Intro To Process

Milestones, Estimation, Planning

17-313 Fall 2024

Foundations of Software Engineering

https://cmu-313.github.io

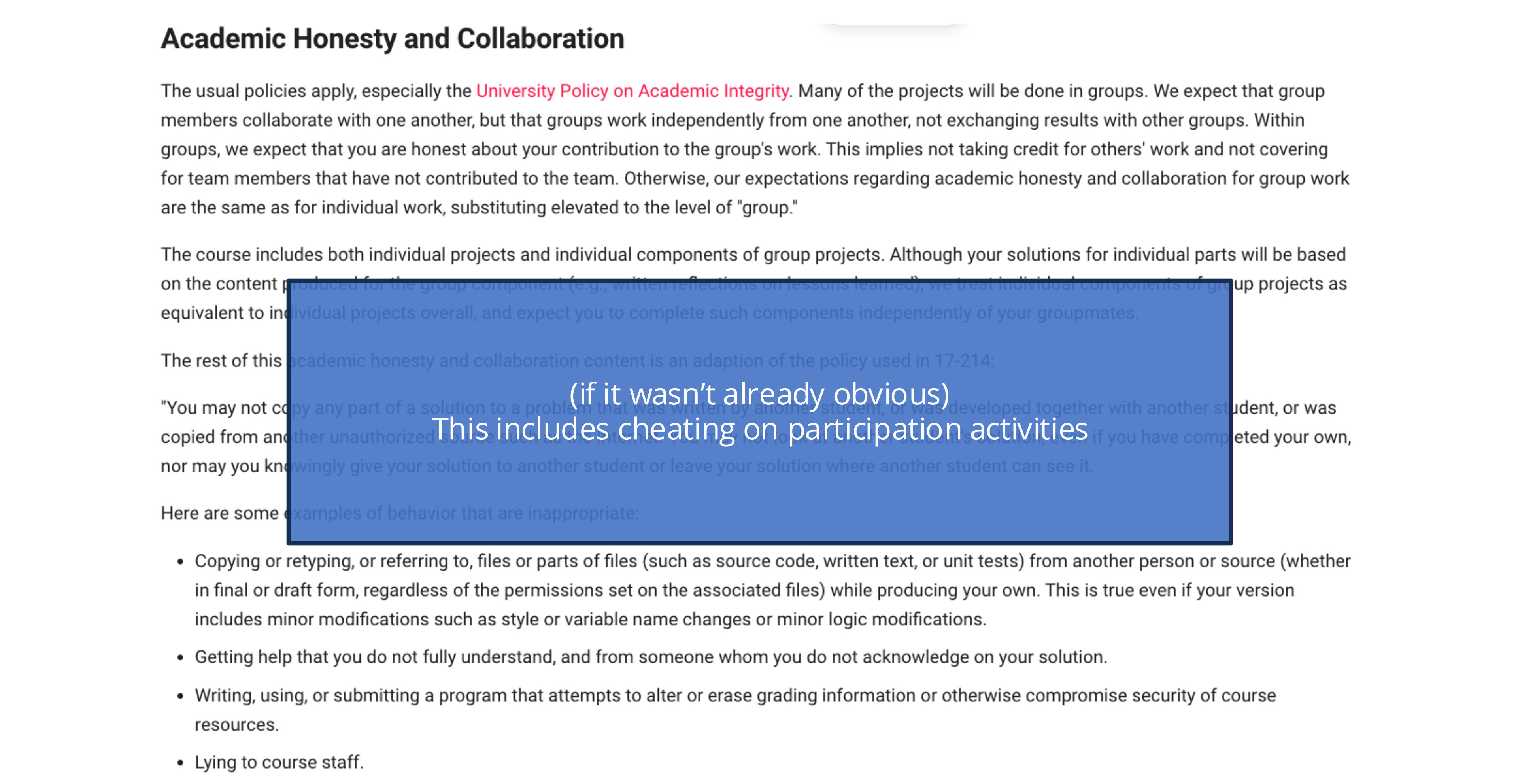
Michael Hilton and Rohan Padhye

Administrivia

- Project 2(a) due on Thursday (Sep 12th) at midnight
  - Section D students have until Saturday (Sep 14th) due to a team-assignment mix-up. Sorry about that!
- Meet with your teams!
- Extra credit: Go out with your teams socially.
  - Share a photo/screenshot of your team activity with your TA before Thursday night.

Smoking Section

- Last full row



Academic Honesty and Collaboration

The usual policies apply, especially the University Policy on Academic Integrity. Many of the projects will be done in groups. We expect that group members collaborate with one another, but that groups work independently from one another, not exchanging results with other groups. Within groups, we expect that you are honest about your contribution to the group's work. This implies not taking credit for others' work and not covering for team members that have not contributed to the team. Otherwise, our expectations regarding academic honesty and collaboration for group work are the same as for individual work, substituting elevated to the level of "group."

The course includes both individual projects and individual components of group projects. Although your solutions for individual parts will be based on the content produced for the group component (e.g., written reflections on lessons learned), we treat individual components of group projects as equivalent to individual projects overall, and expect you to complete such components independently of your groupmates.

The rest of this academic honesty and collaboration content is an adaption of the policy used in 17-214:

"You may not copy any part of a solution to a problem that was written by another student, or was developed together with another student, or was copied from another unauthorized source such as the Internet. You may not look at another student's solution, even if you have completed your own, nor may you knowingly give your solution to another student or leave your solution where another student can see it.

Here are some examples of behavior that are inappropriate:

- Copying or retyping, or referring to, files or parts of files (such as source code, written text, or unit tests) from another person or source (whether in final or draft form, regardless of the permissions set on the associated files) while producing your own. This is true even if your version includes minor modifications such as style or variable name changes or minor logic modifications.
- Getting help that you do not fully understand, and from someone whom you do not acknowledge on your solution.
- Writing, using, or submitting a program that attempts to alter or erase grading information or otherwise compromise security of course resources.
- Lying to course staff.

(if it wasn't already obvious)
This includes cheating on participation activities

Today’s Learning Goals

- Recognize the importance of process
- Identify why software development has project characteristics
- Understand the elements of Scrum
- Create and evaluate user stories
- Use milestones for planning and progress measurement
- Understand the difficulty of measuring progress

What does this mean?
What else can we do apart from coding?
Processes are **key** concerns.

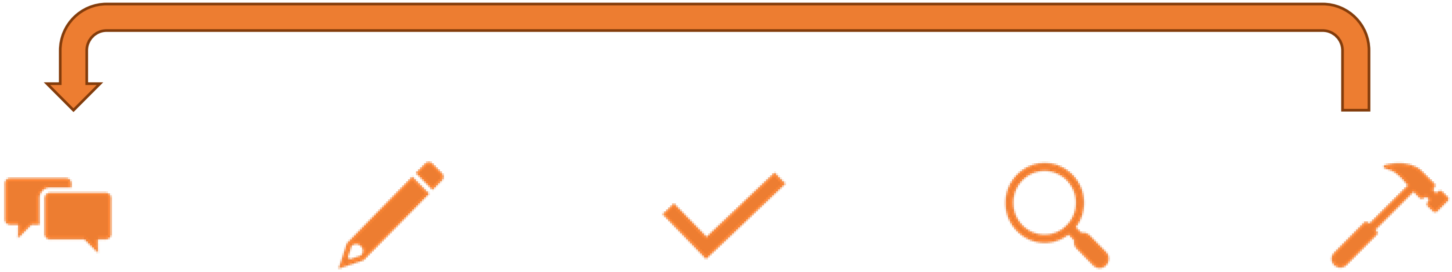
Software Engineering: Principles, practices (technical and non-technical) for confidently building high-quality software.

Software Process

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

Sommerville, SE, ed. 8

How to develop software???



