

Project Inception

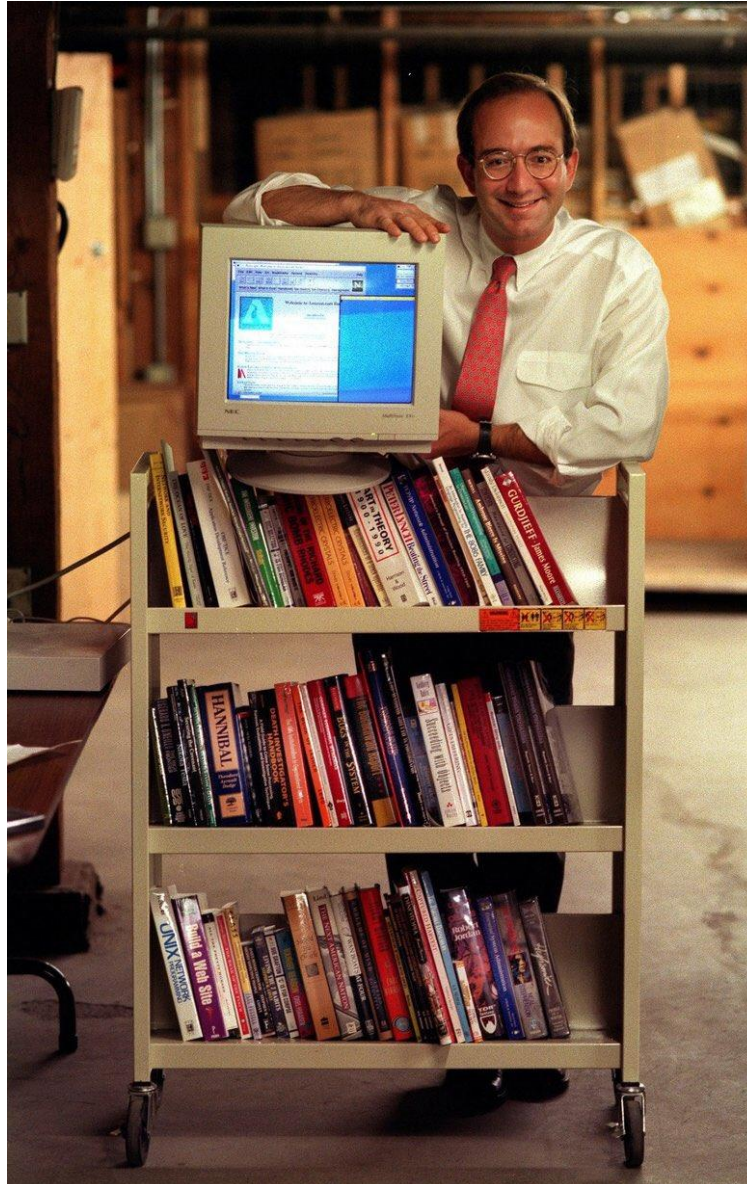
CSE 3311 / 5324
Christoph Csallner
University of Texas at Arlington (UTA)

We Are Doing Agile: Who Writes Docs?

Amazon
started
in a
garage
in '94

Picture for
first story
published
about
Amazon
(Sept 1995)

