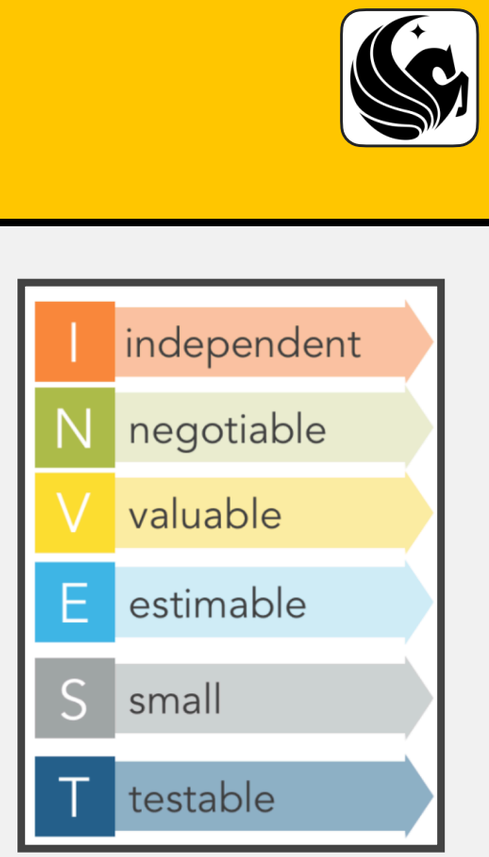
Estimable

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

Small

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

Testable



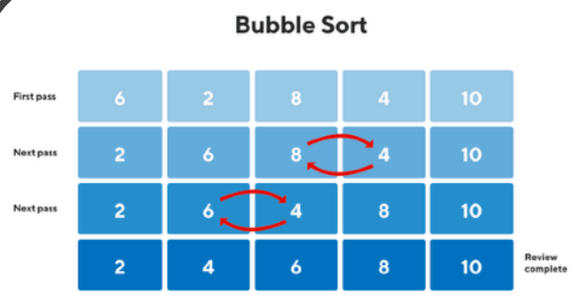
- Ensures understanding of task
- We know when we can mark task “Done”
- Unable to test === I do not understand it

Software Teams & Communication

Learning Goals

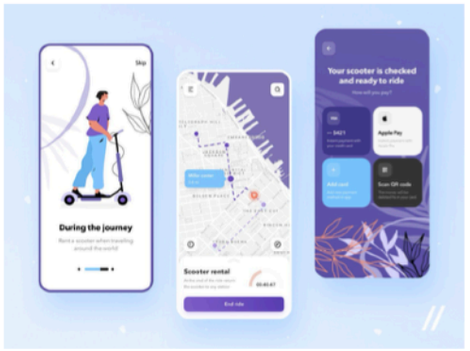
- Describe the pros and cons of working as a team
- Recognize the importance of communication in collaboration
- Recognize the need of having multiple communication channels
- Select an appropriate communication tool for a given communication goal
- Ask technical questions effectively
- Write clear and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

We all Work in a Team

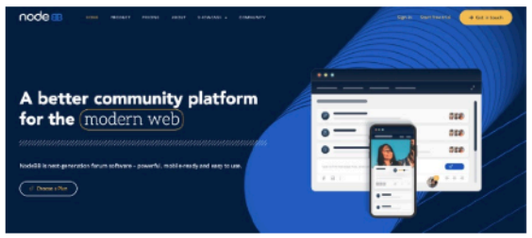


Bubble Sort

Monopoly Game



Scooter App



NodeBB

Autonomous Vehicle

We all Work in a Team

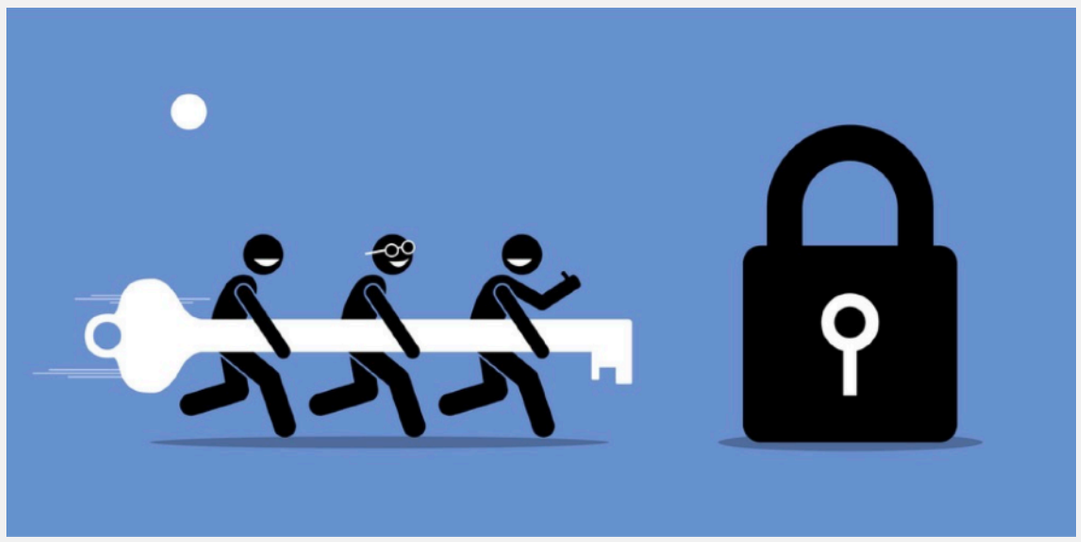
Activity: Working Solo vs. As a Team

- Write down two pros and two cons about working solo on a software project versus working on a team.
- Pair with your neighbor and discuss your answers. Do you agree?
- We will discuss some of your answers!