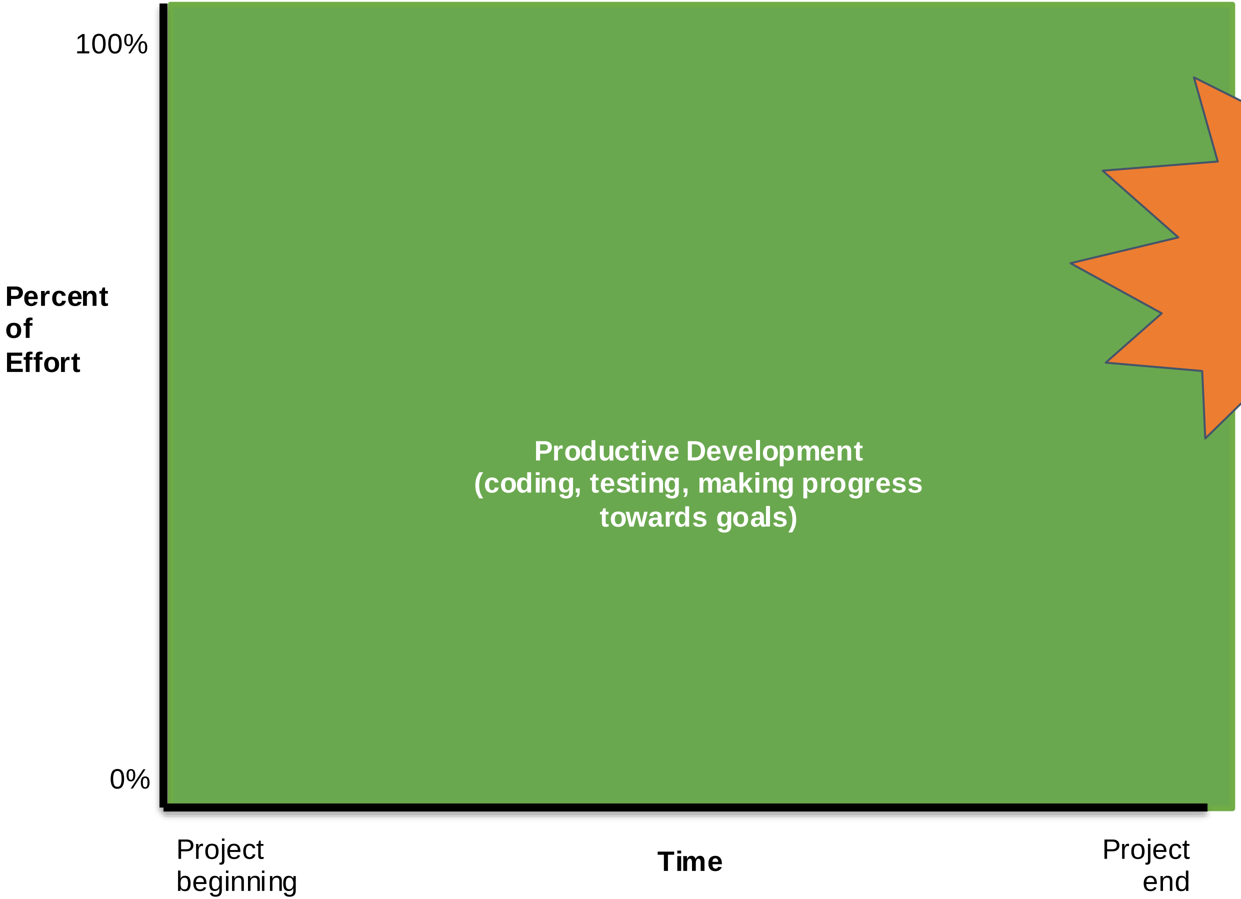
Discuss the software that needs to be written

Write some code

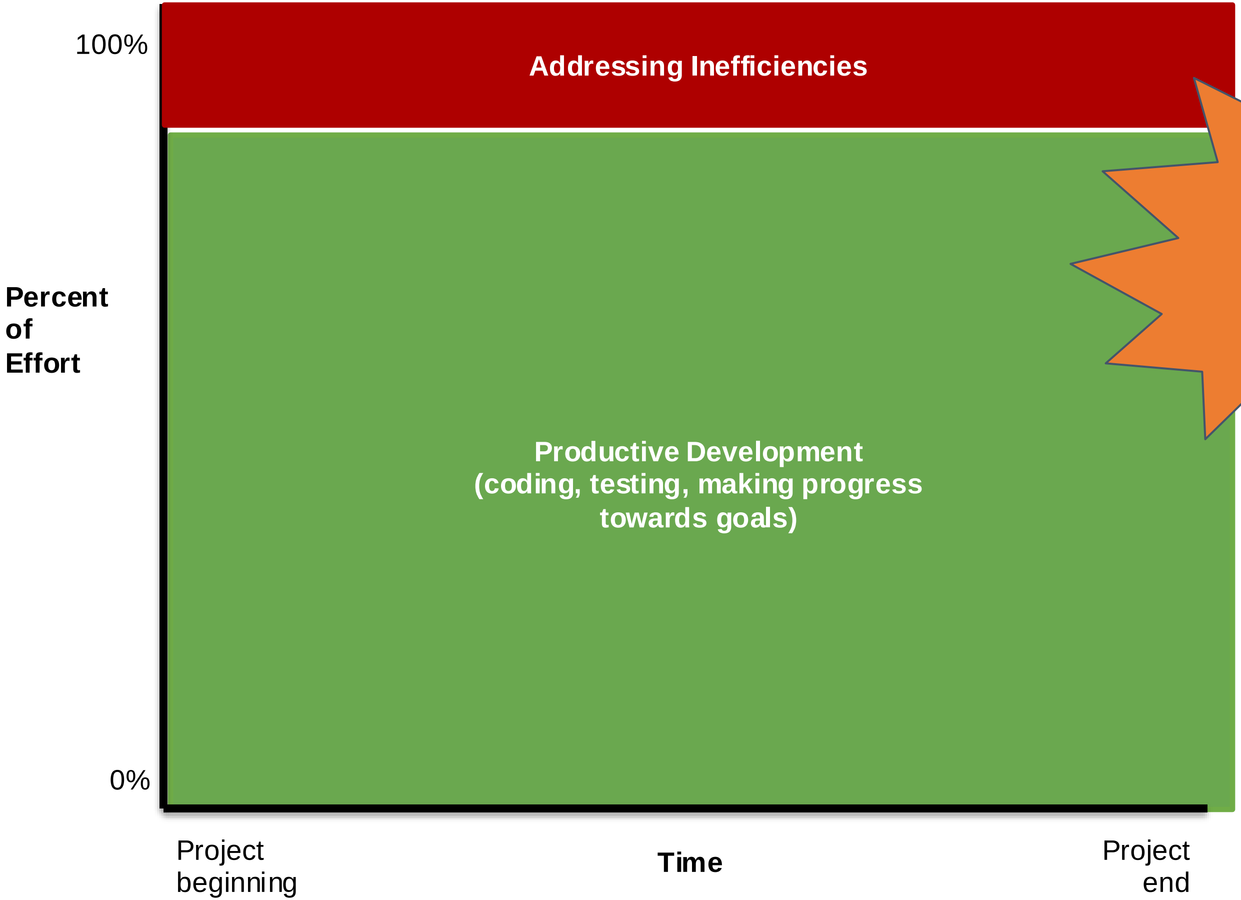
Test the code to identify the defects

Debug to find causes of defects

Fix the defects


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

Idealized?



100%

Addressing Inefficiencies

Percent of Effort

Productive Development
(coding, testing, making progress towards goals)