They
do

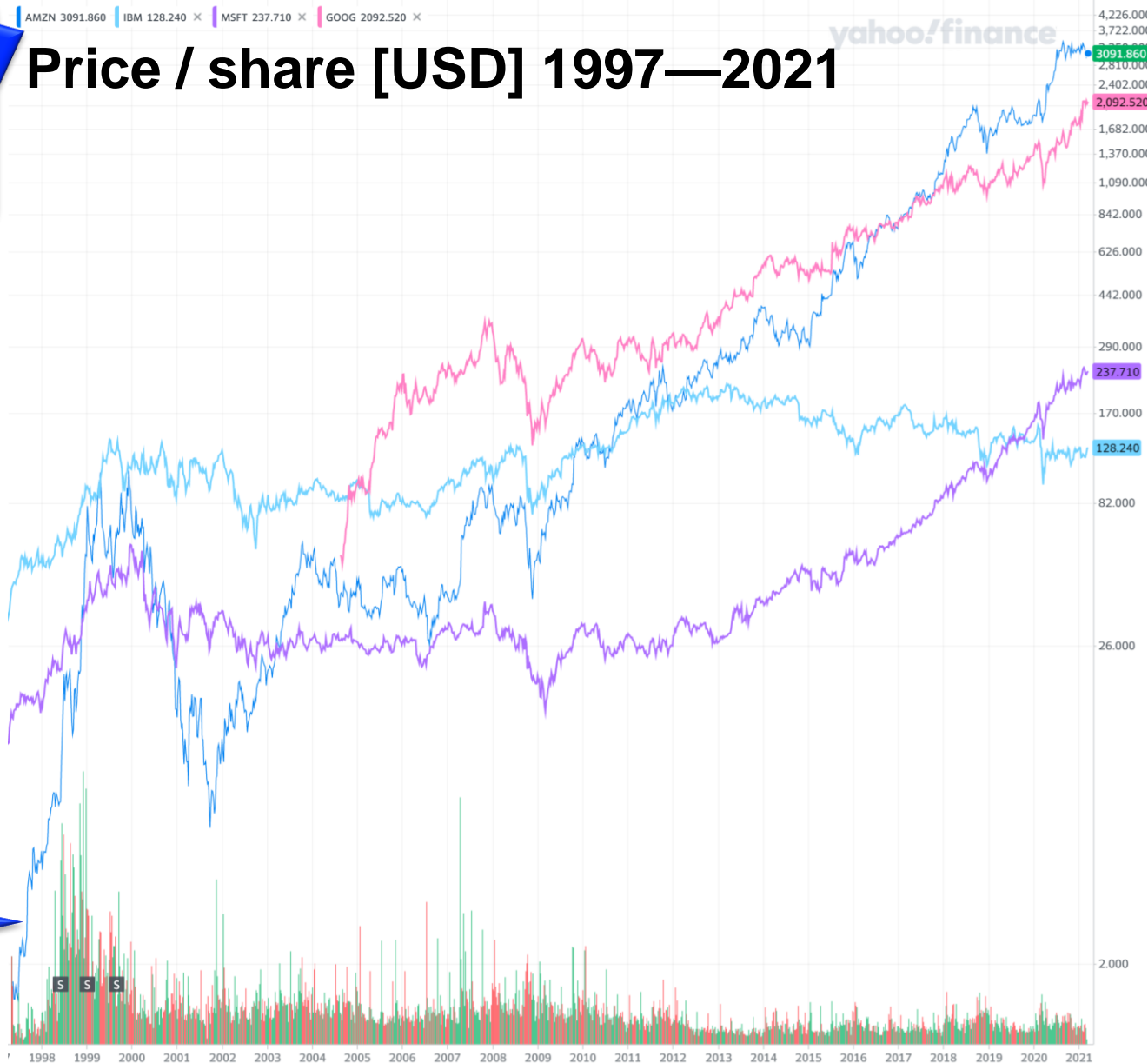


We Are Doing Agile: Who Writes Docs?

Amazon
started
in a
garage
in '94

Price / share [USD] 1997—2021

They
do



They
do

Jeff Bezos Interview (Nov 15 2012)

- Bezos's quotes are copy-paste from 2012 interview
 - I removed interviewer's comments, formatted, and **bolded** some parts
 - <https://charlierose.com/videos/17252>

>> (start quote, from 5:45)

Well, the traditional kind of corporate meeting is somebody gets up in front of the room and presents, let's say a PowerPoint presentation or some kind of slide show. And in our view, that is a very kind of, **you get very little information that way.**

You get bullet points -- **it's kind of easy for the presenter but difficult for the audience.**

Bezos Interview 2/n

And so instead what we do is all of our meetings are structured around a six-page narrative memo. And when you have -- **when you have to write your ideas out in complete sentences and complete paragraphs it forces a deeper clarity of things-**

Yes and so when you -- and so what we do is we just sit and you know, somebody will say why don't you read the memos in advance. Well part of the problem there is that time to read them in advance doesn't materialize out of nowhere.

And so this way, you know **everybody has the time because we're all sitting around the table reading simultaneously.**

You know everybody has actually read the memo. The author who has put a tremendous amount of work into writing the memo gets the nice warm feeling of seeing everybody read it so they know actually it hasn't been a waste of time -- they are getting -- it is getting read.

Bezos Interview 3/n

It is actually being read. And there is another nice thing about this approach too. If you have a -- a kind of a traditional PowerPoint presentation, sometimes -- executives are -- are very good at interrupting. And so the person will get halfway into their presentation and then some -- some -- some executive will interrupt the conversation. And that question that the executive asks probably was going to be answered five slides in.

And so if you read the whole six-page memo, it often happens to me, I get to page two and I have a question. I jot it in the margin and by the time I get to page four the question has been answered so I can just cross it off and it saves a lot of time.

<< (end quote)

Amazon's 2017 Letter to Shareholders

- Quote is copy-paste from Amazon's 2017 letter
 - I just formatted and **bolded** some parts
 - <https://www.aboutamazon.com/news/company-news/2017-letter-to-shareholders>

>> (start quote)

We don't do PowerPoint (or any other slide-oriented) presentations at Amazon. Instead, **we write narratively structured six-page memos**. We silently read one at the beginning of each meeting in a kind of “study hall.” Not surprisingly, the quality of these memos varies widely. Some have the clarity of angels singing. They are brilliant and thoughtful and set up the meeting for high-quality discussion. Sometimes they come in at the other end of the spectrum.

Amazon Letter to Shareholders 2/n

In the handstand example, it's pretty straightforward to recognize high standards. It wouldn't be difficult to lay out in detail the requirements of a well-executed handstand, and then you're either doing it or you're not.

The writing example is very different. The difference between a great memo and an average one is much squishier. **It would be extremely hard to write down the detailed requirements that make up a great memo.**

Nevertheless, I find that much of the time, readers react to great memos very similarly. **They know it when they see it.** The standard is there, and it is real, even if it's not easily describable.

Amazon Letter to Shareholders 3/n

Here's what we've figured out. Often, when a memo isn't great, it's not the writer's inability to recognize the high standard, but instead a wrong expectation on scope: **they mistakenly believe a high-standards, six-page memo can be written in one or two days or even a few hours, when really it might take a week or more!** They're trying to perfect a handstand in just two weeks, and we're not coaching them right.

