Software Development in Practice

How software engineering teams really work?

William Chan
Lead Platform Engineer, 605.tv, Capital One, FreeWheel (Comcast)

Course Objective

- Learn real life software engineering processes outside of the classroom
- Practice real life software engineering practices
- Become software engineers!

2

Course Breakdown

1. Software Development Methodologies
2. Product Development and Requirements
3. Software Design
4. Software Implementation

3

Project Breakdown + Technical Requirements

Choosing a project that will help you succeed in this course for grades sake

- Use a relational database or a non-relational database if you can prove a need for it
- A web or desktop application that stores and reads data from a database
- A working demo by the end of the semester that will be presentable to potential employers

4

Presentation Breakdown

1. **Product**: Discuss the overall product mission and what your application intends to solve; may include business idea and notes
2. **Planning**: Milestones for the application from the beginning of the semester to the end of the semester and how you came to choosing a software development methodology
3. **Requirements**: List the requirements and reasons behind why that constitutes your MVP
4. **Technical Design**: UML diagrams and additional slides that demonstrates the reason to the design
5. **Tests**: What tests will you have in place to ensure the quality of your application
6. **Demo**: Be able to present a partial or preferably a full demo; if its a partial demo, explain why the project could not be completed as is
7. **Retrospective**: knowing what you know after having implemented the project, what would you have changed or think you can do better?
8. Questions: The ability to defend any section of the presentation

Note: there will be assignments that will be the milestones that can be laid out in the presentation

Work Environment

- Project to resemble a real work environment
- All of you will be broken into teams that work on different projects
- All of you will define the product, gather requirements, design the application, implement the application, test the application and demo the application
- All of you will carry out 360 reviewing each other to make sure that all of you carry the weights of the team

6

Software Development Overview

7

Typical Project Roles

- Product Manager/Owner - someone who is close to the users and knows the problem that the application solves; the stakeholder in the most general sense
- Software Engineers - responsible for technical design and implementation of the product
- Project Managers/SCRUM Master - person responsible for removing obstacles that block software engineers from delivering the product
- Program Manager - typically in larger organizations to manage teams that break down into multiple products
- DevOps Engineer - engineers responsible for the automation of the environment and infrastructure the application is deployed to
- QA Engineering/Software Engineering in Test - slowly fading away in favor of software engineers being accountable for their own work

What does a generic software development process look like?

1. Define the product
2. Gather requirements
3. Design the application
4. Implement the application
5. Test the application
6. Release the application
7. Gather user feedback

9

Software Methodologies

- Waterfall - lots of time spent thinking about the design and make sure all edge cases are covered, aim to complete the implementation in one cycle and then demo to client
- Kanban - no defined timeline but stories or tickets are groomed constantly to ensure that priorities are always on the top of the backlog
- SCRUM, SAFe, LeSS and other variants - like kanban but in the form of sprints to measure that given a same length cycle, determine the velocity at which the team completes tickets and stories
- Lean - based on doing the minimal viable set every time and work to eliminate wasteful work

Note: More to come in future slides

10

Software Organization

11

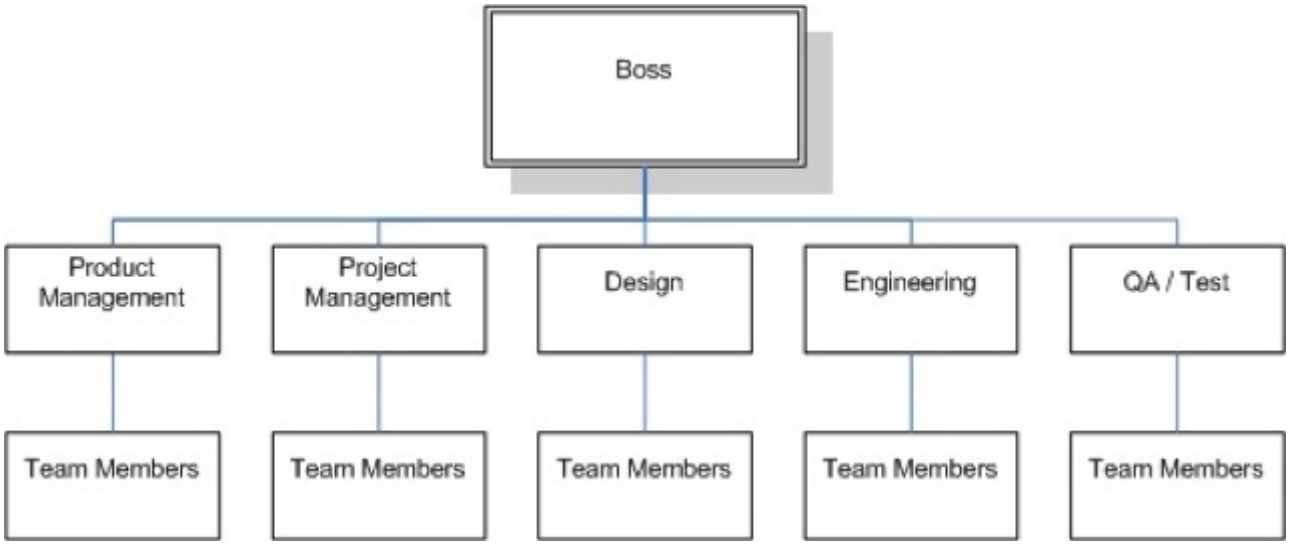
Software Engineering Organization Structure