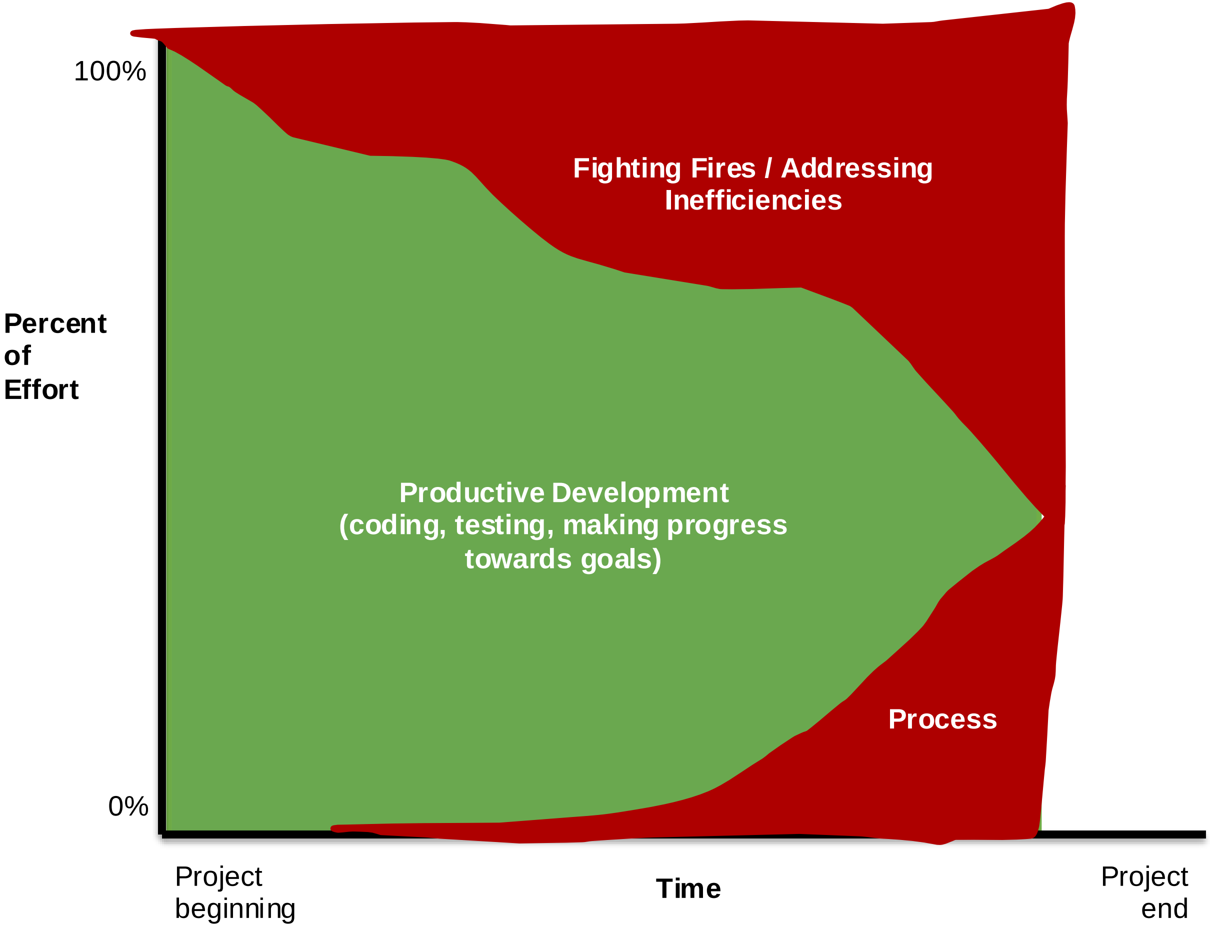
0%

Project beginning

Time

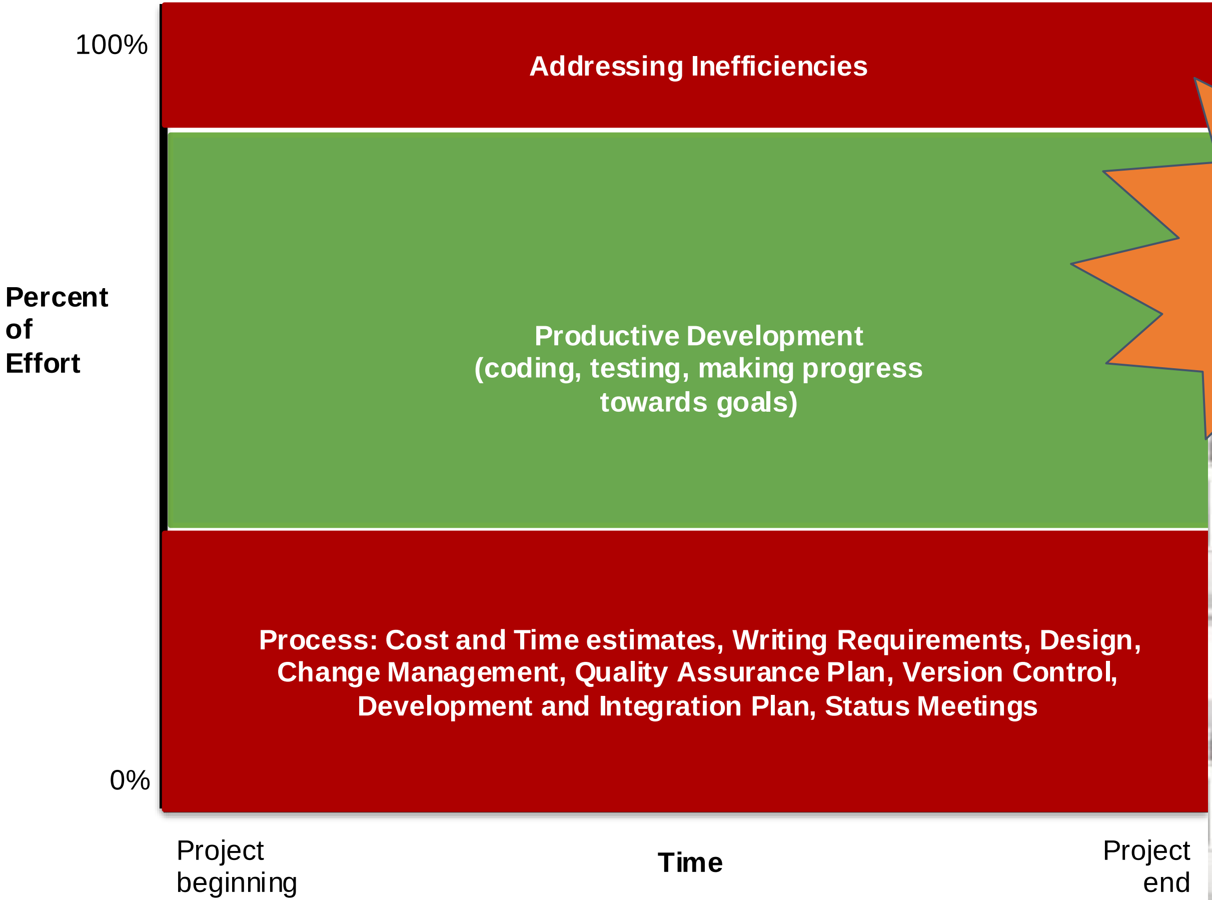
Project end

Idealized?


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

Let’s improve the reliability of this process

- Writing down all requirements
  - Review requirements
  - Require approval for all changes to requirements
- Use version control for all changes
  - Code Reviews
- Track all work items
  - Break down development into smaller tasks
  - Write down and monitor all reported bugs
  - Hold regular, frequent status meetings
- Plan and conduct quality assurance
- Employ a DevOps framework to push code between developers and operations



100%

Percent of Effort

0%

Addressing Inefficiencies

Productive Development
(coding, testing, making progress towards goals)