The great memos are written and re-written, shared with colleagues who are asked to improve the work, set aside for a couple of days, and then edited again with a fresh mind. They simply can't be done in a day or two. The key point here is that you can improve results through the simple act of teaching scope – that **a great memo probably should take a week or more.**

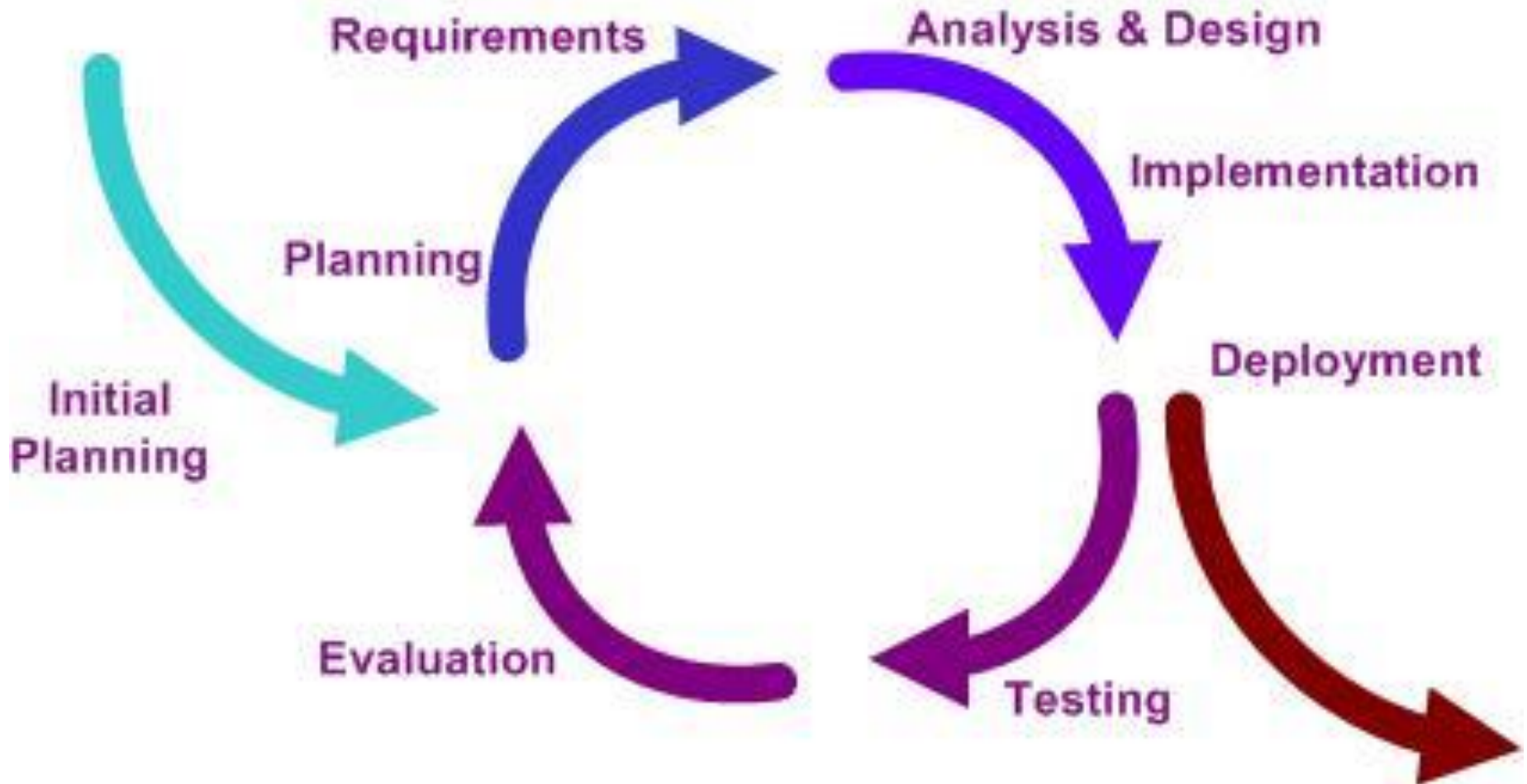
Amazon Letter to Shareholders 4/n

Beyond recognizing the standard and having realistic expectations on scope, how about skill? Surely to write a world class memo, you have to be an extremely skilled writer? Is it another required element? In my view, not so much, at least not for the individual in the context of teams. The football coach doesn't need to be able to throw, and a film director doesn't need to be able to act. But they both do need to recognize high standards for those things and teach realistic expectations on scope.

