Carnegie Mellon

17-356/17-766 SOFTWARE ENGINEERING FOR STARTUPS

institute for SOFTWARE RESEARCH | Carnegie Mellon University School of Computer Science

Michael Hilton & Heather Miller

Administrativa

HW2 released.

Aside from project planning, you will need to have repos, etc, set up and ready for you to iterate on in subsequent sprints.

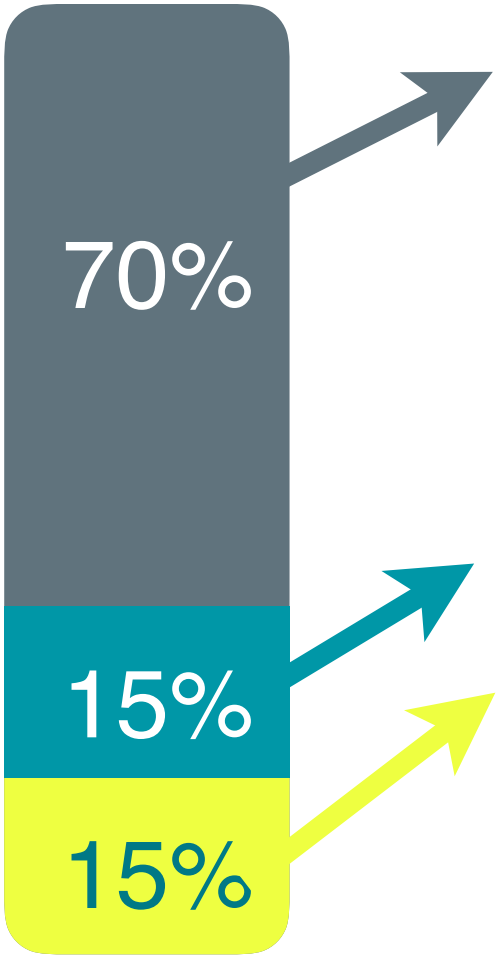
That means deployment too. If you've never done this before, get started ***today!***

Administrativa

Teamwork groups assigned.

Group	Last name	First name	Andrew ID
1	Chen	Joyce	jtc1
1	Gao	Andy	agao
1	Fung	Jonathan	jfung1
2	Chen	Tricia	triciac
2	Bittleston	Eli	dbittles
2	Yuan	Albert	ayuan1
3	Zhang	Judy	judyz
3	Barrett	Ryan	rnbarret
3	Yan	Derek	dyyan
4	Fishbaum	Benji	bfishbau
4	Chang	Michael	machang
4	Agarwal	Saransh	saransha
5	Shum	Rui Yuan	ruiyuans
5	Tao	Sean	shtao
5	Chen	Bobbie	bobbiec
5	Kotturi	Yasmine	ykotturi