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

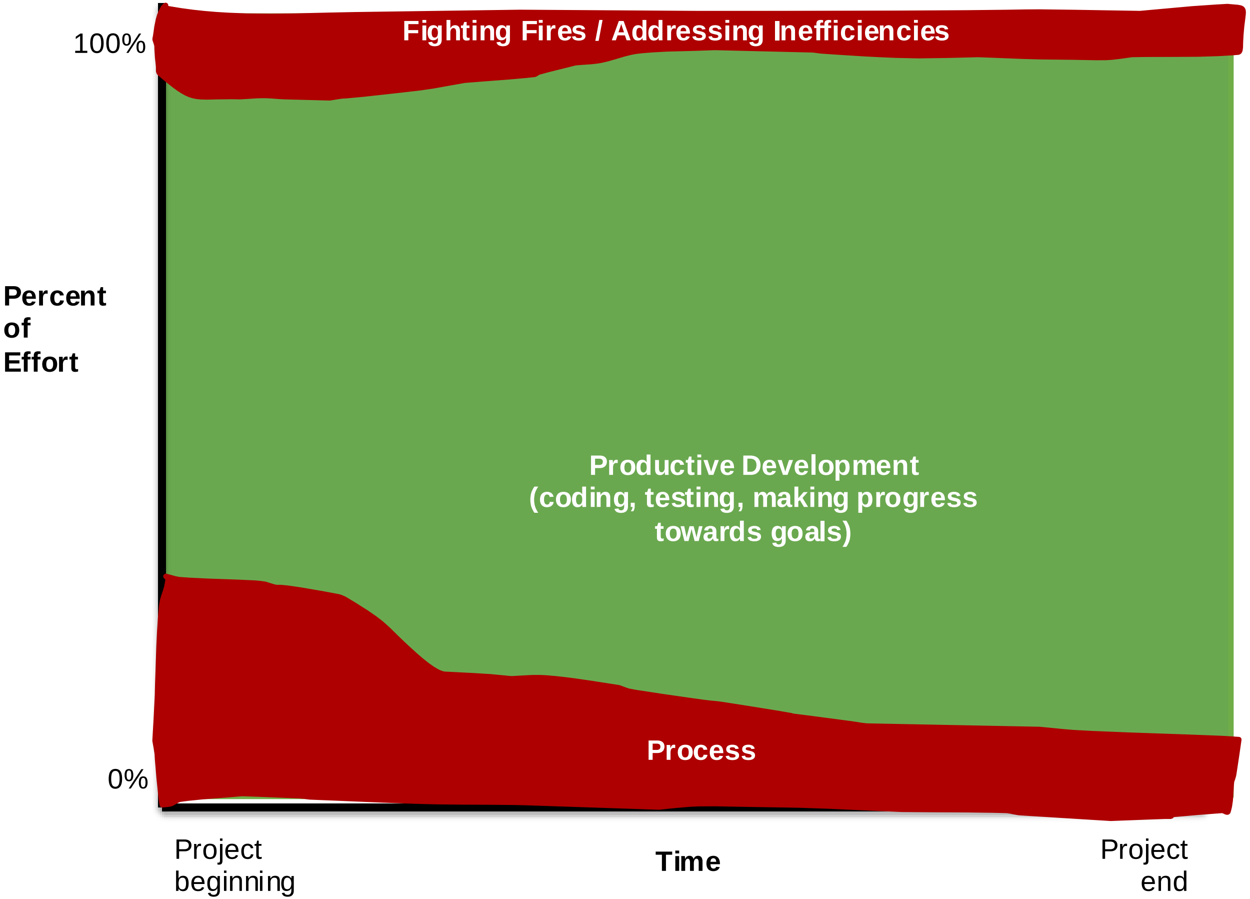
Project beginning

Time

Project end

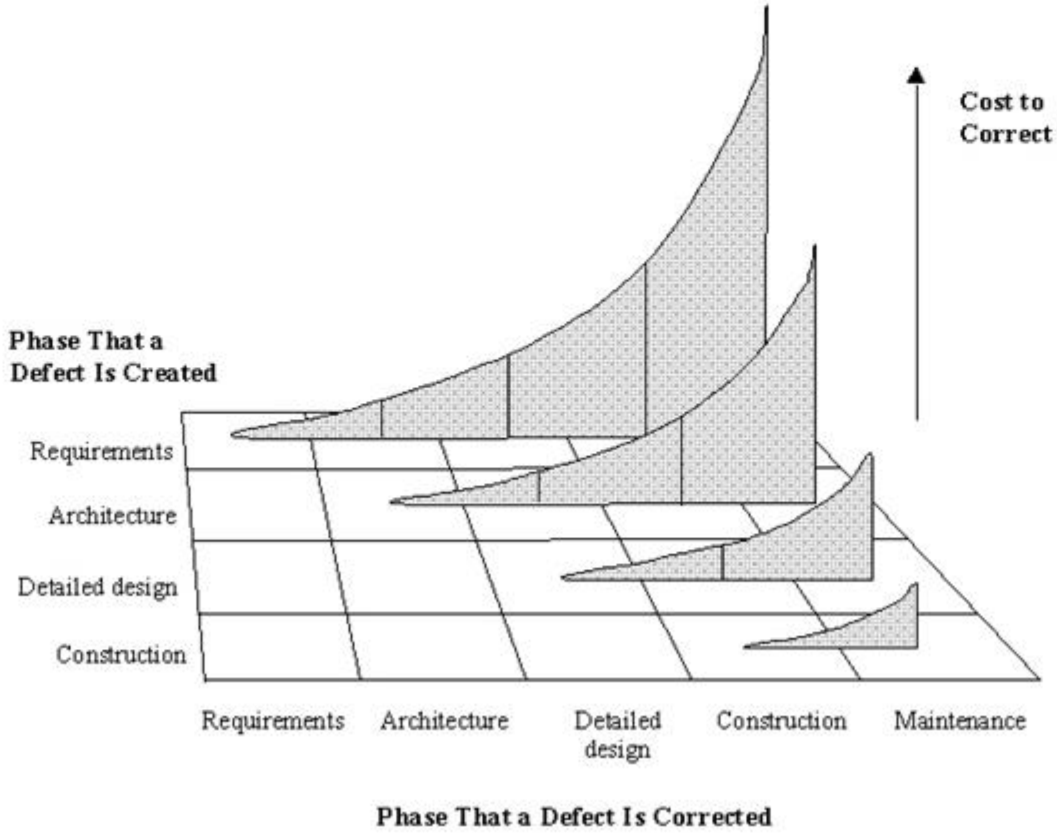
Negative View of Process

ACCEPT JOB AS SENIOR ENGINEER
8 HOURS OF MEETINGS A DAY
Posted in r/ProgrammerHumor
reddit

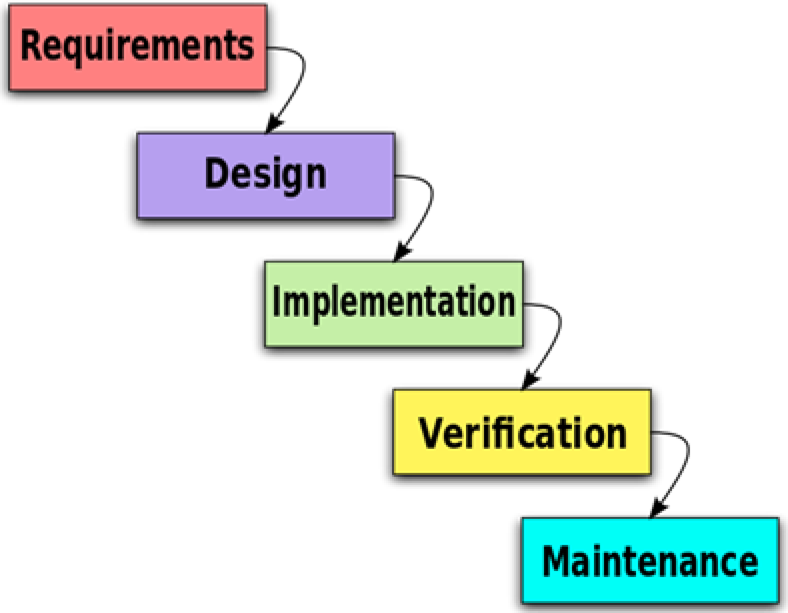


Hypothesis: Process increases flexibility and efficiency

Ideal Curve: Upfront investment for later greater returns

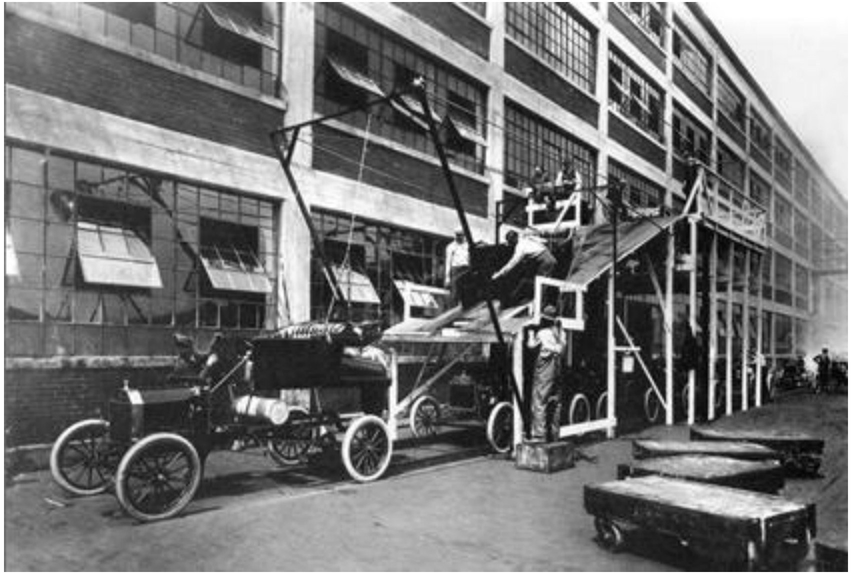

Cost to Correct
Phase That a Defect Is Created
Requirements
Architecture
Detailed design
Construction
Requirements
Architecture
Detailed design
Construction
Maintenance
Phase That a Defect Is Corrected

Waterfall model was the original software process



Waterfall diagram CC-BY 3.0 **Paulsmith99 at en.wikipedia**

… akin to processes pioneered in mass manufacturing (e.g., by Ford)



Lean production adapts to variable demand

Toyota Production System (TPS) Late 1940s

Build only what is needed, only when it is needed.

Use the “pull” system to avoid overproduction. (Kanban)

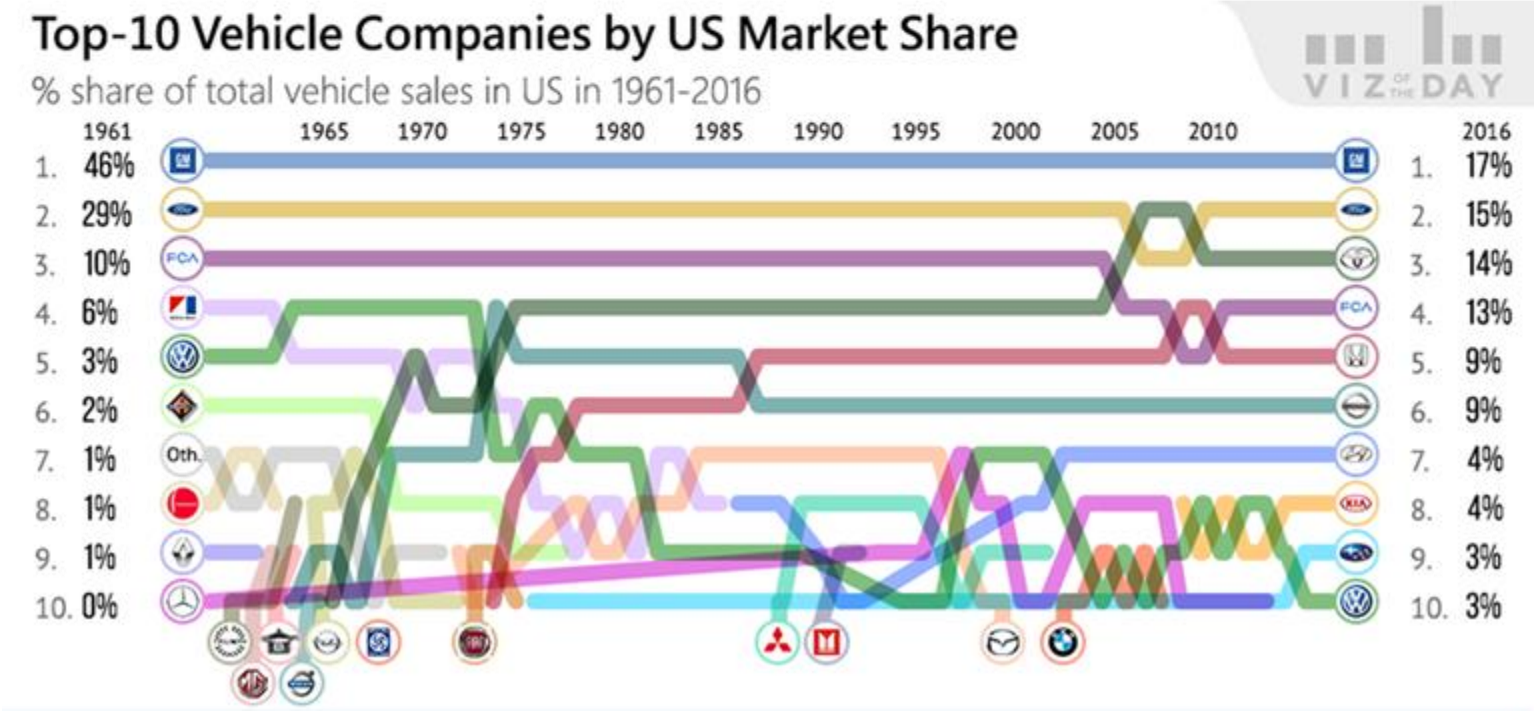
Stop to fix problems, to get quality right from the start (Jidoka)