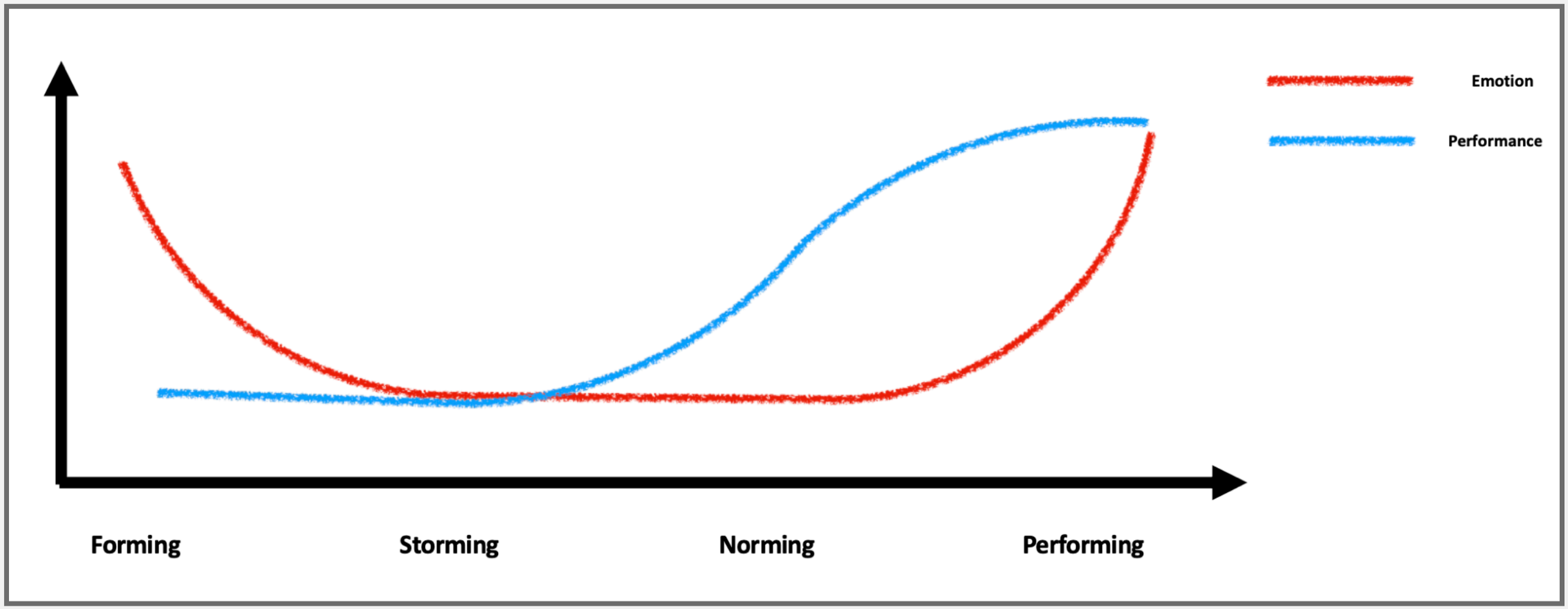
http://tinyurl.com/cen5016-act3

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



Norming

- 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?

What Not to Do 🧐


SIMPLE SABOTAGE

FIELD MANUAL

Strategic Services

(Provisional)

What Not to Do 🧐

(11) *General Interfere*
Production

(a) Organizations

(1) Insist or
"channels." Never
in order to expedi

(2) Make "sp
possible and at
"points" by long a
sonal experiences.
appropriate "patr

(3) When p
committees, for '
tion." Attempt to
as possible — nev

(4) Bring up
as possible.

(5) Haggle
munications, min

(6) Refer ba
the last meeting
question of the a

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

(8) Be worri
decision — raise
action as is cont
diction of the gro
with the policy o

(b) Managers and Supervisors

(1) Demand written order

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

(3) Do everything possib
delivery of orders. Even though
may be ready beforehand, don'
it is completely ready.

(4) Don't order new wo
until your current stocks have b
hausted, so that the slightest
your order will mean a shutdo

(5) Order high-quality ma
hard to get. If you don't get th
it. Warn that inferior materia
ferior work.

(6) In making work assi
sign out the unimportant job
the important jobs are assign
workers of poor machines.

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

(8) Make mistakes in rout
and materials will be sent to th

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

(13) Multiply the procedur
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 qua
when you are copying orders
names. Use wrong addresses.

(2) Prolong corresponde
ment bureaus.

(3) Misfile essential docu

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

(5) Tell important caller
or talking on another teleph

(6) Hold up mail until t

(7) Spread disturbing r
like inside dope.

(d) Employees

(1) *Work slowly.* Think
crease the number of movem
your job: use a light hammer
one, try to make a small wre
one is necessary, use little for
able force is needed, and so o

(2) Contrive as many int
work as you can: when char
on which you are working, g
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 pos-
sible 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 man-
agement. 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.

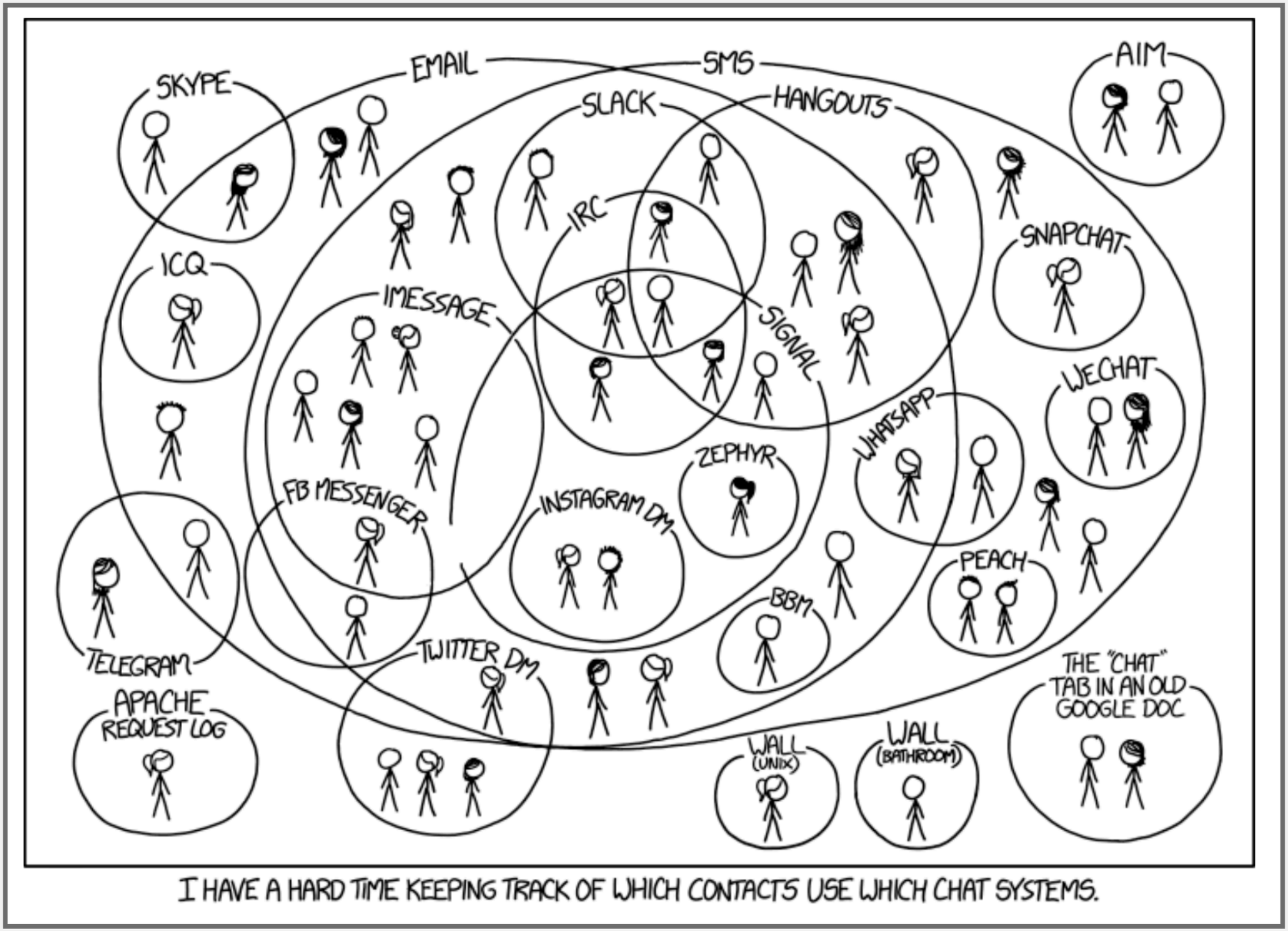
General Devices for Lowering Morale and Creat-
onfusion

(a) Give lengthy and incomprehensible expla-
tions when questioned.