Structures and hierarchies of organizations often prioritizes certain capabilities over others

- Functional - groups of people under functions such as front end developers, back end developers, etc
- Divisional - cross functional teams where there may be front end developers, back end developers, product managers, etc
- Matrix - organized around different products that are delivered
- Flat - everyone reports to the CEO or where have minimal layers of managers

12

The Functional Organization



Source: http://itsadeliverything.com/using-a-product-led-matrix-in-lean-agile

Why the functional organization?

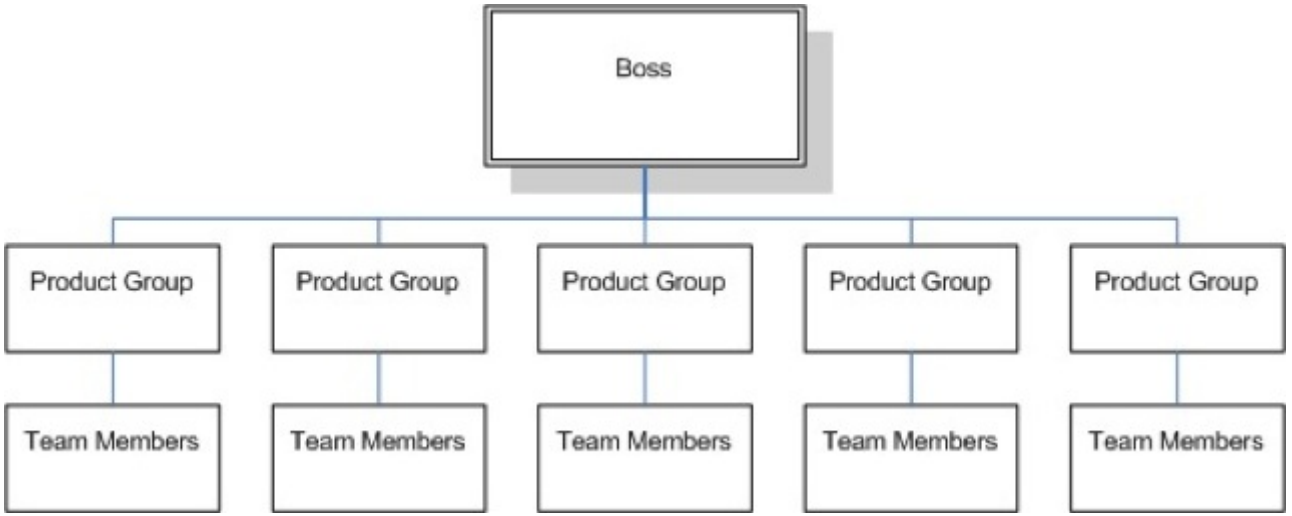
Advantages

- People are specialists
- Recruiting and performance is vetted on the functional aspect
- Easier to allocate specialist resources to different teams
- Training can be more uniform about peoples' functions

Disadvantages

- Staff are not project focused and are not dedicated to teams
- May have a team function bias

The Divisional Organization



Source: http://itsadeliverything.com/using-a-product-led-matrix-in-lean-agile

Why the divisional organization?