Workers are multi-skilled and understand the whole process; take ownership

Enabling teams to have autonomy and control to change/improve quickly/continuous improvement (kaizen)

Taiichi Ohno

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



From TPS to Agile

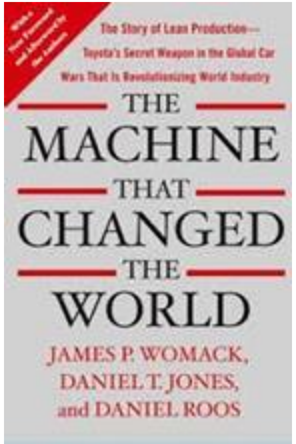
The new new product development game

Stop running the relay race and take up rugby

Hirotaka Takeuchi and Ikujiro Nonaka

[illegible]

1986



With a New Foreword and Afterword by the Authors

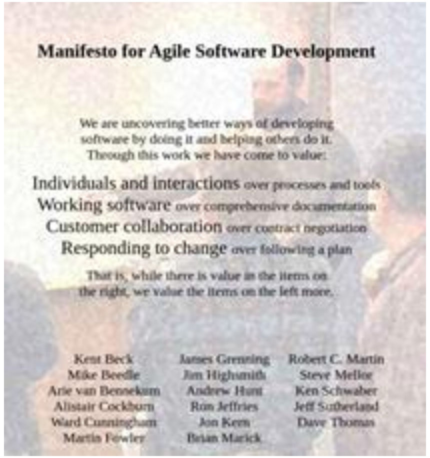
The Story of Lean Production— Toyota's Secret Weapon in the Global Car Wars That Is Revolutionizing World Industry

THE MACHINE THAT CHANGED THE WORLD

JAMES P. WOMACK, DANIEL T. JONES, and DANIEL ROOS

1990

…



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more.

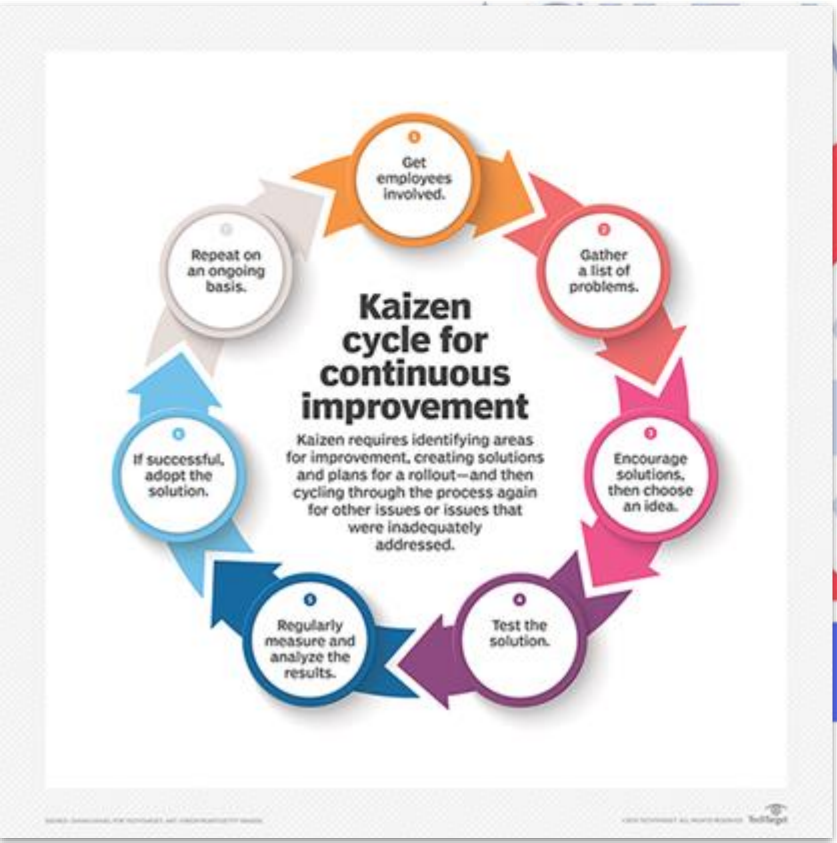
Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

2001

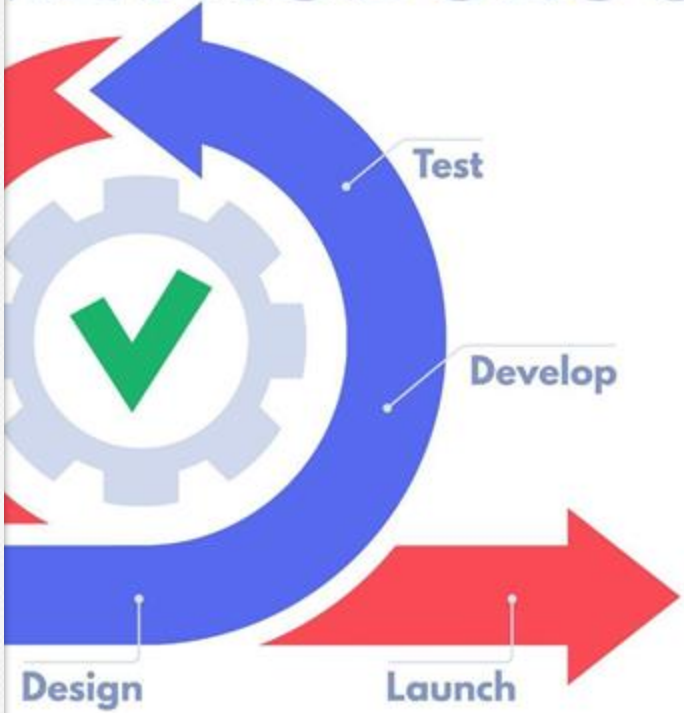
METHODOLOGY



Kaizen cycle for continuous improvement

Kaizen requires identifying areas for improvement, creating solutions and plans for a rollout—and then cycling through the process again for other issues or issues that were inadequately addressed.

1 Get employees involved.
2 Gather a list of problems.
3 Encourage solutions, then choose an idea.
4 Test the solution.
5 Regularly measure and analyze the results.
6 If successful, adopt the solution.
7 Repeat on an ongoing basis.
TechTarget



Scrum

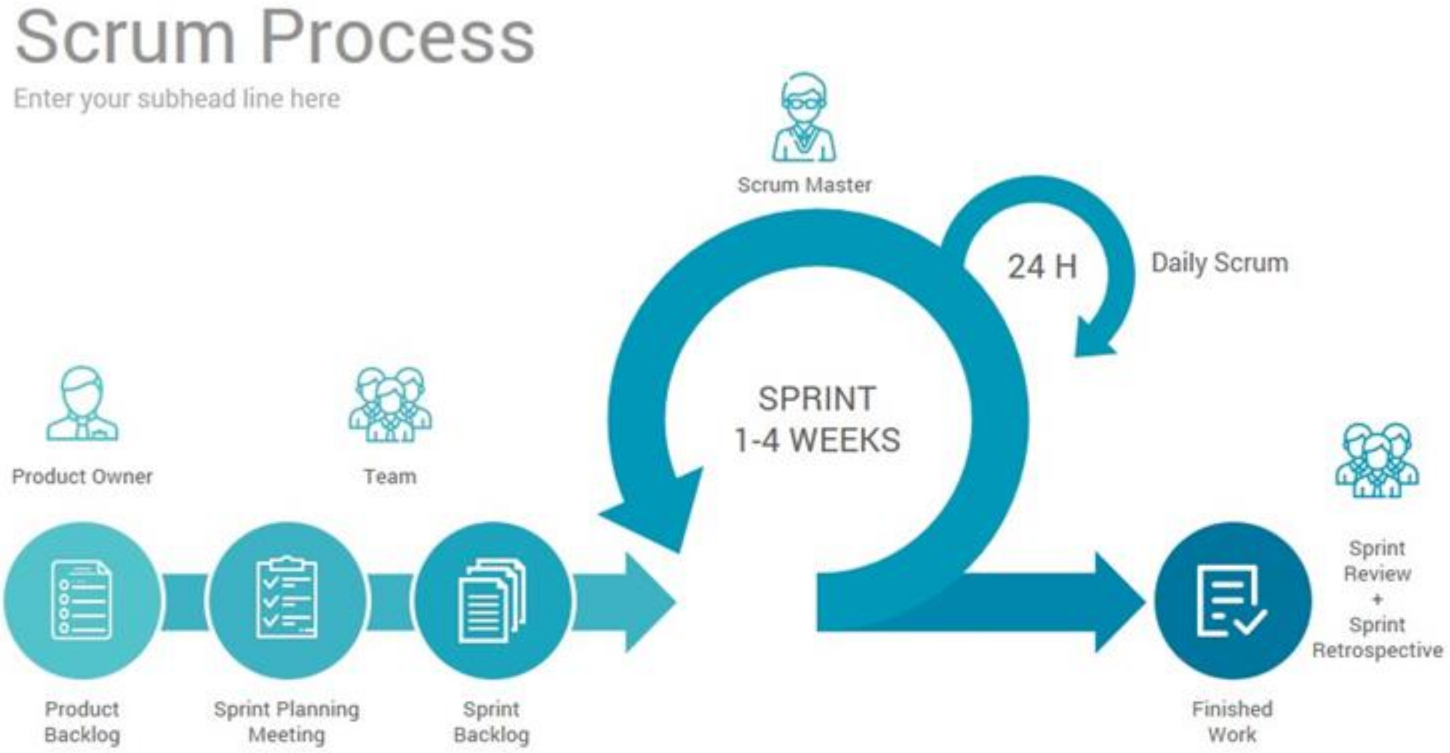
(Only a brief intro)