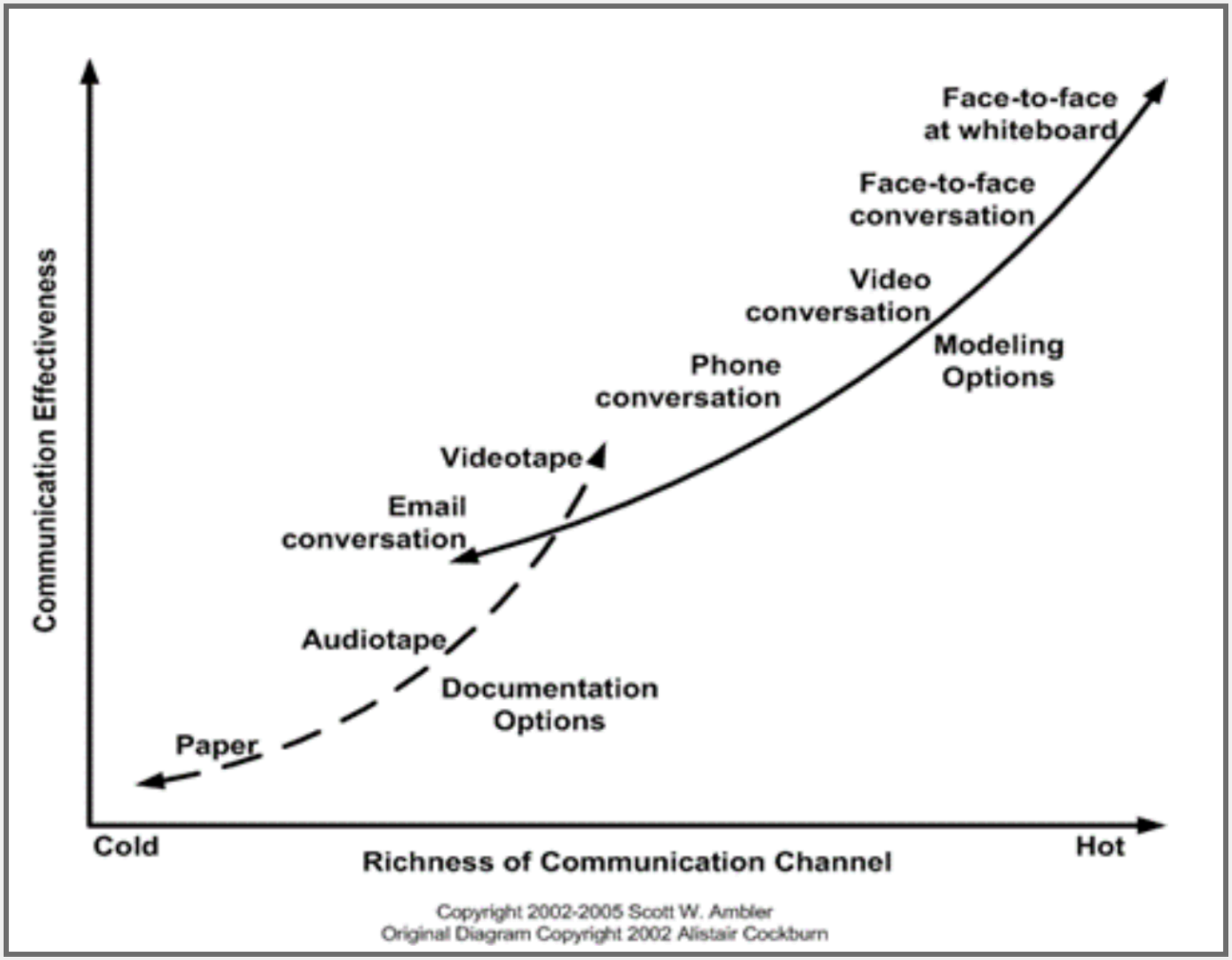
(b) Report imaginary spies or danger to the
tape or police

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

CEN 5016 Communication Channels

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

Check Out Other Projects

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

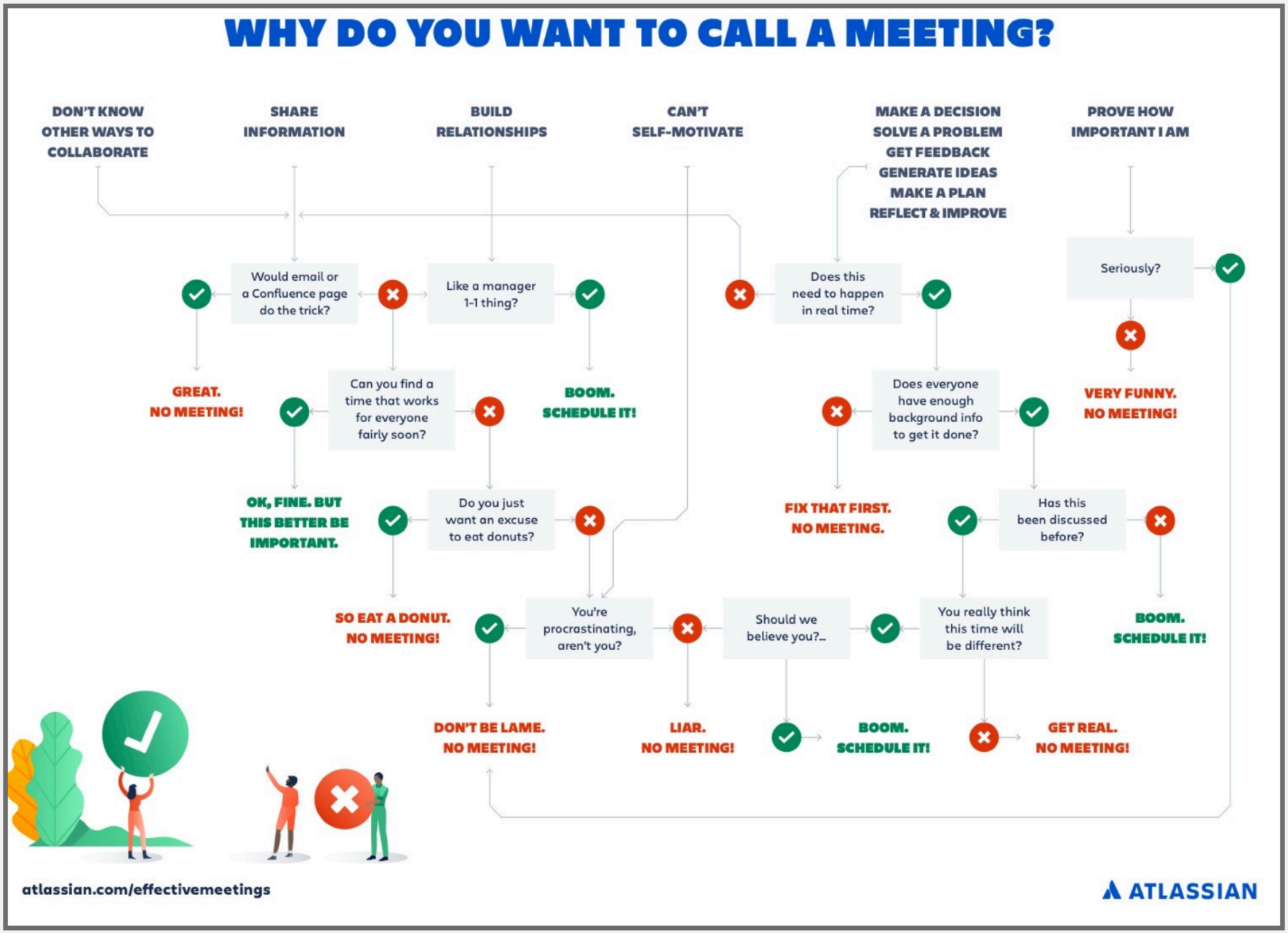
How to Run a Meeting

- 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 communicate between team members.