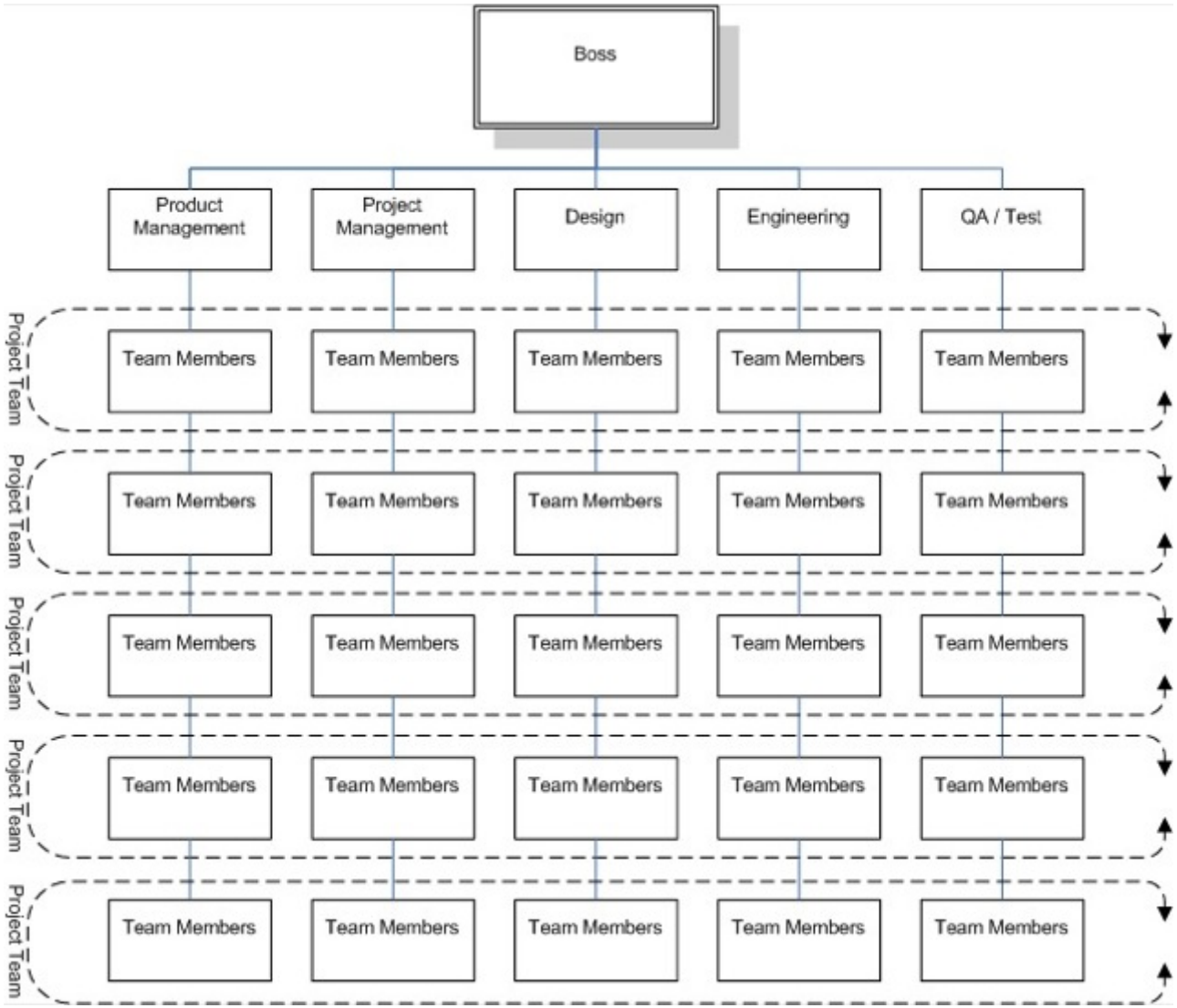
Advantages

- Teams are dedicated to products that deliver business value
- Product knowledge is maximized
- Product output is maximized

Disadvantages

- Team members are confined to a given product
- Emphasis given on product rather than functional skills
- Some products might require developers to have strong product knowledge

The Matrix Organization



Source: http://itsadeliverything.com/using-a-product-led-matrix-in-lean-agile

Why the matrix organization?