Agile Software Development with Scrum

red
yellow
green
blue
red
blue
yellow
green
blue

Ken Schwaber Mike Beedle

Elements of Scrum



Backlogs

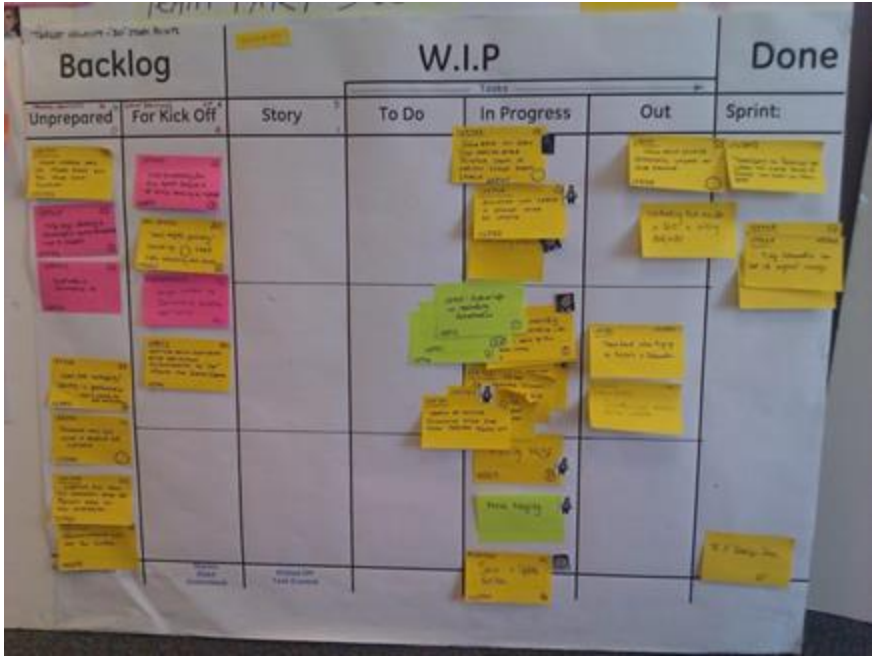
The **product backlog** is all the features for the product

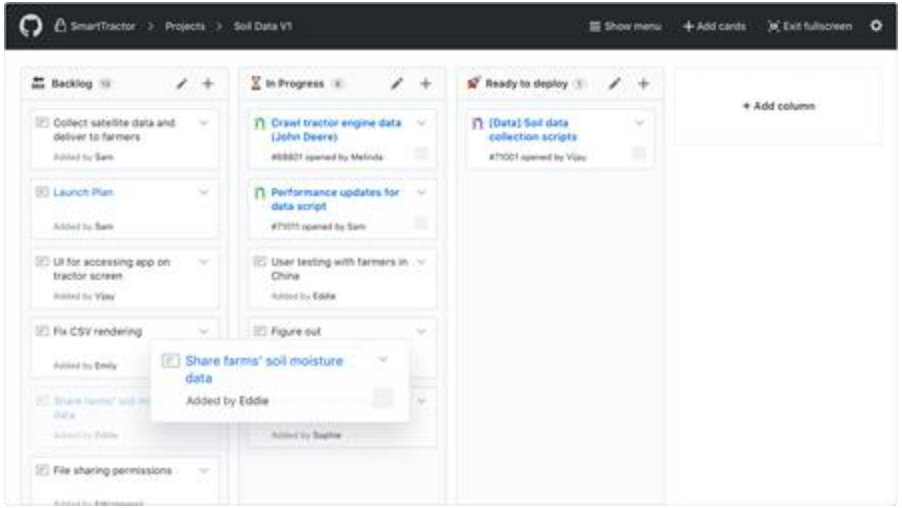
The **sprint backlog** is all the features that will be worked on for that sprint. These should be broken down into discrete tasks:

- Fine-grained
- Estimated
- Assigned to individual team members
- Acceptance criteria should be defined

User Stories are often used

Kanban boards





Backlog 13

- Collect satellite data and deliver to farmers
  Added by Sam
- Launch Plan
  Added by Sam
- UI for accessing app on tractor screen
  Added by Vijay
- Fix CSV rendering
  Added by Emily
- Share farms' soil moisture data
  Added by Eddie
- File sharing permissions
  Added by [illegible]

In Progress 6

- Crawl tractor engine data (John Deere)
  #68801 opened by Melinda
- Performance updates for data script
  #71011 opened by Sam
- User testing with farmers in China
  Added by Eddie
- Figure out
  Added by Sophie

Ready to deploy 1

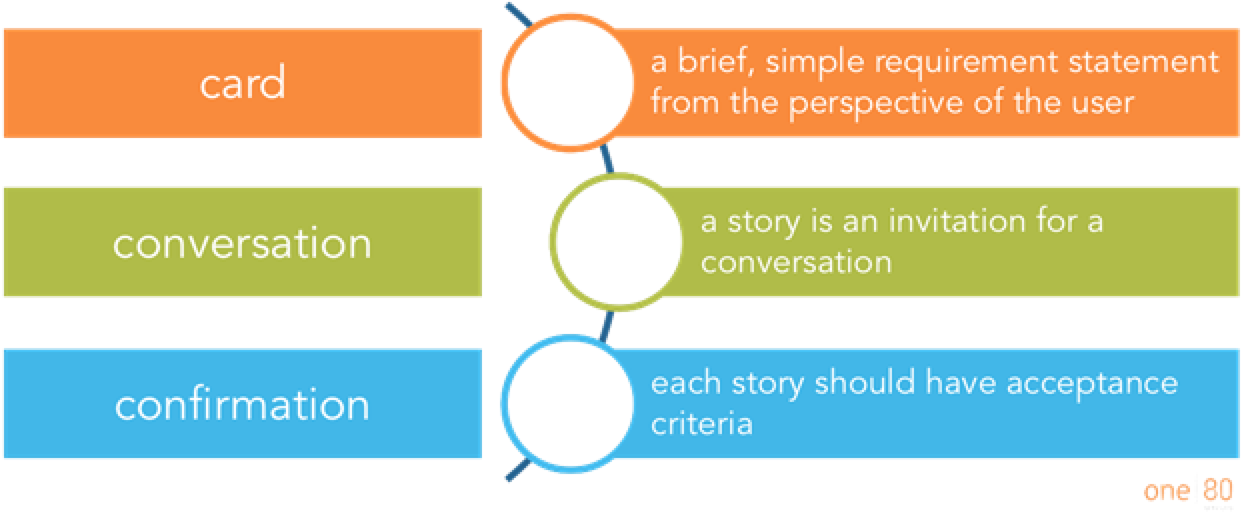
- [Data] Soil data collection scripts
  #71001 opened by Vijay

Share farms' soil moisture data
Added by Eddie

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

Source: http://one80services.com/user-stories/writing-good-user-stories-hint-its-not-about-writing/

User story cards (3”x5”)

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

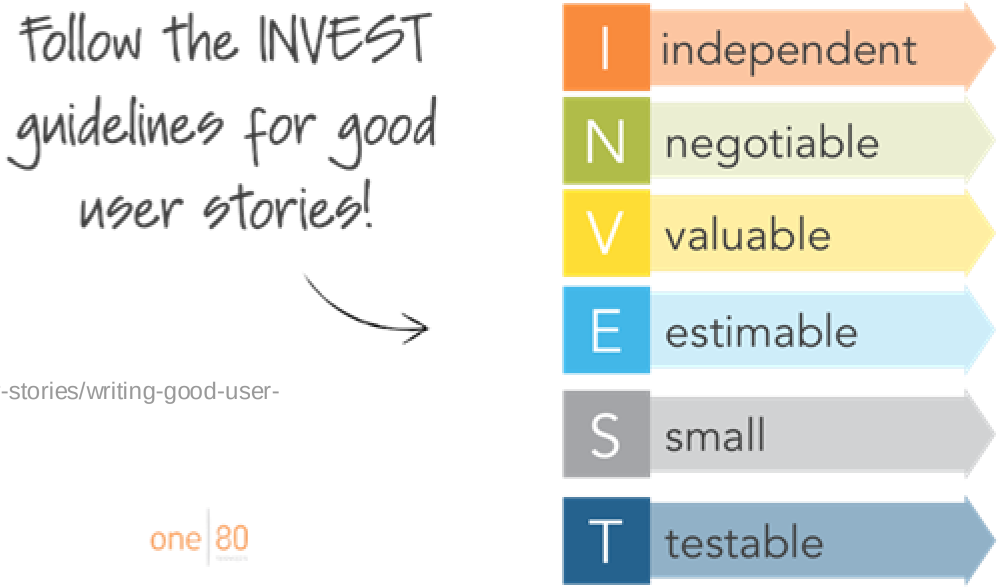
Conversation

- 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 user story?