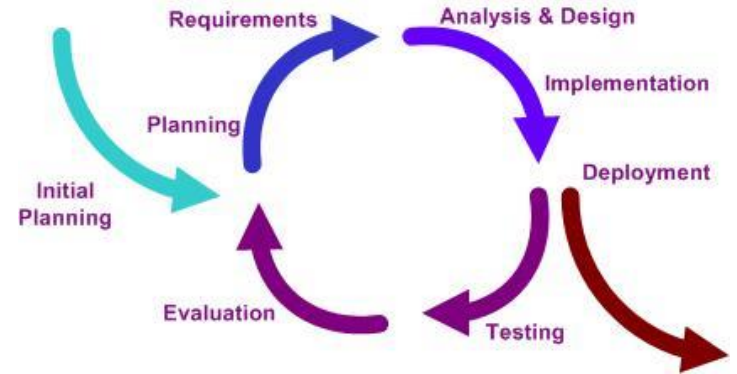
Even in the example of writing a six-page memo, that's teamwork. **Someone on the team needs to have the skill, but it doesn't have to be you.** (As a side note, by tradition at Amazon, **authors' names never appear on the memos – the memo is from the whole team.**)

<< (end quote)

Big Picture



Big Picture



http://en.wikipedia.org/wiki/File:Iterative_development_model_V2.jpg

- Before iteration 1
 - High-level overview of requirements
 - **Pick small subset X = those with highest risk exposure for project**
 - Analyze requirements of X
- Iteration 1: Design, code, test part of X
- More high-level planning in early iterations
 - Overview of all requirements, analyze requirements
 - Schedule requirements into early or late iterations
- More coding and testing in later iterations
 - Less need for high-level planning

Game Plan

- Establish common vision and scope of project
- Analyze some 10% of use cases
- Analyze critical non-functional requirements
- Create business case
- Prepare subsequent development

Background: Brainstorming

- Follow common rules of professional brainstorming
 - Group activity
 - » Limit group size to 10 or less, so everyone can participate
 - » **Round-robin, one idea per turn, so everyone can participate**
 - Anything goes
 - » (Except for distasteful ethnic, religious, sexist, etc. comments)
 - No criticism allowed
 - » Want to get all ideas out
 - » **“Silly” ideas may trigger important ideas later**
 - » Will weed out silly ones later
- Second session (after a break)
 - Prioritize identified ideas
 - Use a voting scheme

Start Documents

- Start them now, re-use & update throughout project
- Document = anything you can share and later re-use
 - **Fancy MS Word file likely not the best way**
 - Imo, the following are all easier to share and manage
- Google Docs: <http://docs.google.com/>
 - [+] Easy to use, share [-] Outside your repository
- Plain (text / html) file
 - [+] Easy to use, share [+] Within your repository
- Following slides briefly describe the documents
 - CL, pages 50, 58

Vision and Business Case

- High-level requirements
 - The system does X
- High-level goals and constraints
- Business case, executive summary
 - Fast, Cheap, Good: Choose any two. [CL, page 101]
- **List most closely related competitors**
 - Both open-source and closed-source (commercial, ..)
 - Apps, tools, existing web services, etc.
 - How do you improve on them?
- Use creative group work: Brainstorming, etc.
- Example Vision document: CL, page 109

Shared, elevating, exclusive, and achievable

PROJECT VISION

Vision should make objectives clear

- What are the project objectives?
 - To produce the software cheaply?
 - To make the software reliable, adaptable, portable?
- Will be important throughout project
 - Guide specification & design
 - Guide reviews

Team needs to buy in to shared vision

- High-performance team work requires a shared vision
 - Study of 75 projects showed:
Each effective team had a shared vision
- Helps streamline decisions on smaller issues
- Helps avoid time-wasting side-activities
- Builds trust among team members
 - Everyone working on same goal
 - Enables good cooperation among team members
- Do not have to revisit big questions over and over again throughout project

Vision should be elevating

- Effective vision motivates team
- Team needs a challenge / mission
 - Team does not form around mundane goal
- E.g.: Will a team form around the following vision?
 - “We’d like to create the third-best web site designer on the market and deliver it 25% later than the industry average.”

Vision must be achievable

- Some people can be motivated with impossible goals
 - Sales and marketing people?
- Software developers tend to see impossible goals as illogical
 - Impossible goals often de-motivate software developers
- Same applies if each sub-goal is achievable but the collection of goal is not achievable, e.g.:
 - Short schedule + Low cost + Complex features

Vision should be exclusive

- Vision statement should be written such that it is clear what is part of project **and what is not**
- Word the vision statement such that it excludes things that should not be part of the project
- **An exclusive vision helps team to produce the minimal amount of software**

Committing to the vision

- Vision statement should be written down
- Vision statement is the first item put under version control
 - Check into your code repository
- Should not be changed frequently
- But change vision (+ vision statement) when your understanding of project changes significantly



IN-CLASS EXERCISE: PROJECT VISION

Project Vision

- Get together with your team
 - Join Teams breakout room “Team N”
- Develop a vision statement for your class project that has the discussed properties:
 - Elevating
 - Achievable
 - **Exclusive: State what you won’t do as part of project**
 - Shared
 - (The vision here is just for this exercise and does not commit you to follow through with it in your project)
- Post your vision stmt to class meeting’s Teams chat

Use-Case Model

- Functional requirements
 - (not related with “functional programming”)
- **Now: Identify names of most use-cases**
 - **Name of a use-case = verb and a noun phrase**
- **Now: Analyze 10% of use-cases in detail**
- Later more on use-cases