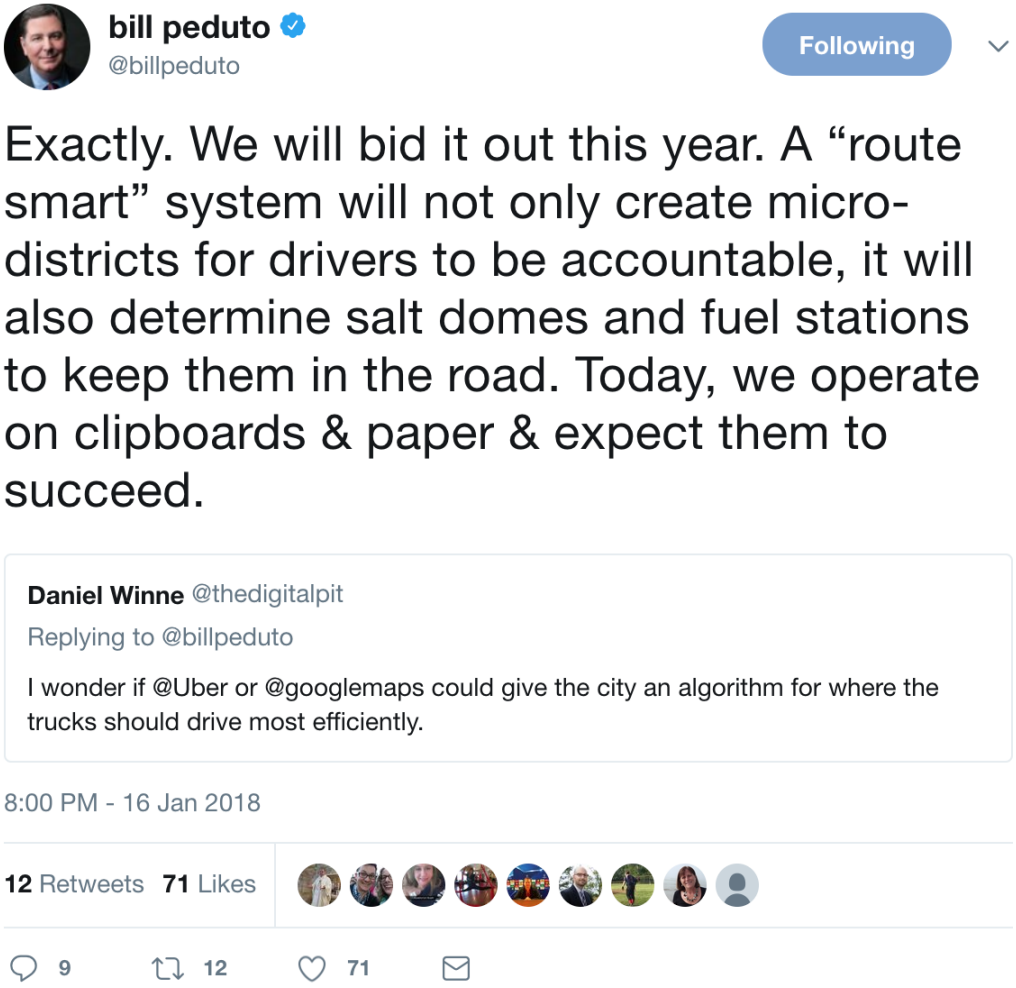
Today



- Coming up with a high-level idea for what we want our product to be, and feature prioritization
- Sprints (what is it?)
- Prepping for the HW, or, **Sprint 0**! (huh?)