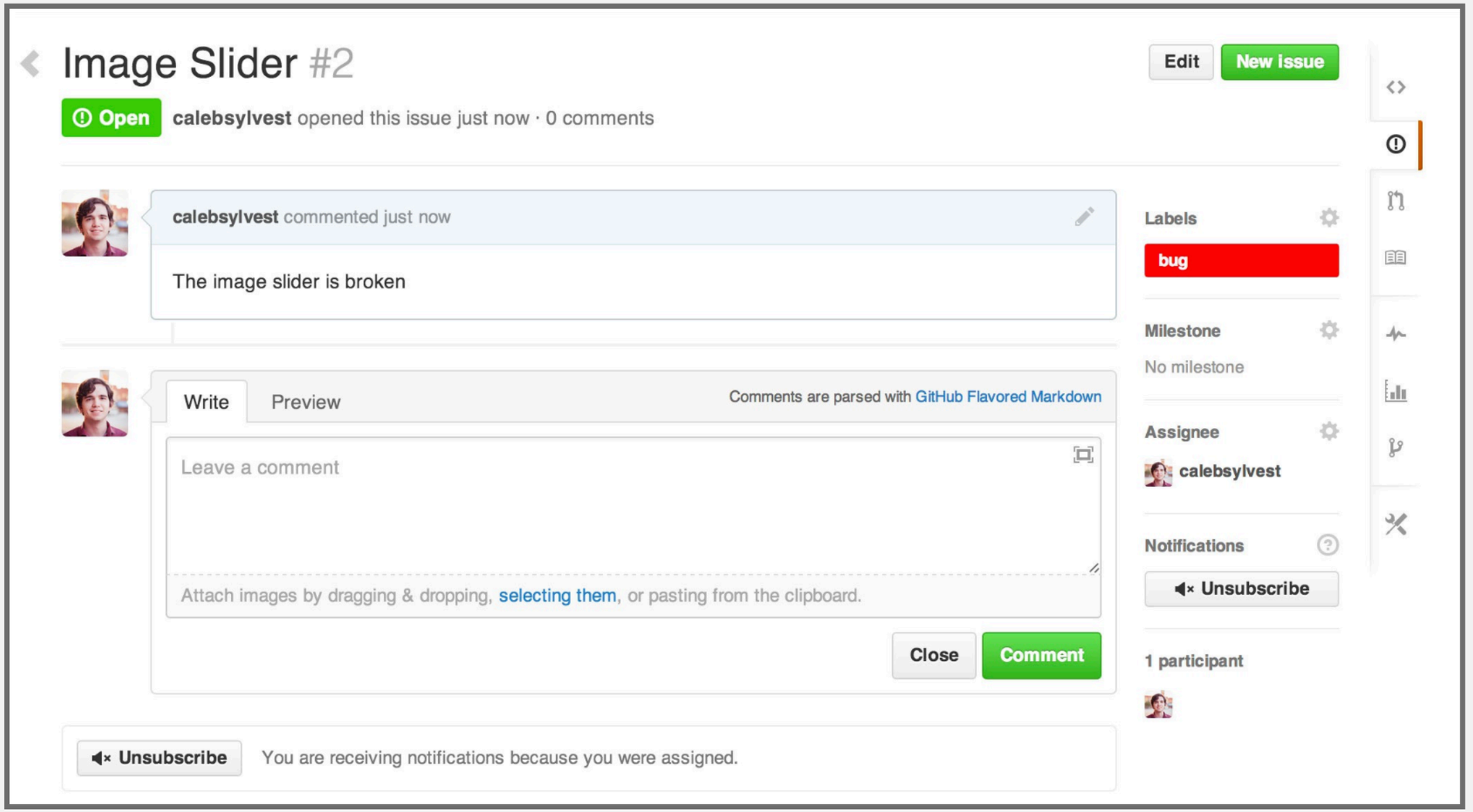
Choosing a Team Leader

- 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.

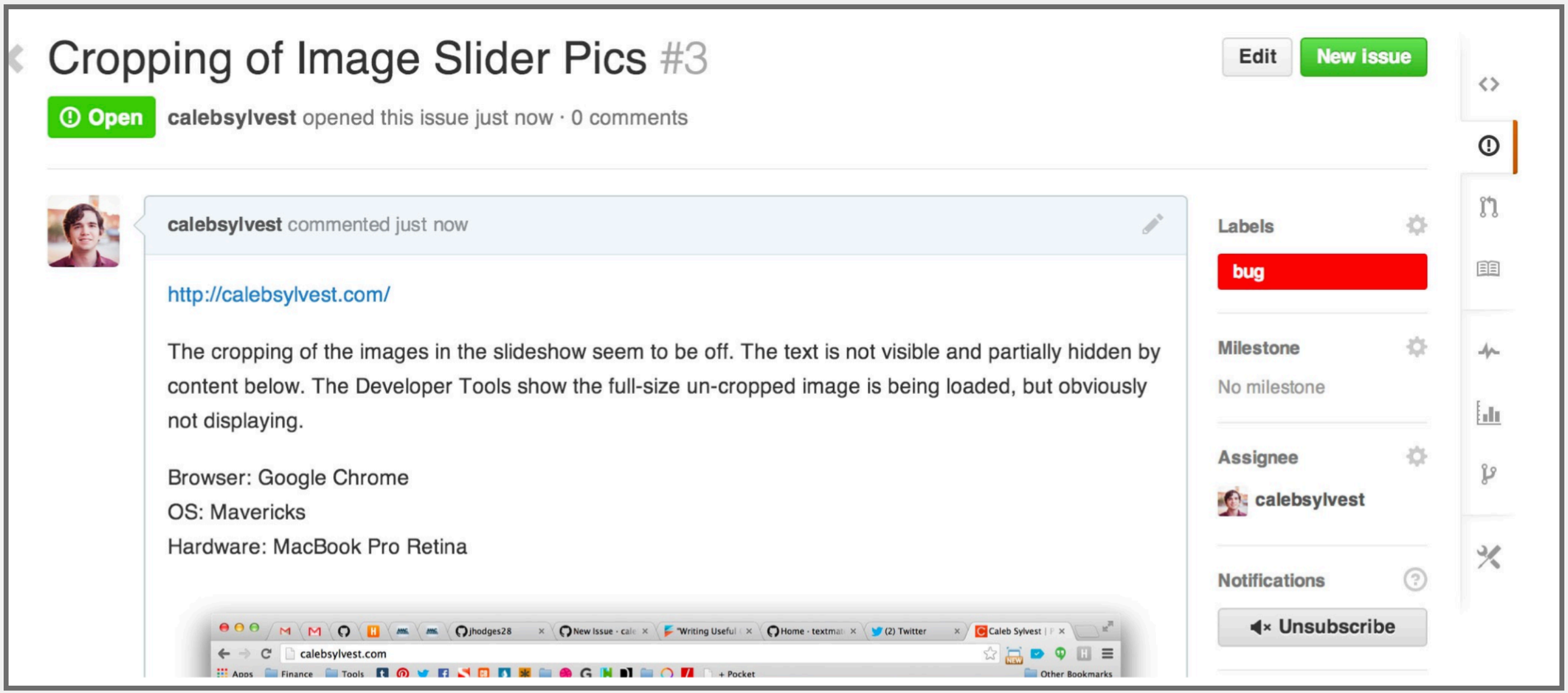
Why do you want to be team leader?