Supplementary Specification

- Other requirements, non-functional
- Requirements that are not in the use-case model
- (Business / domain) (rules / knowledge)
 - E.g., tax laws, physical laws
 - Example: CL, page 116
- Example Supplementary Specification: CL, page 104

Start with Quick & Dirty

UI PROTOTYPE

Background: Prototype is Just a Model

- Should reflect only a few aspects of full product
- The full product will likely look very different
- Example from hardware:
 - Bill English built first computer mouse prototype (1964)

» [<https://www.sri.com/hoi/computer-mouse-and-interactive-computing>]



[https://en.wikipedia.org/wiki/The_Mother_of_All_Demos#/media/File:SRI_Computer_Mouse.jpg]

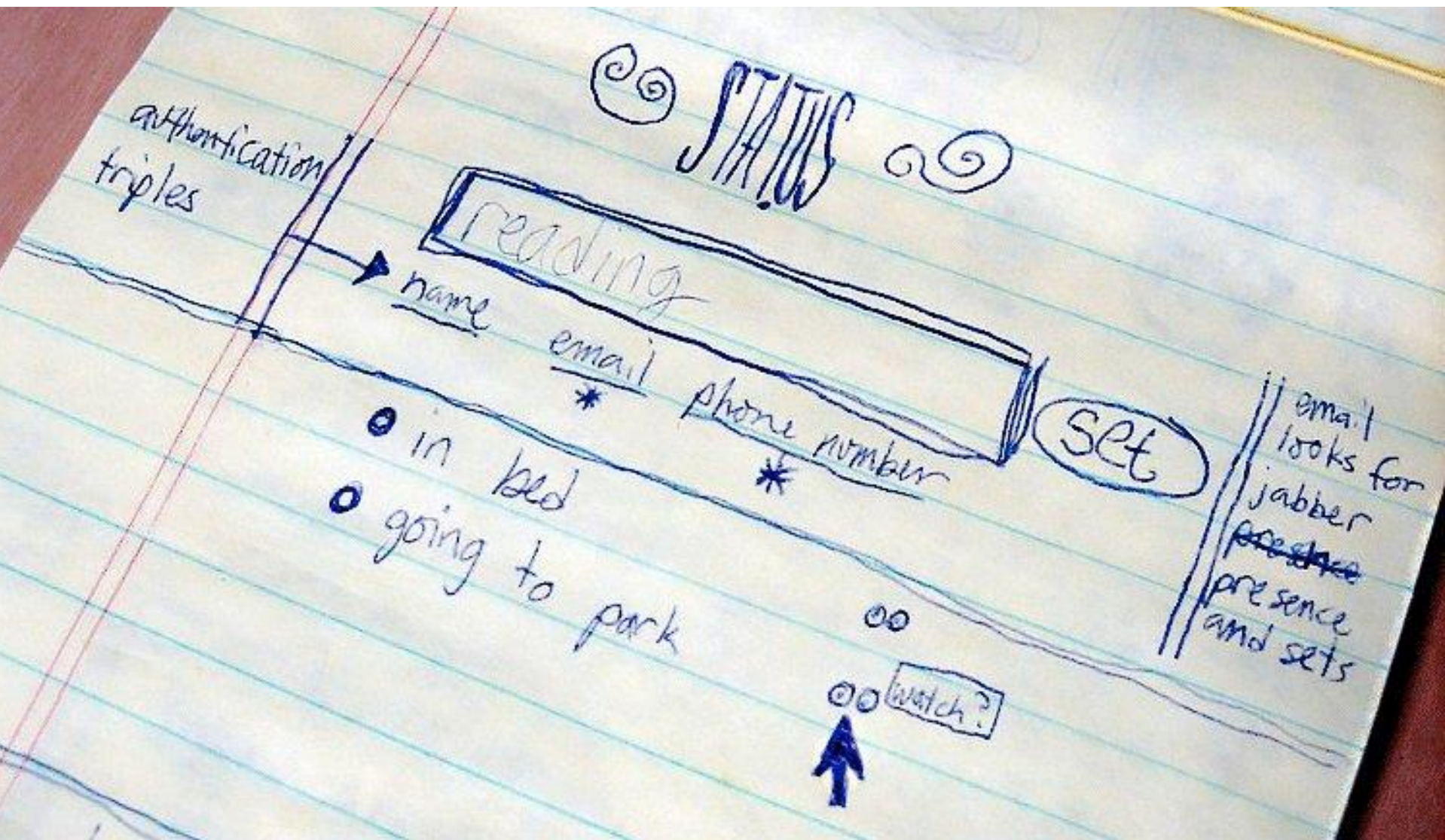
UI Prototype: Clarify the Vision

- **Mock screenshots**
- Can (and often should) be extremely quick & dirty
 - Pencil-on-paper
 - » Throw away and iterate or
 - » **Scan / take-phone-picture and include in document**
 - (PowerPoint)
- Interactive user interface demo
 - Third-party prototyping tools
 - » <https://proto.io/>
 - » <https://www.invisionapp.com/>
 - » <https://pencilcase.io/>
 - » <https://neonto.com/nativestudio>

Additional Benefits [CS]