Continued Prioritization and Estimation

Activity



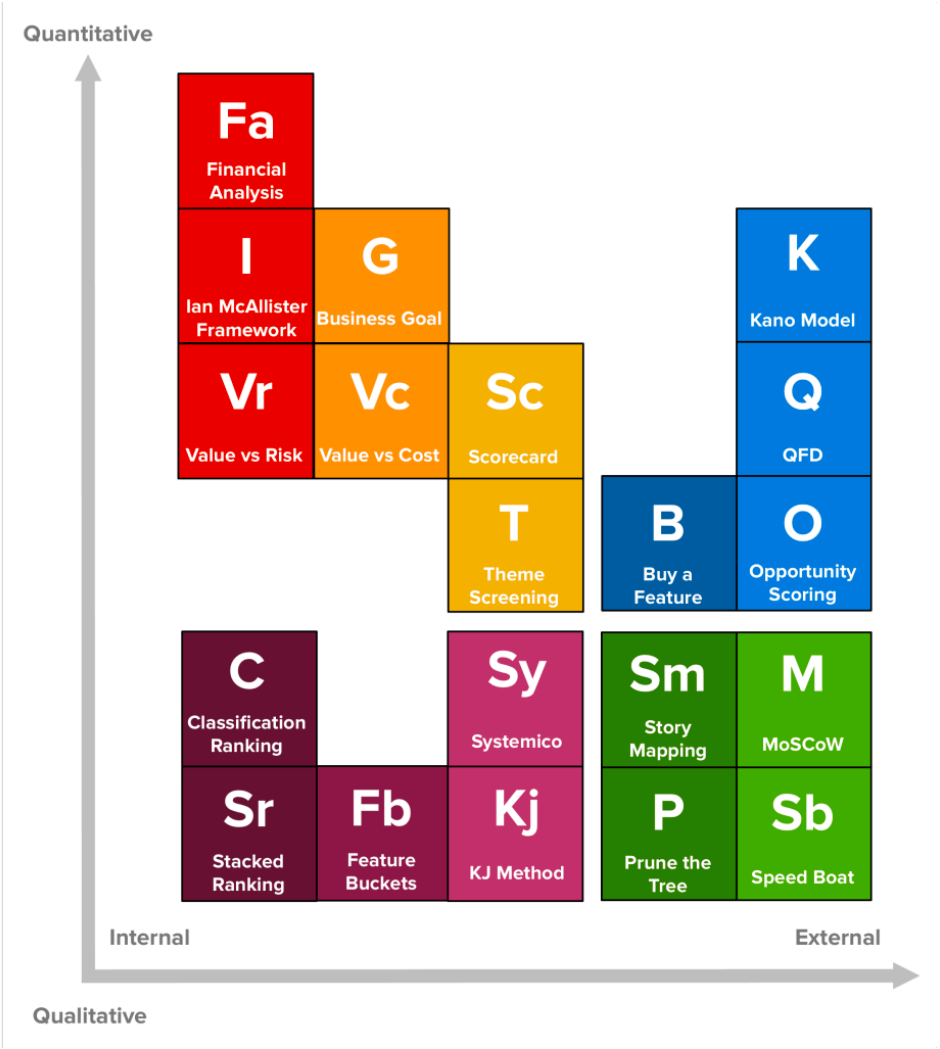
Activity:

Prioritize features to be developed

Estimating

Estimate one user story, feature, and tasks

How should we prioritize features?



There are *a lot* of prioritization techniques out there!

Literally called the "periodic table of product prioritization techniques"

How should we prioritize features?

How do we decide, ***together***, what are the best engineering tasks to focus on?

How should we prioritize features?

HiPPO - Highest Paid Person's opinion?

Famous for some well-known mishaps.

E.g., the decline of J.C.Penny's.

> When Ron Johnson, former head of retail at Apple who was responsible for the highly profitable Apple Stores, took over as CEO at J.C. Penney, he suffered from the HiPPO Effect. Without reviewing the existing data or investing in new data about the very different retail store he was now leading, he went full throttle ahead on his strategy for the department store chain. When his strategy was launched and he checked in to see if it was working, few had the courage to give him the unvarnished truth and be labeled as a resistor. Needless to say, his strategy wasn't succeeding with J.C. Penney's customers.

How should we prioritize features?

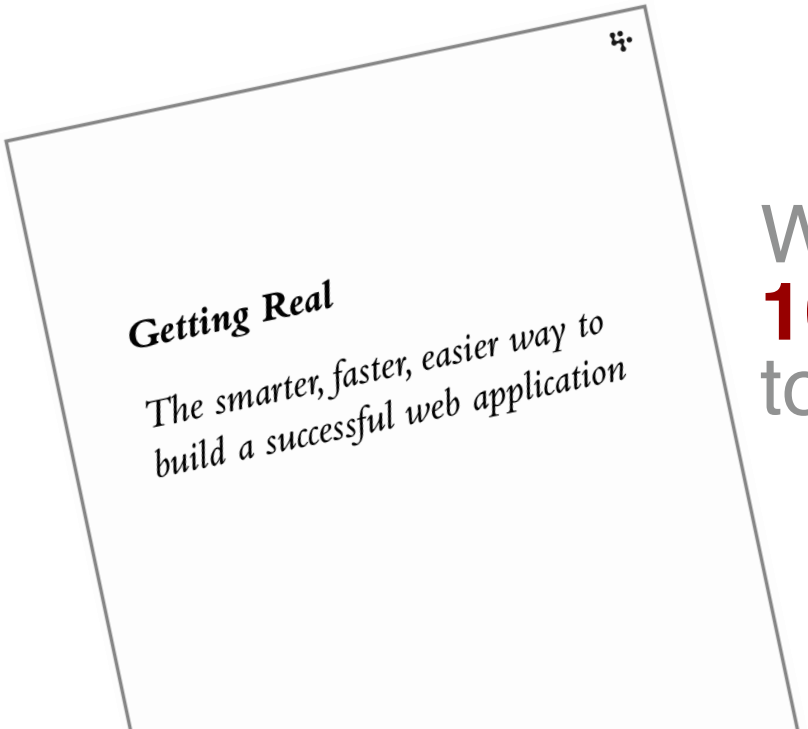
HiPPO - Highest Paid Person's opinion?