Divide Work and Integrate

Is this Issue Useful?



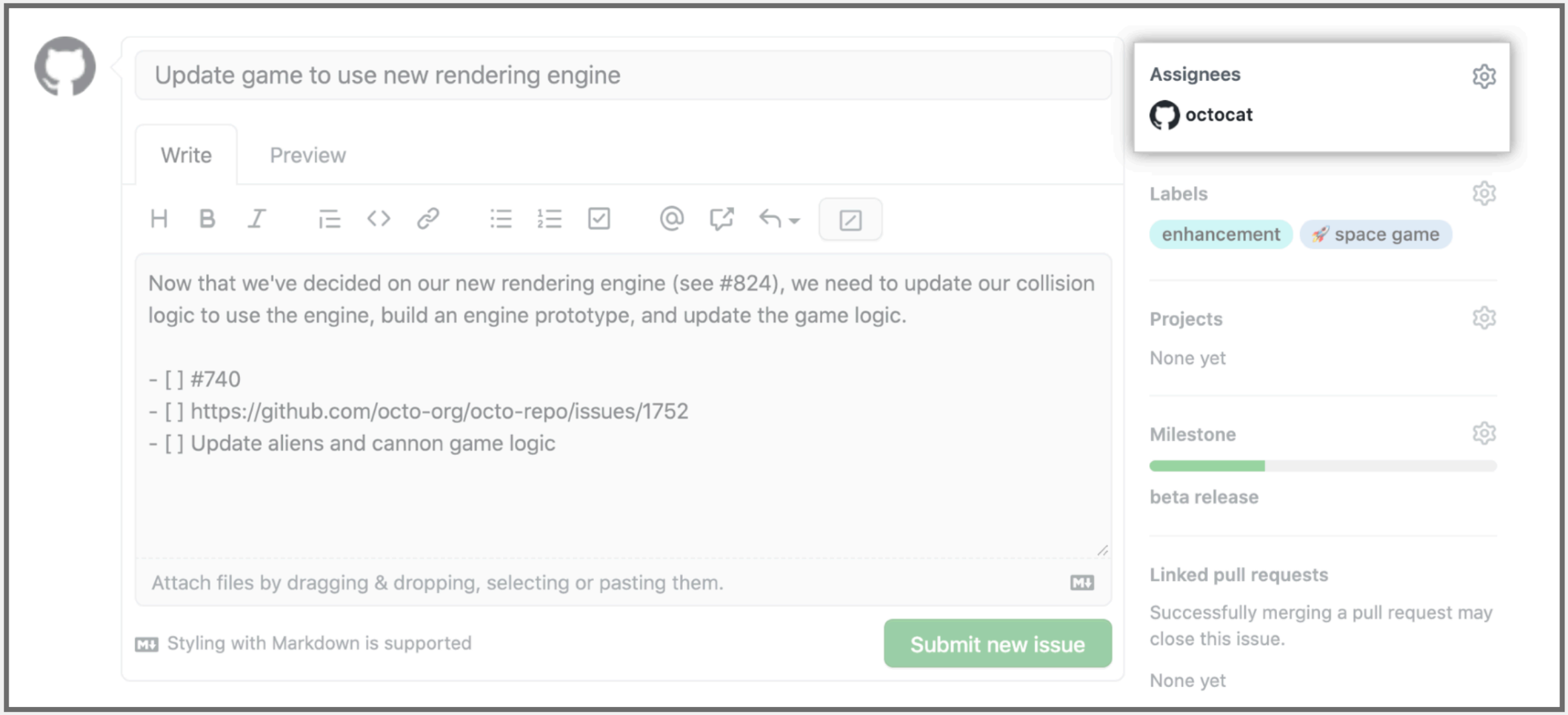
Writing Useful Github Issues



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

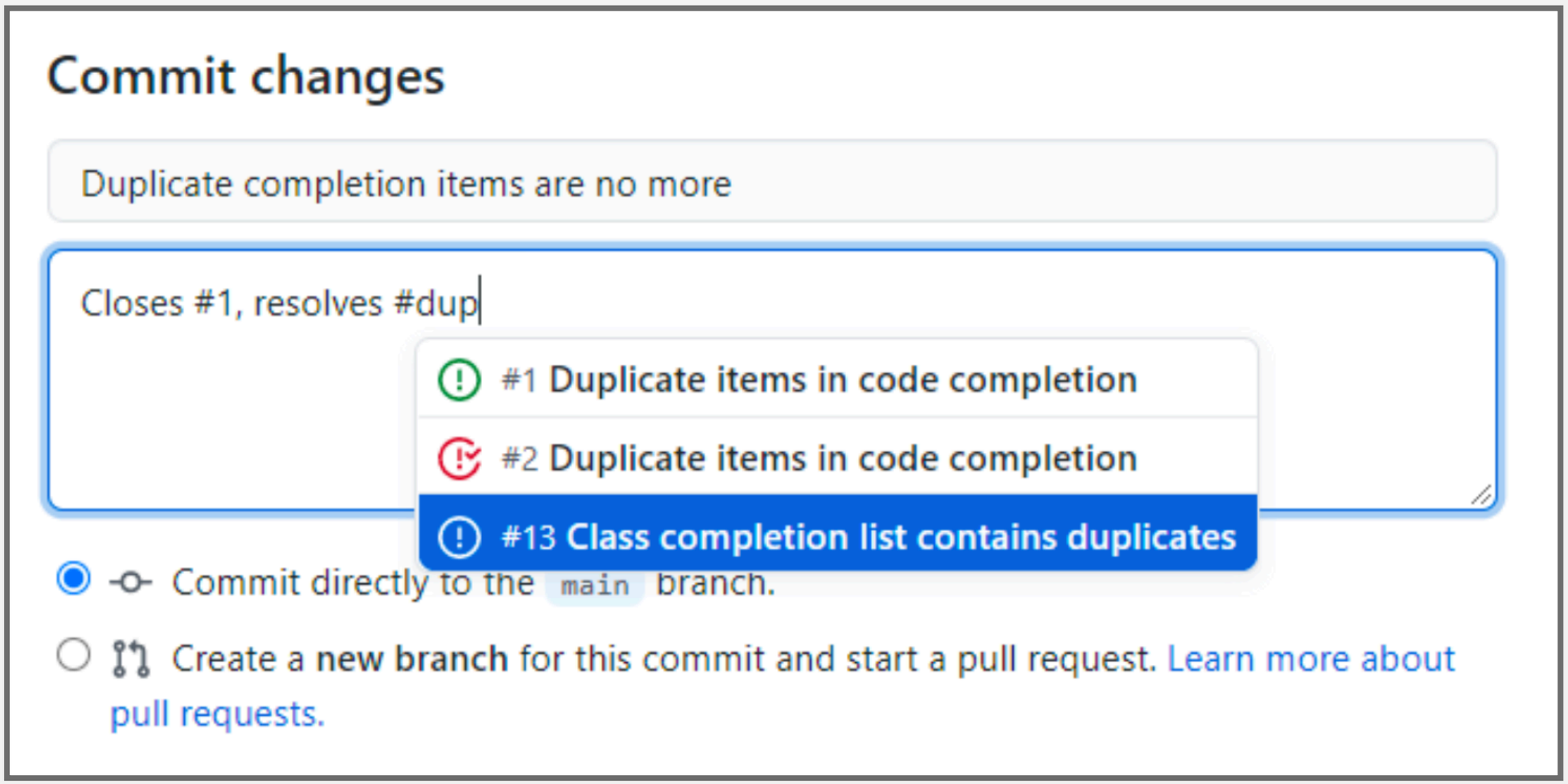


Use Labels

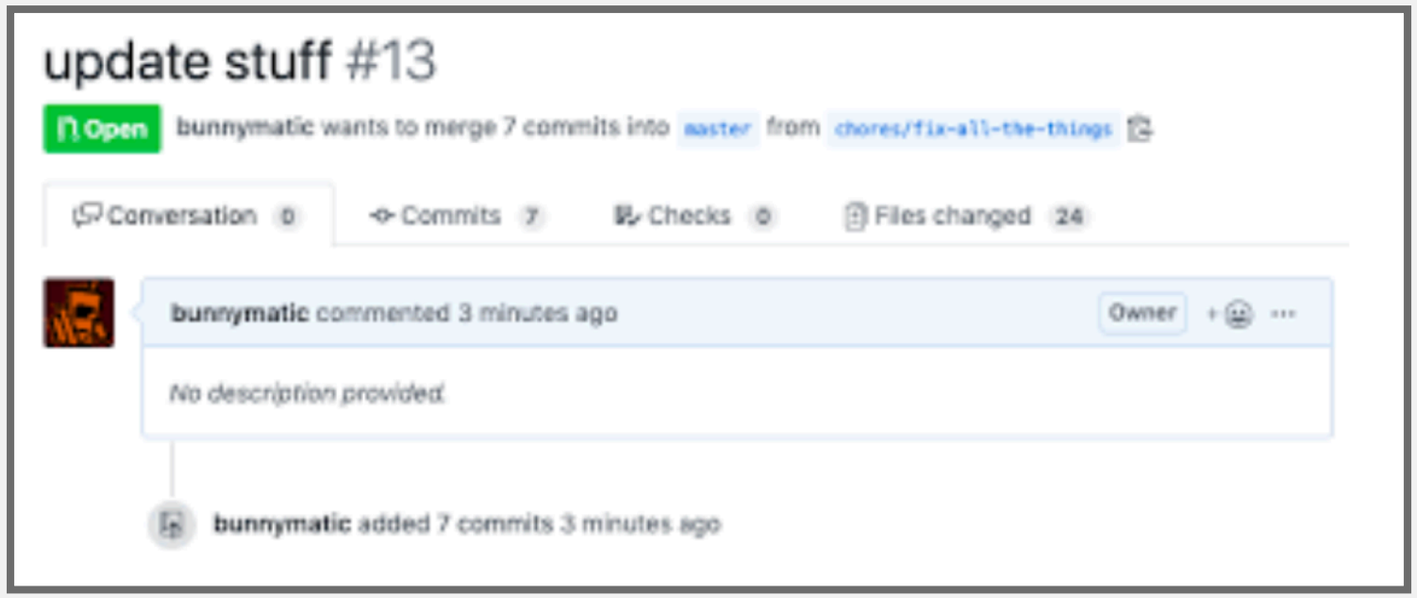
- Break the project down by areas of responsibility
- Mark non-triaged issues
- Isolate issues that await additional information from the reporter
- Example:
  - Bug / Duplicate / Documentation / Help Wanted / Invalid / Enhancement
  - status: wip, status: ready to implement, status: needs discussion

Don’t Forget to Follow Up and Close Issues

- closes/resolves #issue_number



Pull Requests



How to Write Good Pull Requests

```
## What?

## Why?

## How?

## Testing?

## Screenshots (optional)

## Anything Else?
```