- Less intimidating than a full program
 - Especially when you use a paper prototype
- Discourages low-level feedback on minor flaws
 - Paper prototype does not show high-res details
 - Polished UI invites low-level comments (“move by 1 pixel”)
- Encourages more creative feedback
 - Looks like design is “still on the drawing board”
 - Sketchy drawings makes people feel included in process
 - Feedback not seen as requiring expensive changes
- Allows you to show progress later
 - Put initial paper sketch next to current screenshot

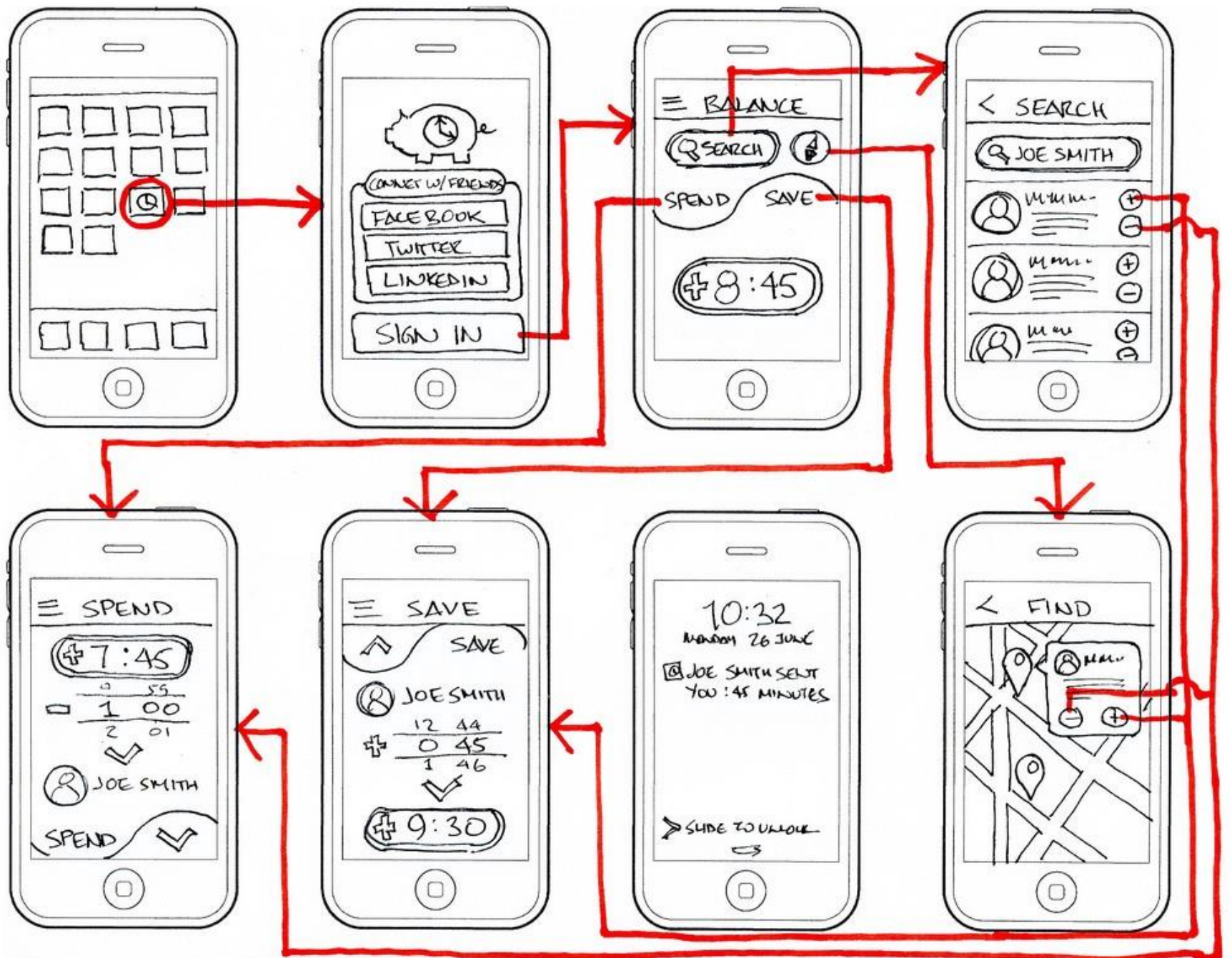
Example: Twitter's 2005 UI Prototype



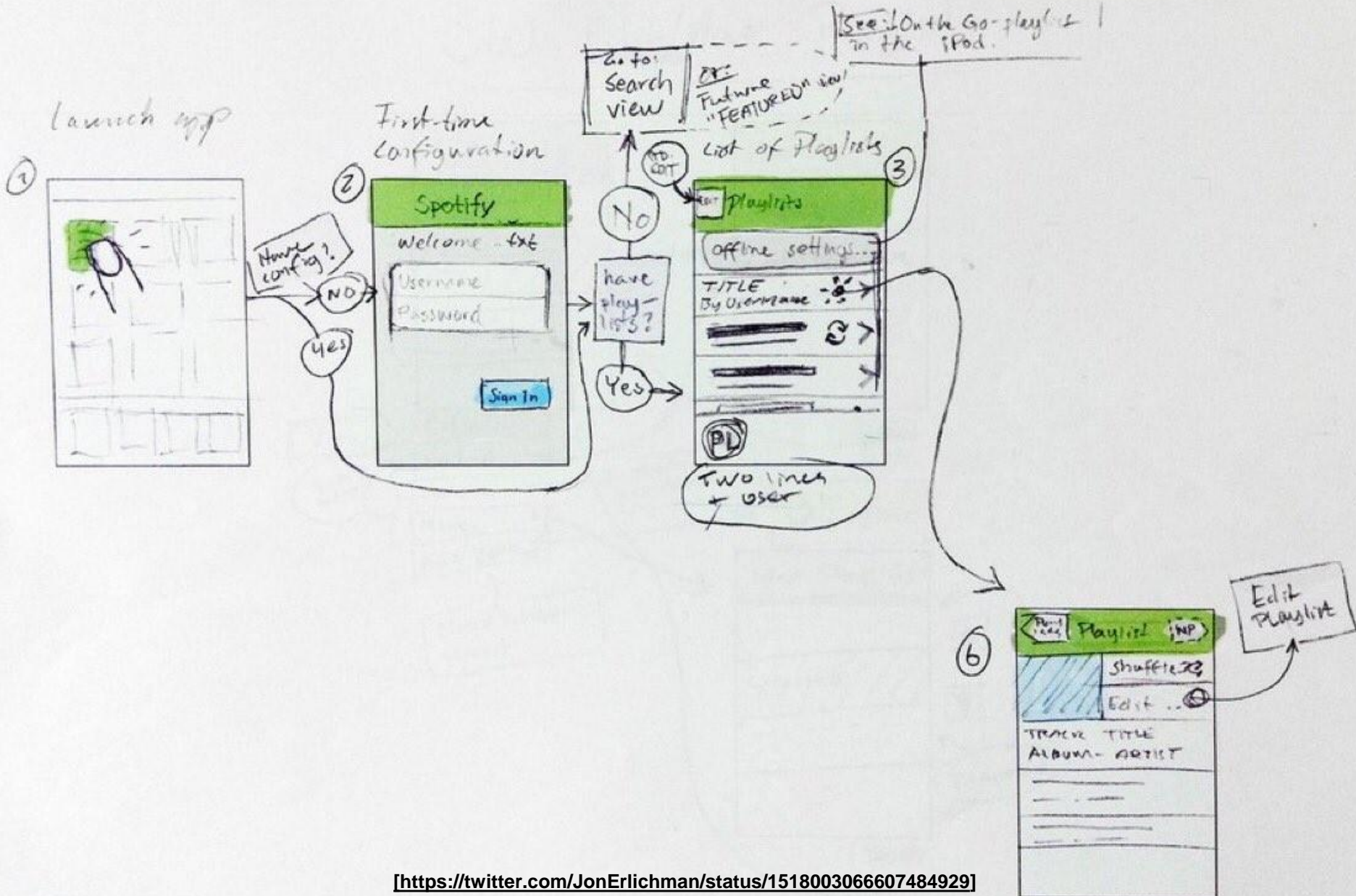
Flow Between Screens

- Show how user navigates between screens
- Use informal notation, paper/pencil often enough
- Various names
 - Wireframe
 - Storyboard
- Examples:
 - Tap setting button → settings menu
 - Tap product title → product detail page





Example: Spotify's 2006 UI Prototype





IN-CLASS EXERCISE: USER INTERFACE PAPER / PENCIL PROTOTYPE

UI Prototype: Pencil on paper

- Get together with your team
 - Join Teams breakout room “Team N”
- Pick one of your project ideas
- **Develop a UI prototype for at least one screen for which you do not have any prototype or code yet**
 - This is just for the purpose of this exercise – it is not a commitment to implement it as part of your project