Follow the INVEST guidelines for good user stories!

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

Source: http://one80services.com/user-stories/writing-good-user-stories-hint-its-not-about-writing/

Example

The university is looking to enhance student and staff engagement by creating an online platform where all university-related events are easily accessible. The goal is to provide a user-friendly website that serves as a central hub for information on various activities, ranging from academic seminars to sports events and club meetings.

Independent

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

- Schedule in any order.
- Not always possible

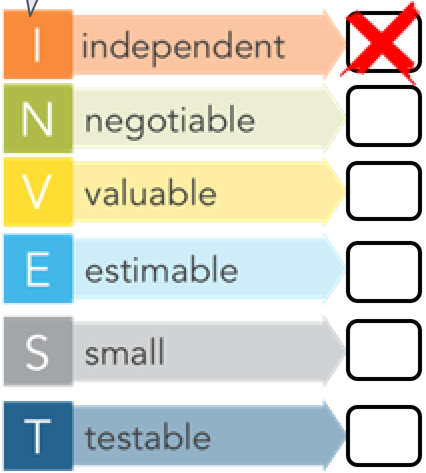
Counterexample

Assume that the homepage with an event calendar is already in place.

As a student, **I want to** receive notifications for events that are about to start, for those I have shown interest in, **so I** don't miss them.

Acceptance Criteria:

- An option is provided to 'Set a Reminder' for each event.
- Notifications are sent to users who have opted for reminders, shortly before the event starts.



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

Negotiable

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

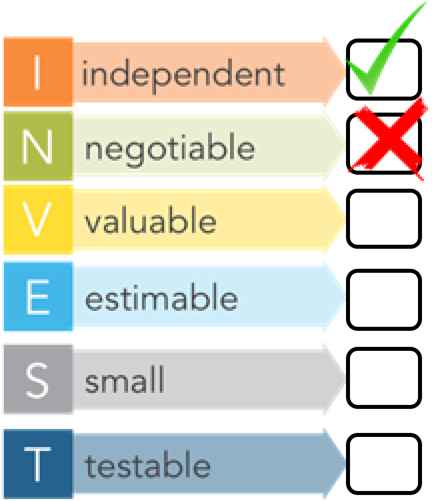
- Details to be negotiated during development
- Good Story captures the essence, not the details

Counterexample

As a student, **I want to** view the upcoming events at the university, **so I** can decide which ones to attend.

Acceptance Criteria:

- Add an interactive grid layout of upcoming events at the top of the homepage.
- Each event card in the grid is visible for a 2 seconds before automatically rotating to display the next set of events.
- Each card in the grid includes the event's name, type (e.g., seminar, sports game), duration, a brief description, and scheduled times.
- This grid of events is displayed under a prominent H1 heading that reads “Discover What’s Happening on Campus!”



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

Valuable

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

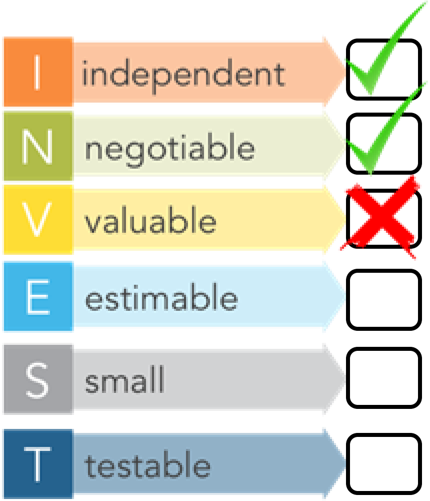
- This story needs to have value to someone (hopefully the customer)

Counterexample

As the Events Coordinator, **I want** a database to store details of students and staff interested in university events.

Acceptance Criteria:

- A database is constructed to manage user information.
- The database stores details such as name, email, phone number, favorite event types, date of birth, and history of event attendance or registrations.



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

Estimable

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

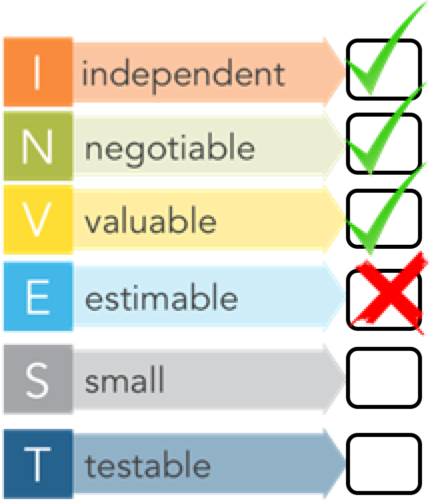
- Helps keep the size small
- It should provide enough details to estimate the amount of effort needed
- More on estimates later…

Counterexample

As an undergraduate student, **I want to** be able to filter university events, **so I** can choose the ones that align with my interests.

Acceptance Criteria:

- Filters are added to the event listings on the website.



I independent ✓

N negotiable ✓

V valuable ✓

E estimable ✗

S small □

T testable □

Small

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

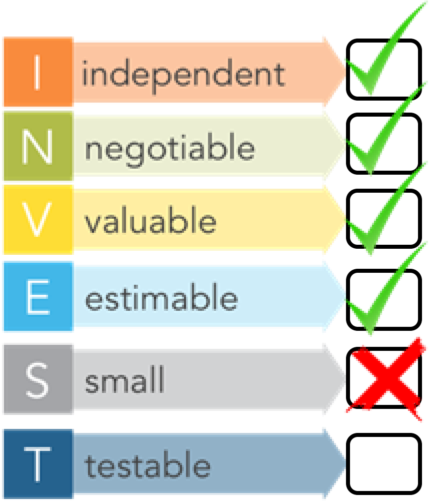
- Fit on 3x5 card
- At most two person-weeks of work (one sprint)
- Too big == unable to estimate

Counterexample

As a student, **I want to** easily find information about upcoming events, **so I** can participate in activities that interest me.

Acceptance criteria:

- A homepage is created displaying the university's name, motto, location, email, and contact information.
- The homepage features a calendar of upcoming university events.
- The event calendar includes details such as the event title, type (e.g., seminar, sports game, club meeting), a brief description, location, date, and time.
- Users can filter the event list by event type, date, and hosting department or club.
- The admin can update the event calendar as new events are planned or existing events are modified.



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

Testable

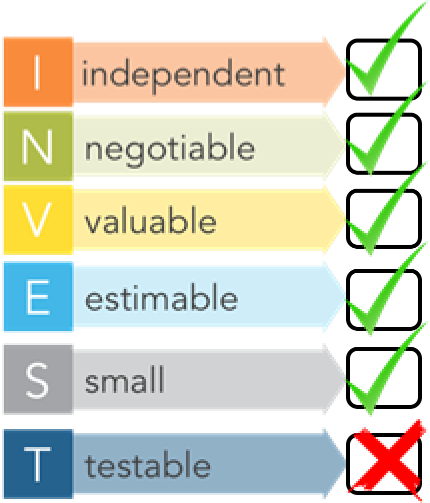
- Ensures understanding of task
- We know when we can mark task “Done”
- Unable to test == do not understand

Counterexample

As a student, **I want to** easily view promotional videos or trailers of university events, **so I** can decide which events to attend.

Acceptance Criteria:

- Promotional videos can be embedded on each event detail page.
- Videos are of high quality.
- The embedded video is well-integrated into the page design.
- The video size is large enough to ensure clarity.
- The video controls are user-friendly.



I independent ✓

N negotiable ✓

V valuable ✓

E estimable ✓

S small ✓

T testable ✗

Activity: Evaluate using INVEST

Follow the INVEST guidelines for good user stories!

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

User Story #1

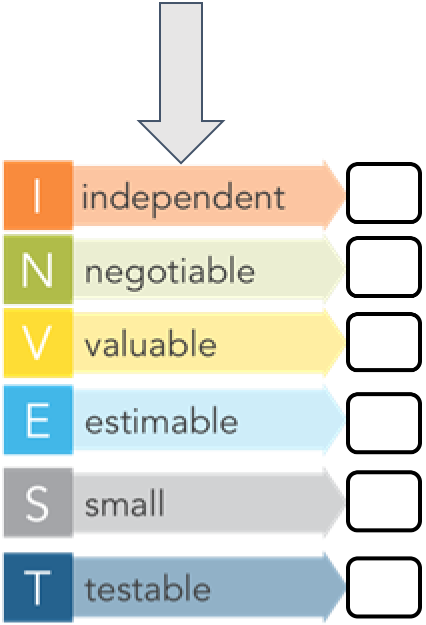
Select the most serious flaw

As the Events Coordinator, **I want** the website to seamlessly integrate with various academic calendars and departmental schedules, **so that** event information is always synchronized and accurate.