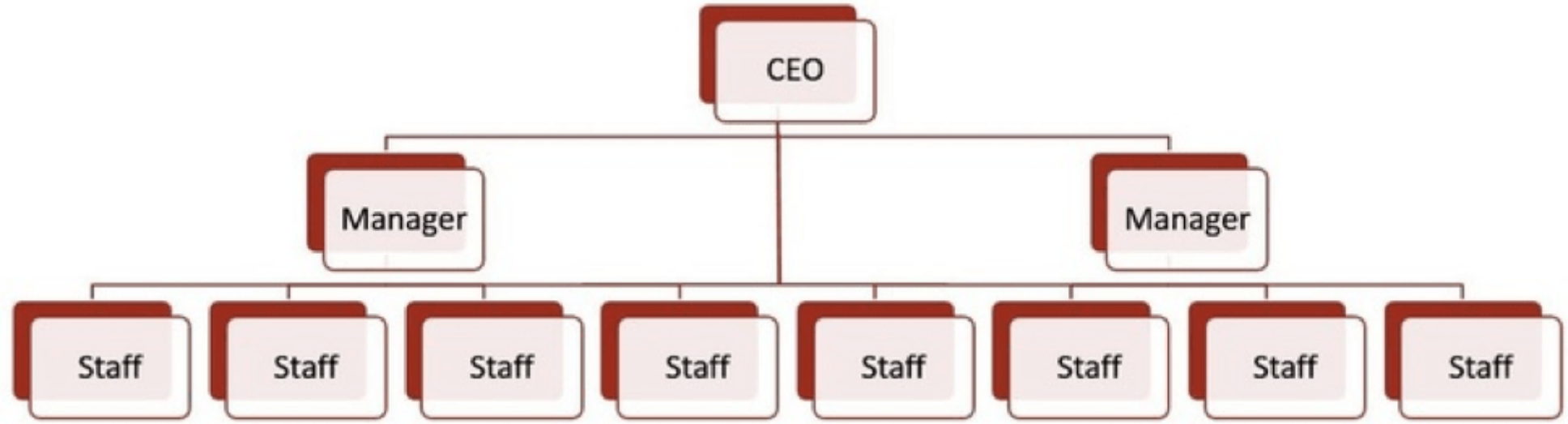
Advantages

- Functional expertise
- Divisional expertise
- Teams can be broken down in terms of need to respond flexibly

Disadvantages

- Balancing between functional and divisional priorities could be difficult
- Performance review of individual groups may be difficult due to individuals spanning multiple groups
- Multiple supervisors create congestion and difficulty

The Flat Organization



Source: https://pingboard.com/org-charts/evolution-org-charts

Why the flat organization?

Advantages

- Communication to vision leaders are more easily accomplished
- Everyone has equal ownership of the product, their responsibilities and commitments to the organization

Disadvantages

- Visionary leaders may overloaded with managing and communicating with implementers
- Decision making and reaching consensus could be more difficult
- Requires more self-direction of team members
- Achieving accountability may be difficult

Software Engineering Career Tracks

- Managerial - people manager, generally responsible for the well being of the team and growing people
- Technical - growing people by technical means and ensure technical delivery of a project

The Managerial Track

Focuses on developing people and ensure timely deliverables. Below are some example position titles

- Engineering Manager
- Director of Engineering
- VP of Engineering
- SVP of Engineering
- CTO

The Technical Track

Focuses on the big picture view of a system that may spans multiple teams. Below are some example position titles

- Software Engineer
- Senior Software Engineer
- Lead Engineer
- Staff Software Engineer
- Principal Softwware Engineer
- Software Architect
- Technical Architect

Software Engineering Culture

- Distributed teams - remote workers organized in small cross functional teams usually
- Pair programming - often two engineers that sit together, one being the driver, telling the other how to implement a feature
- Mob programming - the whole team sits in a room to design and build a feature together
- Extreme Programming - a combination of pair programming but largely taking a more test driven approach

Why Distributed Teams?

Advantages

- Greater access to engineers located anywhere
- Salaries paid could be minimized if your team is not based in a large metro area

Disadvantages

- It could be an HR logistics nightmare
- Communication and team coordination has increased overhead
- Managing and aligning team expectations could be difficult

Why Pair Programming?

The driver in the pair programming practice makes decisions and drives the implementation of a feature while the navigator reviews the code as its being written. It is better to switch up the roles every 25 minutes to keep both engineers engaged otherwise the navigator may slowly disengage.

Advantages

- Two people can collaborate and design a feature
- Additional person to check for mistakes and bugs
- Transfer of knowledge from a senior engineer to a junior engineer

Disadvantages

- The more senior engineer may not be learning or may be disengaged
- The more junior engineer may rely too much on the more senior engineer

Why Mob Programming?