 - Optional: Add additional screens & inter-screen transitions
- Post your prototype to class meeting’s Teams chat
 - Phone camera snapshot of pencil on paper
 - Screenshot of Whiteboard built into Teams

Technical Prototype

- Validate technical ideas
 - Proof of concept implementations
 - Address “show-stopper” technical questions
- Examples
 - Does 3rd-party library work on hardware we want to use?
 - Does 3rd-party web service provide features we need?

Software Development Plan

- Iteration plan
 - Describe what you plan to do in the first iteration
- Tools
- People
- Education
- Other resources

“[R]isk is neither bad nor good, but is always present, and can be effectively dealt with to increase the probability of project success”

[Marvin J. Carr, Suresh L. Konda, Ira Monarch, F. Carol Ulrich, Clay F. Walker. Taxonomy-Based Risk Identification, Technical Report CMU/SEI-93-TR-6, June 1993]

RISKS

Informal Survey (Spring 2021)

- Naive questions for software people from my perspective of teaching a software engineering class: If you are managing a software development team, how important is risk management for you?
 - Do you explicitly keep track of the top-N risks your team is currently facing?
 - How much time (if any) do you spend on estimating risk likelihood and potential impacts?
 - Maybe you do this automatically and intuitively all day long without having to write it down anywhere?