Acceptance Criteria:

- The website integrates with different academic and departmental calendars.
- Event information on the website reflects real-time updates from these calendars.

How can you fix it?



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

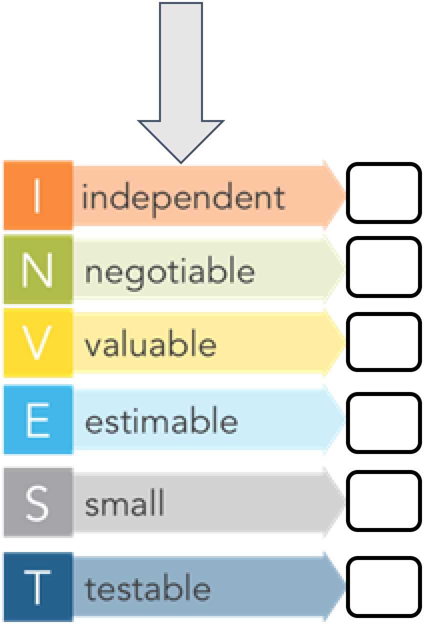
User Story #2

Select the most serious flaw

As a student, **I want** the website to have an intuitive navigation system **so that** I can find events effortlessly.

Acceptance Criteria:

- The website's navigation is intuitive to users.
- Users can find events with minimal effort.
- The navigation system feels natural and easy to understand.



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

How can you fix it?

“Plans are nothing,
planning is everything”

-Dwight D. Eisenhower

Time estimation

I'M JUST OUTSIDE TOWN, SO I SHOULD BE THERE IN FIFTEEN MINUTES.
ACTUALLY, IT'S LOOKING MORE LIKE SIX DAYS.
NO, WAIT, THIRTY SECONDS.

THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

https://xkcd.com/612

Activity: Estimate Time

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

Developer Team: just you

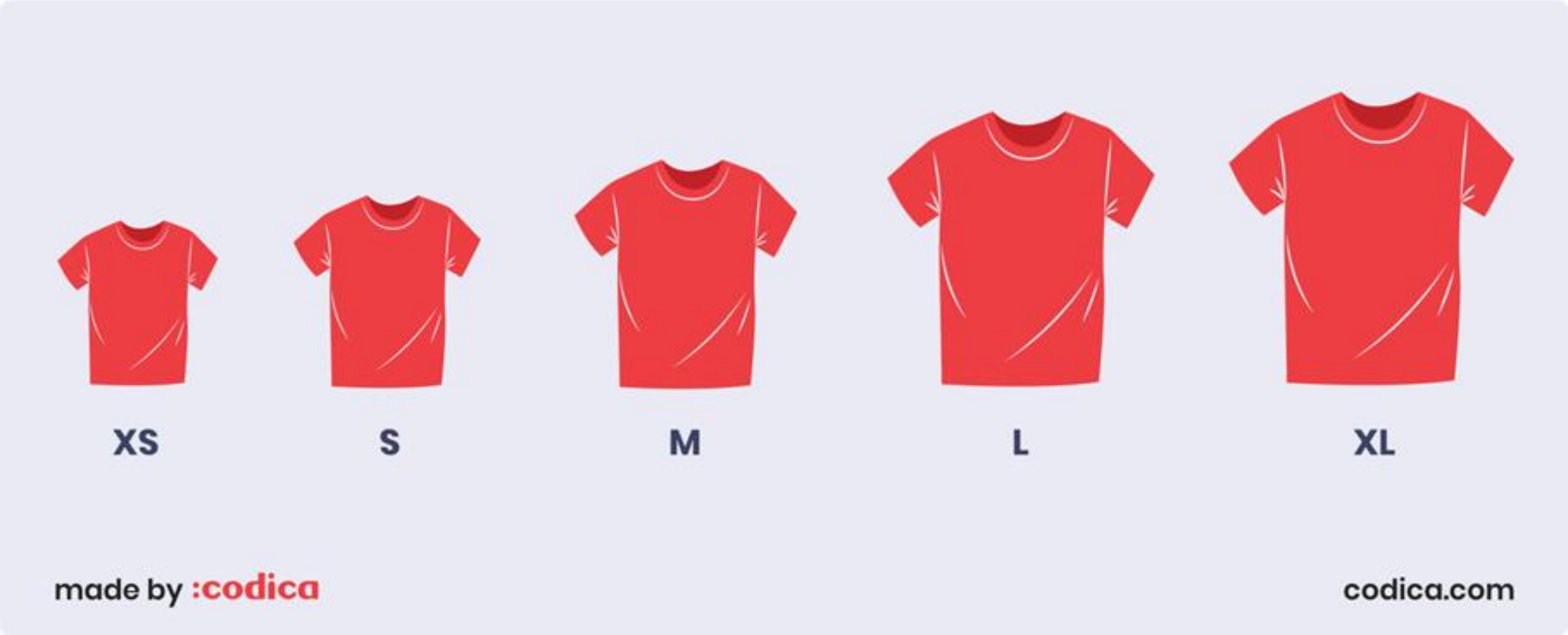
Task B: Bank smartphone app

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

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

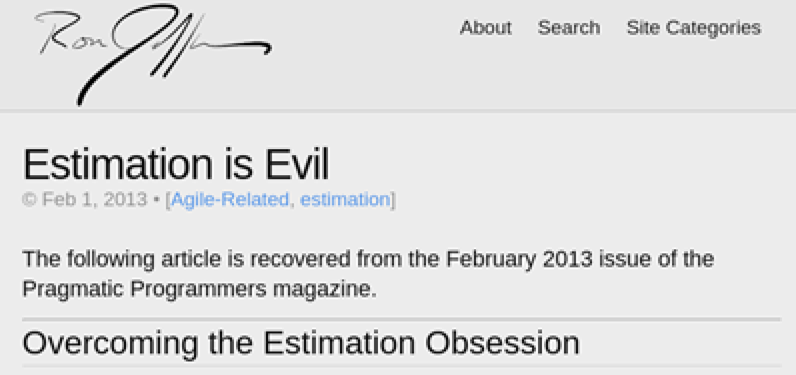
Improving Time Estimates

- Prevent conformity bias
- Do you have a comparable experience to base an estimate on?
- How much design do you need for each task?
- Break down the task into smaller tasks and estimate them.


XS
S
M
L
XL
made by :codica
codica.com

$$\times \pi$$

Is Estimation Evil?



Ron

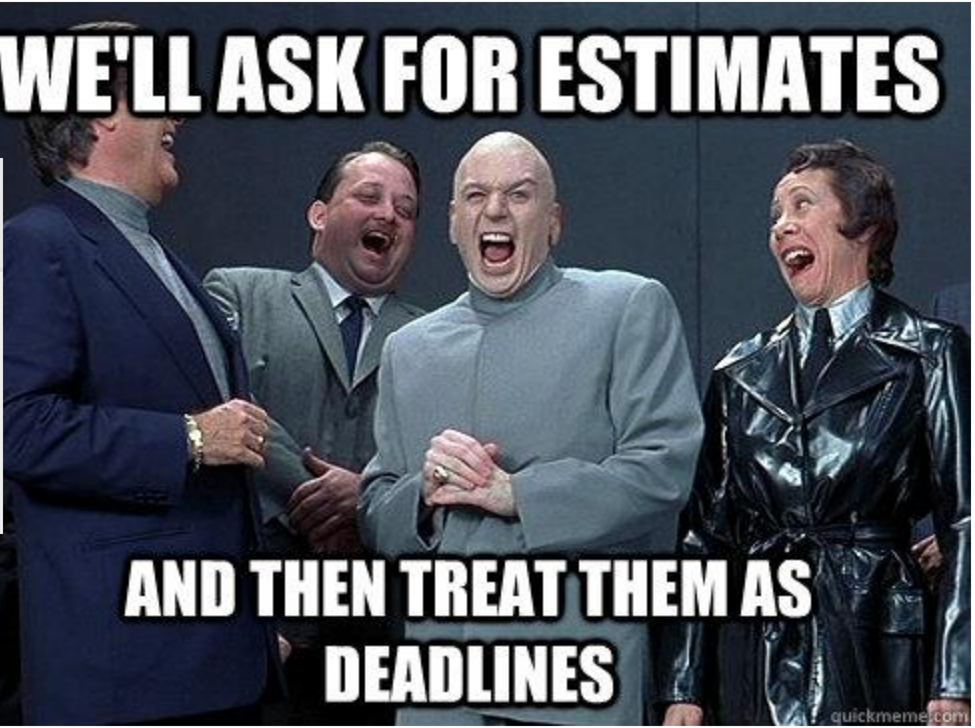
About Search Site Categories

Estimation is Evil

© Feb 1, 2013 • [Agile-Related, estimation]

The following article is recovered from the February 2013 issue of the Pragmatic Programmers magazine.

Overcoming the Estimation Obsession



Ron Jeffries's essay Estimation is Evil

Milestones and deliverables make progress *observable*

Milestone: clear end point of a (sub)tasks

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

Deliverable: Result for customer

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

What you need to know

- Recognize the importance of having a software process
- Main ideas of Agile/Scrum
- Understand backlogs and user stories
- Understand the difficulty of estimating tasks and progress
- We use milestones for planning and progress measurement