Advantages

- Engineering team can get transparency and understand the whole entire system
- Design can be improved since it will be well understood by the entire team
- Works well when there is a fair amount of uncertainty in the feature that is being delivered

Disadvantages

- Requires everyone in a single room
- Some engineers may disengage as in pair programming
- Doesn't work well if the feature is well understood
- There may be equal priority items that needs attention

Why Extreme Programming?

Advantages

- Same as pair programming where knowledge transfer is present and bugs can be minimized with someone reviewing as code is being written
- Tests are often more meaningful and acts as feature specifications

Disadvantages

- Rigid approach that will take time getting used to
- Work that has a fair amount of uncertainty is often difficult to write tests for

Software Development Workflow

29

Product Management & Project Management Workflow

1. Collaborate with software engineers to decide what the breakdown of the tasks are
2. Define the **acceptance criteria** for which a product will accept a feature as complete
3. Software engineers estimate the complexity or the time of each story
4. Product managers and project managers define the priorities with the estimates in mind

30

Software Development Workflow

1. Technical design - technical approach to solving the problems; high level components and how they interact with each other
2. Implementation - code is written and logic is verified
3. Open a pull request - became popular through the adoption of GitHub with distributed version control
4. Code review - ensures standards and quality
5. Merge to a branch to be deployed in a QA or staging environment that mimics production
6. Feature validated by the product owner and get pushed to production through an approval process

Software Development Tools

- Version Control: Git, SVN, CVS, Mercurial, Bazaar
- Repository Management: GitHub, GitLab, Bitbucket
- Project Management: Jira, Pivotal Tracker, Trello, Asana
- UML Diagrams: PlantUML, yUML, Mermaid
- Text Editors/IDE: Intellij IDEA, Eclipse, Sublime Text 2/3, Atom, Visual Studio Code
- Continuous Integration & Delivery: TravisCI, Jenkins, CodeShip, CircleCI, Weave

Why Version Control?

- Helps keep track of changes that are made to the system
- Helps people work together as a team where team members can see the changes other people make
- Helps debug to see when a bug was introduced

33

Why Respository Management?

- To manage the contents that makes your application run
- To gain a higher level of transparency than the command line tool
- To allow code reviews to happen more in tandem
- To enable team members to coordinate code changes together
- To allow extensions such as plugins and hooks to streamline the building and testing of your application

34

Why Project Management Software?

- To better coordinate between team members what tasks should be worked on
- For managers to keep track of the progress of tickets
- To better break down tasks into manageable and completable pieces of work

Why UML Diagrams?

- To describe the high level implementation of the application and help other developers understand the core components
- To help developers break down tickets and stories into small componentized pieces
- To force developer to think and understand of interactions and state of different components of the system

Why Certain Text Editors or IDEs?

- To allow developers to easily spot syntax errors with syntax highlighting
- To allow developers to explore code with features such as go to implementation, go to definition
- To automatically help developers write and think about code

Why Continuous Integration & Delivery Tools?

- To help automate the running of tests and deploying to different environments so that product managers can check work