MoSCoW -

- Must have
- Should have
- Could have
- Won't have?

Design Considerations

Many of these taken from
“Getting Real” by 37signals.com



We’ll talk about **10** of these today.

Build Less

#1!

Focus, focus, focus

Less today means:

- Less Time
- Less Maintenance
- Less Support
- Less Testing

Lower your cost of change #2!

If you can easily change, you don't have to get it right the first time.

Small modifications are easier than 100% correct guesses on the first attempt.

Even if you are correct, the world around you might change

Embrace Constraints #3!

Constraints breed creativity

What is the Big Idea #4!

You should have a single sentence description of what you are building

What is your product's reason for existing?

What makes your product different than others?

Example:

Basecamp: Project management is communication
Backpack: Bring life's loose ends together
Campfire: Group chat over IM sucks
Ta-da List: Competing with a post-it note
Writeboard: Word is overkill

Ignore details early on #5!

“Perfect is the enemy of done”

Deciding details upfront leads to sunk cost fallacy

Use representations that abstract away details early in the design process, more details later in process

Ignore details early on #5!

The Devil's in the Details

I really got over the "get into details right away" attitude after I took some drawing classes...If you begin to draw the details right away you can be sure that the drawing is going to suck. In fact, you are completely missing the point.

You should begin by getting your proportions right for the whole scene. Then you sketch the largest objects in your scene, up to the smallest one. The sketch must be very loose up to this point.

Then you can proceed with shading which consists of bringing volume to life. You begin with only three tones (light, medium, dark). This gives you a tonal sketch. Then for each portion of your drawing you reevaluate three tonal shades and apply them. Do it until the volumes are there (requires multiple iteration)...

Work from large to small. Always.

–Patrick Lafleur, Creation Objet Inc. (from Signal vs. Noise)

Don’t worry about tomorrow #6!

It’s not a problem till it is a problem

Don’t waste time on problems you don’t have

Worry about getting today right, worry about tomorrow tomorrow

Don’t worry about scale before you have users

Half, not Half-baked #7!

Only build what is essential

Start with half the features you think you need

What you do implement, implement well

Learn to say “NO” #8!

Say NO to new features by default

“Innovation is not about saying yes to everything. It’s about saying NO to all but the most crucial features.” -Steve Jobs

Learn to say "NO" #8!

"We Don't Want a Thousand Features"

Steve Jobs gave a small private presentation about the iTunes Music Store to some independent record label people. My favorite line of the day was when people kept raising their hand saying, "Does it do [x]?", "Do you plan to add [y]?". Finally Jobs said, "Wait wait – put your hands down. Listen: I know you have a thousand ideas for all the cool features iTunes could have. So do we. But we don't want a thousand features. That would be ugly. Innovation is not about saying yes to everything. It's about saying NO to all but the most crucial features."

-Derek Sivers, president and programmer, CD Baby and HostBaby (from Say NO by default)

Think about hidden costs#9!

With each new feature you need to:

- -Gather Requirements
- -Manage in backlog
- -Design UI
- -Design backend
- -Code it
- -Test it
- -Document it
- -Deploy it
- - ...

Think about hidden costs#9!