How to Write Good Pull Requests

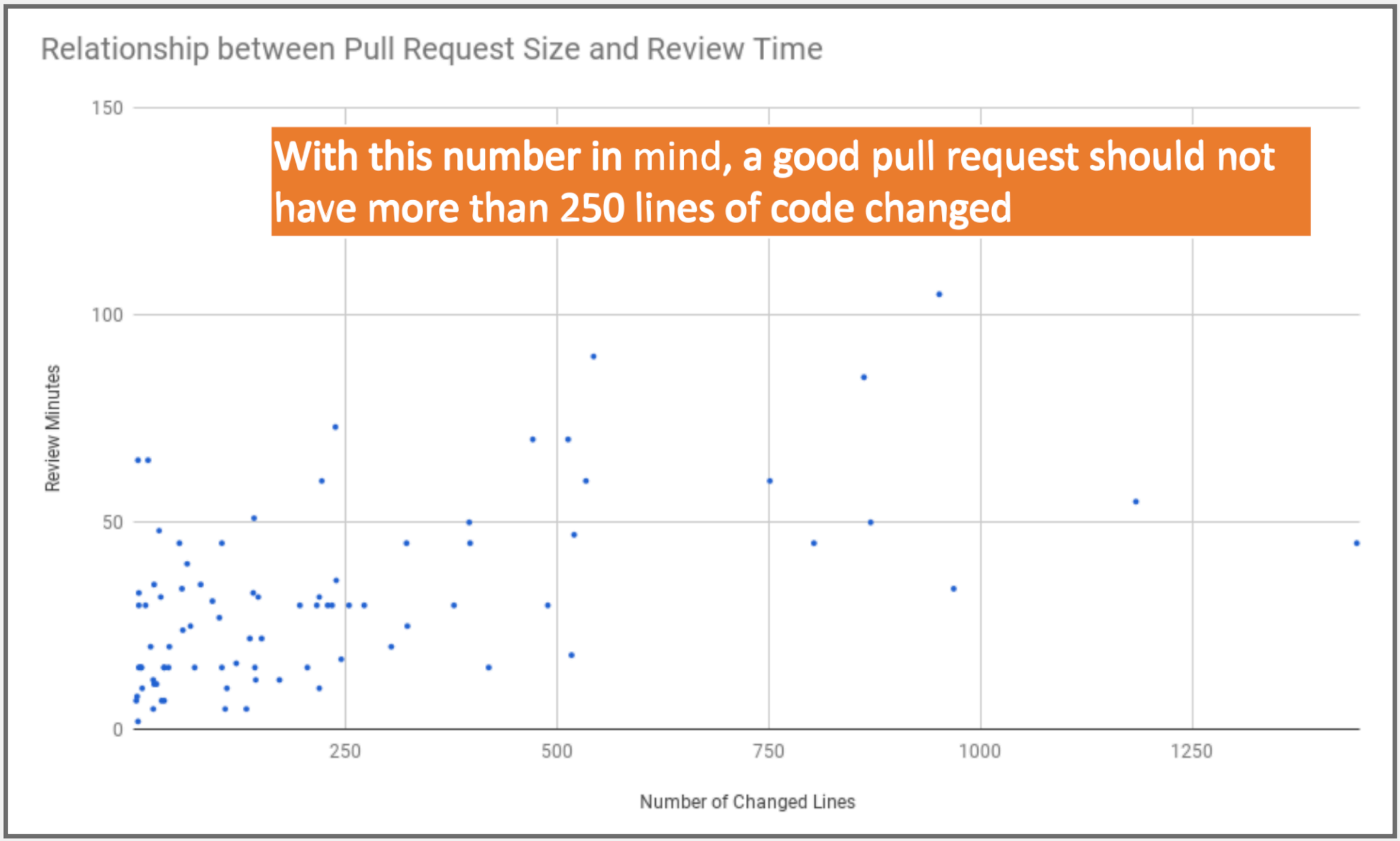
```
## What?
I've added support for authentication to implement Key Result 2 of OKR1. It includes model, table,
controller and test. For more background, see ticket
#JIRA-123.
## Why?
These changes complete the user login and account creation experience. See #JIRA-123 for more
information.
## How?
This 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

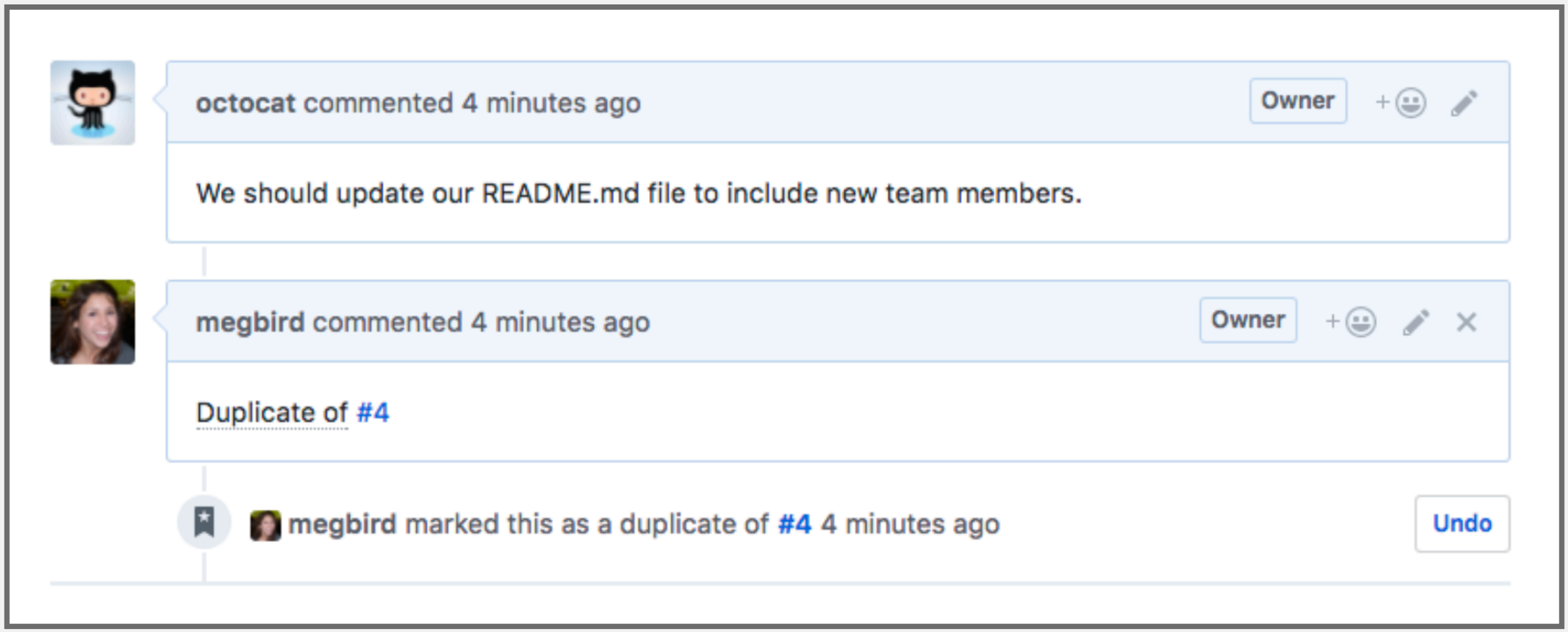


Offer Useful Feedback

- 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



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

Importance of Documentation

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

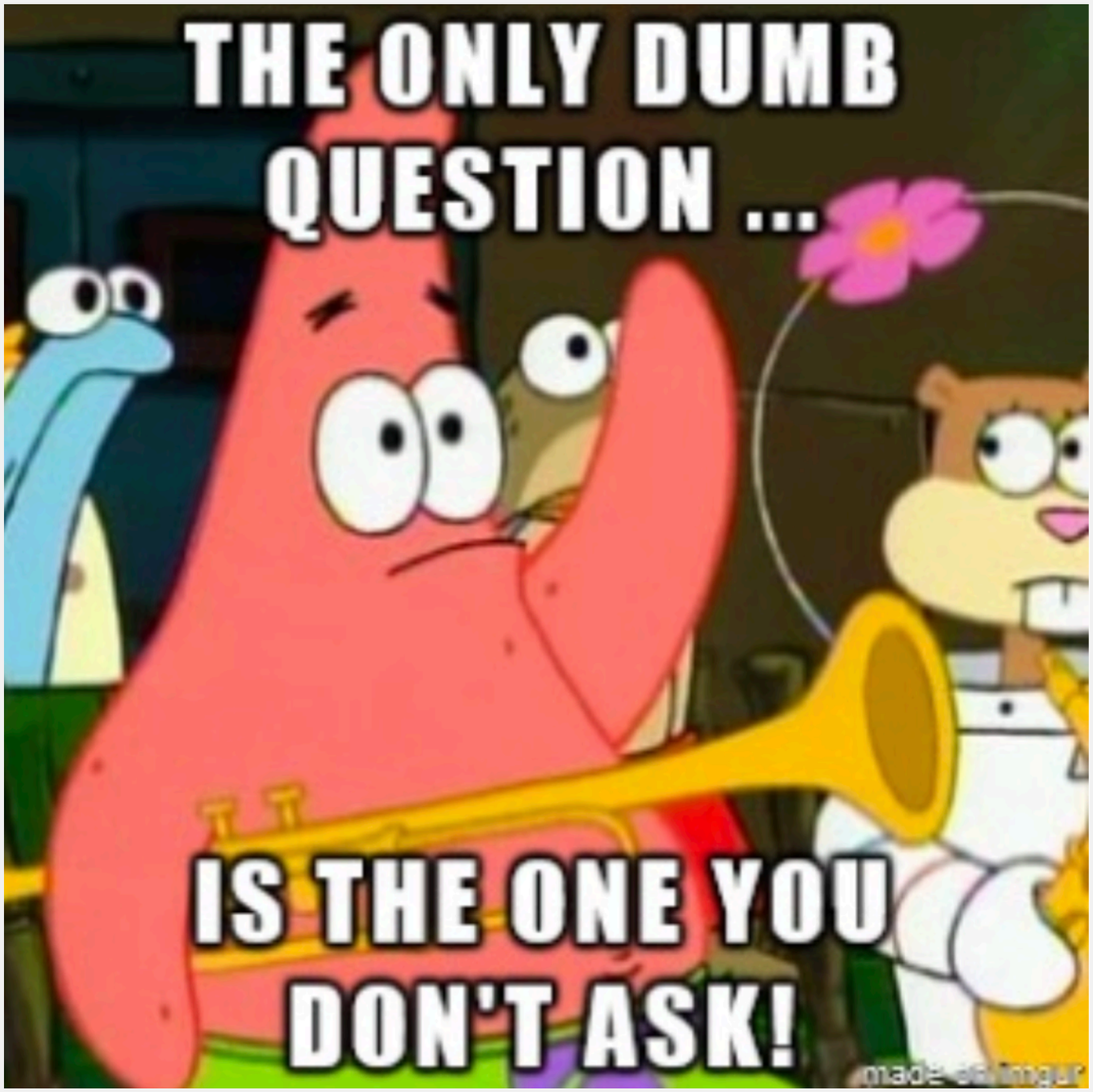
Knowledge Type	Description (Excerpt)