38

Git Cheatsheet

Below are some commands that you can use for your software development workflow

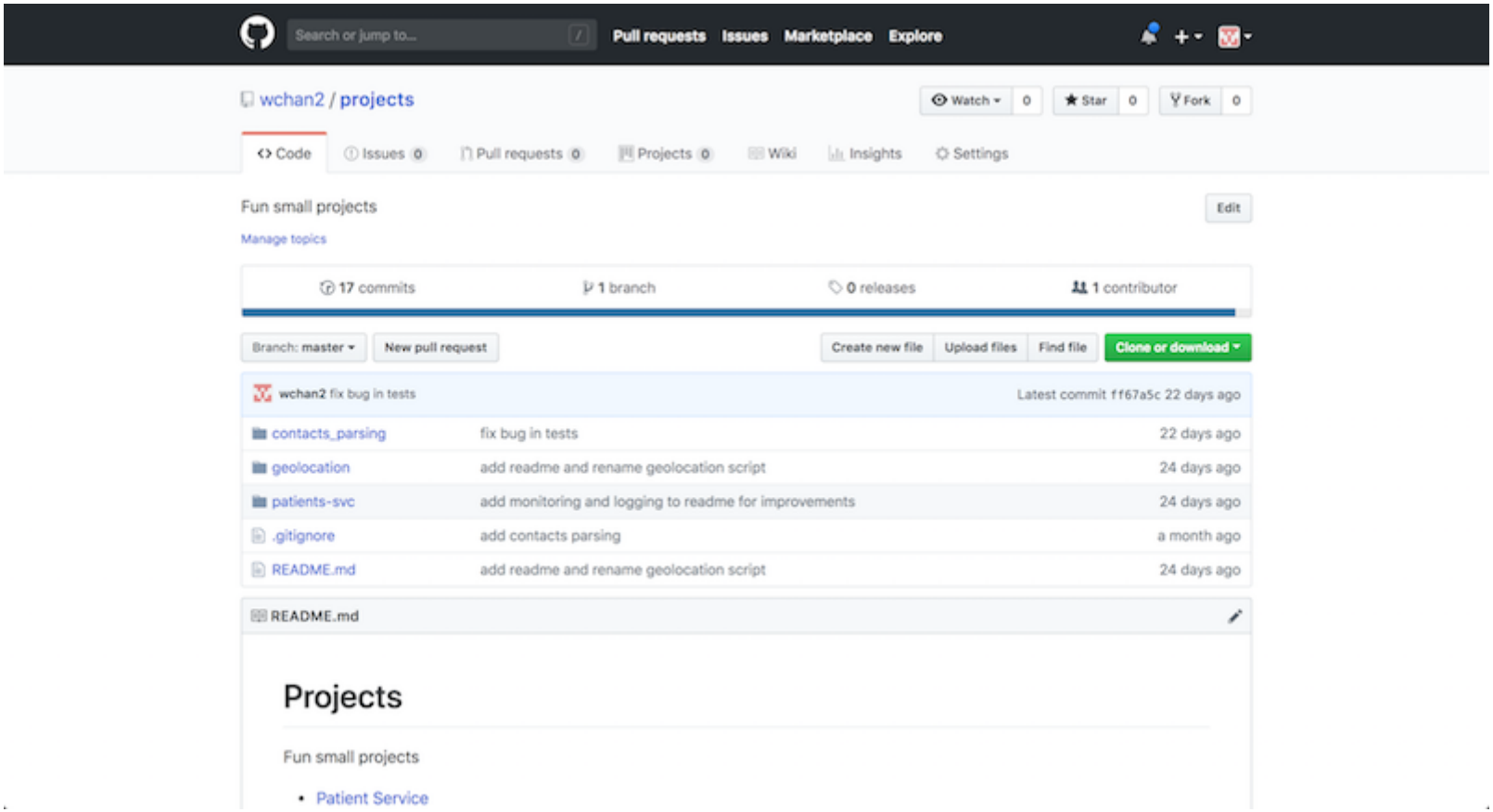
```
git clone <name of your repo>          # pull down from an initiated project
git pull origin master                 # assuming you have a branch called master
git checkout -b <name of your branch> # checks out a new branch
git push origin <name of your branch> # allows you to create a pull request
```