The addition of a new feature might lead to a feature loop!
i.e., features that lead to more features.

"We've had requests to add a meetings tab to Basecamp. Seems simple enough until you examine it closely. Think of all the different items a meetings tab might require: location, time, room, people, email invites, calendar integration, support documentation, etc. That's not to mention that we'd have to change promotional screenshots, tour pages, faq/help pages, the terms of service, and more. Before you know it, a simple idea can snowball into a major headache." – Getting Real

Think about hidden costs #9!

For every new feature you need to...

1. Say no.
2. Force the feature to prove its value.
3. If "no" again, end here. If "yes," continue...
4. Sketch the screen(s)/UI.
5. Design the screen(s)/UI.
6. Code it.
7-15. Test, tweak, test, tweak, test, tweak, test, tweak...
16. Check to see if help text needs to be modified.
17. Update the product tour (if necessary).
18. Update the marketing copy (if necessary).
19. Update the terms of service (if necessary).
20. Check to see if any promises were broken.
21. Check to see if pricing structure is affected.
22. Launch.
23. Hold breath.

Design your interface first #10!

This is how users will interface with your application

All features in the back-end should support the interface somehow

As soon as you have your interface you can start “hallway usability testing”

Epicenter Design #10!

Start by designing the core

Design the most important part first (page, UI element, etc)

If something is “below the line” of features that make the cut, it should be less important than everything above it

Design considerations, listed:

1. Build Less
2. Lower your cost of change
3. Embrace Constraints
4. What's the Big Idea?
5. Ignore details early on
6. Don't worry about tomorrow
7. Half, not Half-Baked
8. Learn to say "NO"
9. Think about hidden costs
10. Design your interface first

Activity

What is the minimal feature set for snow removal application?

What features should we say NO to?

How can we design it using “epicenter design”?

Activity: last week's user stories

As a <>, I want <>, so that <>

	driver	know nearest salt dome	minimize downtime
	manager	how much each plow has plowed	pay correctly
	resident	want street plowed	to get places
	salt dome operator	know when trucks are coming	load salt more efficiently
	driver	optimal route	less downtime, maximize effort/efficiency
	commuter	entire route clear	be safe
	resident	know which roads are clear	make plans
EPIC	city manager	prioritize	better outcomes
	"	"	moving most cars
	"	"	deprioritizes, saves effort
	"	"	EMS access/movement
	"	"	plow access
	"	"	contingency plan
	"	"	fairness/equitability
	"	"	public transit
	"	"	physical characteristics of roads

Sprints

a time-box of one month or less during which a "Done", useable, and potentially releasable product increment is created.

Sprints

a time-box of one month or less during which a "Done", useable, and potentially releasable product ***increment*** *is created.*

what's this?

Sprints

a time-box of one month or less during which a "Done", useable, and potentially releasable product ***increment*** *is created.*

An Increment is the sum of all the Product Backlog items completed during a Sprint and the value of the increments of all previous Sprints. At the end of a Sprint, the new Increment must be "Done," which means it must be in useable condition and meet the Scrum Team's definition of "Done."

Sprints

Ok. Where do we start?

with Sprint 0 of course!

Sprint 0!

Sprint Zero enables software development teams to ramp up at the beginning of an engagement.

It happens after the objectives and scope of the project is established in a Discovery or Project Initiation/Inception activity.

Sprint 0!

1. Create the basic architecture and infrastructure for the project and start building your DevOps infrastructure
2. Define the basic UX concepts to be used
3. Set up development, QA, communication and collaboration tools and environments
4. Identify and groom the initial product backlog
5. Hold a sprint planning session for the first few sprints
6. Incorporate a kickoff ceremony with the team to communicate and clarify roles and responsibilities, and to share the product vision and project expectations

HW2:

You will be completing:

- objectives and scope of the project
- sprint 0!

Things you will quickly get up to speed with for this homework:

Node, MVC, Git/GitHub, CI, Docker, Azure