Functionality and Behavior	Describes what the API does (or does not do) in terms of functionality or features. Describes what happens when the API is used (a field value is set, or a method is called).
Concepts	Explains the meaning of terms used to name or describe 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.

Maalej, W., & Robillard, M. P. (2013). Patterns of knowledge in API reference documentation. IEEE Transactions on Software Engineering, 39(9), 1264-1282.

Know Your Audience

- 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 knowldge
- 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

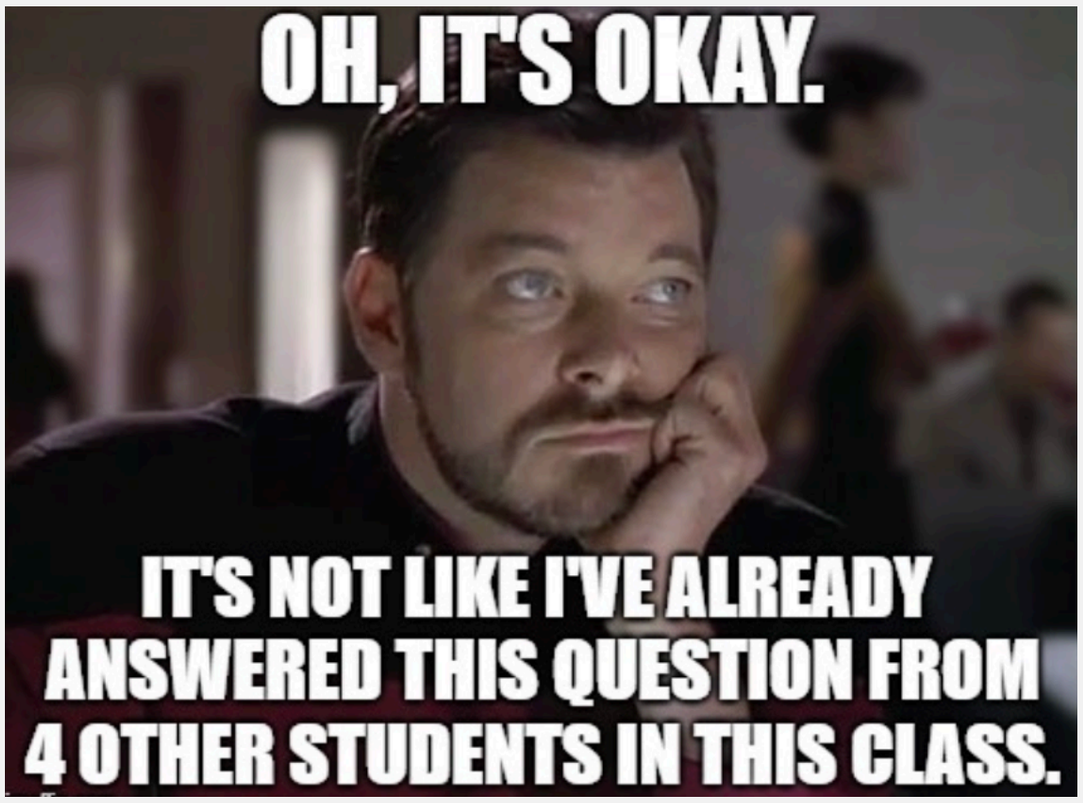
```
import re
print("Welcome To This Software Made By Aaron!")
word = raw_input("Enter Your Words: ")
Check = 0
Right = 0
Length = len(word)
while True:
    if Right == 1:
        if Length < Check:
            Check = Check + 1
            print(Check)
    if Length == Check:
        Right = 1