Note: there are some more sophisticated command but this will do for now

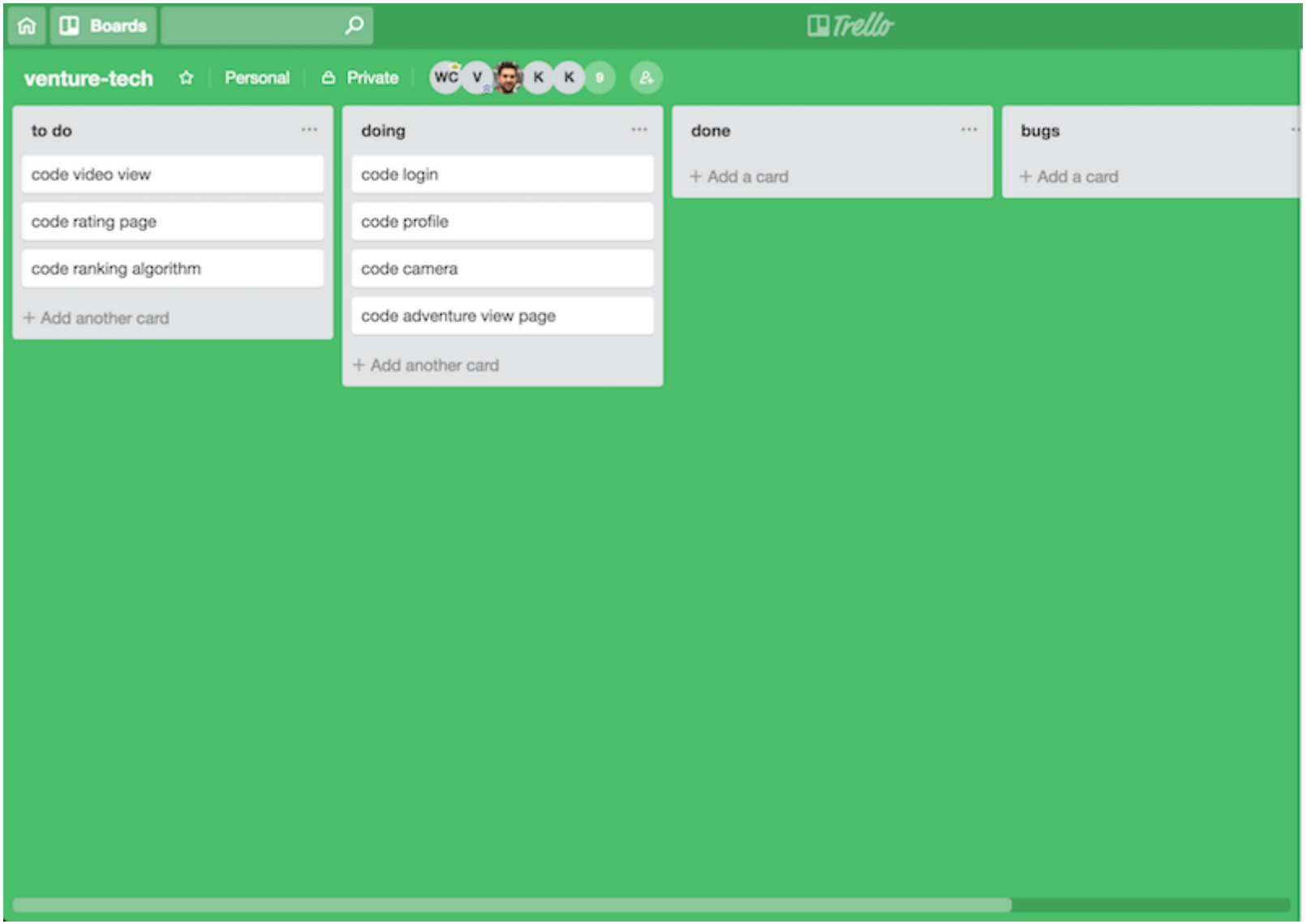
Git Best Practices

- Checkout branches that is specific to a ticket or story number that is being worked on to identify and relate to
- Configure the `git` command line tool to include your name so that people can identify who made the change
- Add meaningful commit messages that communicate to other team members on what the actual change is
- Keep commit messages focused on the change for the feature and less on the technical aspects
- Try to keep commits to a minimal for a given story or ticket

GitHub


Search or jump to...
Pull requests
Issues
Marketplace
Explore
wchan2 / projects
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Fun small projects
Edit
Manage topics
17 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
wchan2 fix bug in tests
Latest commit ff67a5c 22 days ago
contacts_parsing
fix bug in tests
22 days ago
geolocation
add readme and rename geolocation script
24 days ago
patients-svc
add monitoring and logging to readme for improvements
24 days ago
.gitignore
add contacts parsing
a month ago
README.md
add readme and rename geolocation script
24 days ago
README.md
Projects
Fun small projects
Patient Service

Trello


Boards
Trello
venture-tech
Personal
Private
WC
V
K
K
9
to do
code video view
code rating page
code ranking algorithm
+ Add another card
doing
code login
code profile
code camera
code adventure view page
+ Add another card
done
+ Add a card
bugs
+ Add a card

Summary

- Software engineering is an iterative and team based process
- There are often two career tracks, the managerial and technical tracks in engineering organizations
- Version control, project management tools, UML diagrams, text editors, and repository management is often used to help software engineering teams be productive and collaborate
- Generally, there are two independent workflows for actually doing the technical work and working with stakeholders to define requirements and priorities

43

Thank you

William Chan
Lead Platform Engineer, 605.tv, Capital One, FreeWheel (Comcast)
wchan@605.tv (mailto:wchan@605.tv)
http://linkedin.com/in/wchan2 (http://linkedin.com/in/wchan2)