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 ___.

Avoid Duplication



RESEARCH-ARTICLE

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

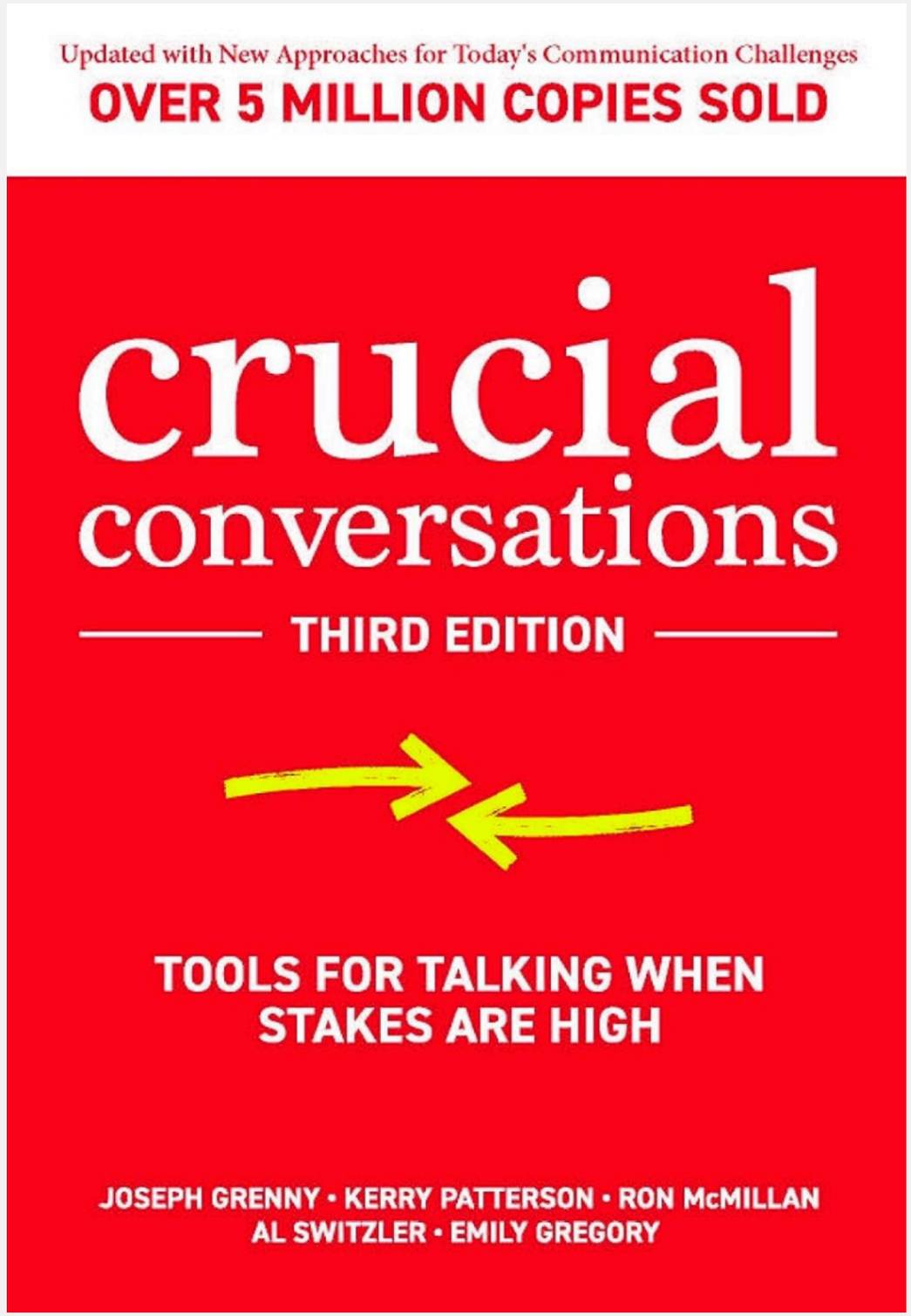
748 Accesses | **19** Citations | **1** Altmetric | Metrics

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

Resolving Conflicts



Communication!

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