The Two Answers I Received

- “Technical Risks we usually try to mitigate once we recognize them (some experiments, POC, etc). **Risks for the business model or regarding time lines, requirements accuracy etc. have to be written down** to address them on management level.”
 - CEO for small software consulting company
- “As a ‘project manager’ **risk management is everything.** That's pretty much the impact likelihood analysis is top of mind. We have RAID (Risks, Assumptions, Issues, Dependencies) log. **They are written down** because there are stakeholders at play and **if you don't make them aware, and the risk becomes an issue, that will be our ‘mistakes’**” [..].
 - Project Manager for medium software consulting company

Risk List & Risk Management Plan

- **List of project risks**

- Specific to your project
- Business, e.g.: **“Not really sure what user really wants”**
- Technical, e.g.:
 - » “We want to use library X but we do not know if it can do Y”
- Resources, e.g.:
 - » “X does not know Android”
 - » “Y has a 30 hours/week job”
- Schedule, e.g.:
 - » “3rd iteration deadline is on same date as major deadline in class X that A,B are also taking”

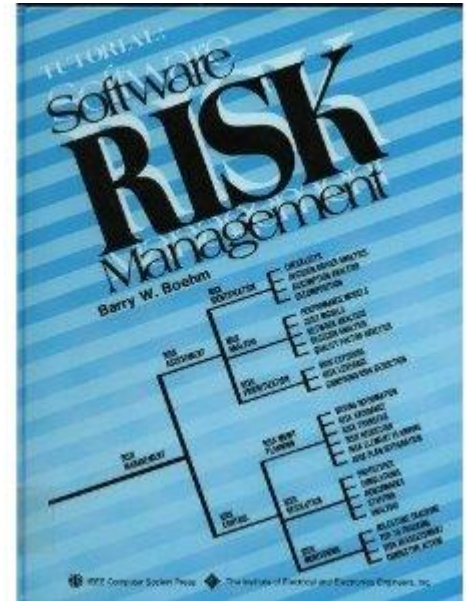
- **Plan of how to mitigate these risks**

Project risk is common

- Project members just not always call it “risk”
 - People often avoid the term “risk”, to avoid talking about possible bad events happening
- Instead project members may have:
 - **Concerns**
 - **Uneasy feeling**
 - **A doubt**
- ... and keep these to themselves
- **Goal: Make these things explicit and manage them**

Top risk factors

- Risks according to Boehm 1989 book: Software Risk Management
- Mitigation strategies by [RF]
- Subset that applies to this class
 - More on software project risks & managing these risks: CSE 4322

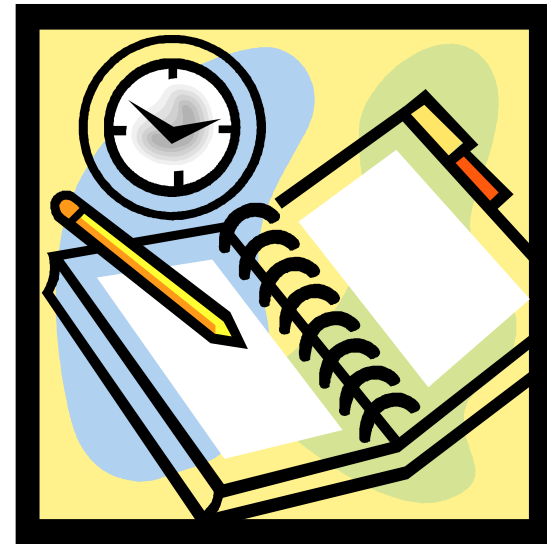


Personnel shortfall

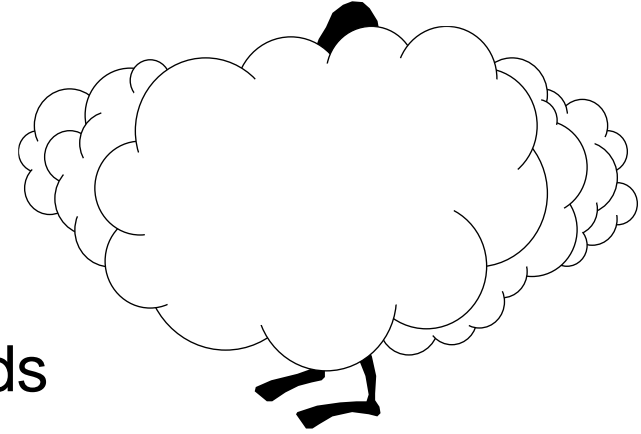
- Caused by
 - **Inexperience with domain, tool, development techniques**
 - Personnel turnover, loss of critical team members
- Mitigate via:
 - **Provide cross-training**
 - **Pre-schedule key personnel**

Unrealistic schedule / budget

- Mitigate via:
 - Multiple estimation techniques
 - » Estimate each feature independently
 - » What did similar projects produce (e.g., in previous semesters)?
 - » How many LOC will we write? How long do we need per LOC to write / debug / refactor / adapt / test it?
 - Design to schedule
 - **Incremental development**
 - Software reuse
 - **Requirements scrubbing**



Wrong functionality



- Caused by
 - Do not fully understand customer needs
 - Communication with client is too complex
 - Engineers have insufficient domain knowledge
- Mitigate via:
 - **Talk to actual customers and users. Frequently.**
 - » Show them prototypes, get feedback
 - Analyze customer organization & mission
 - Survey users

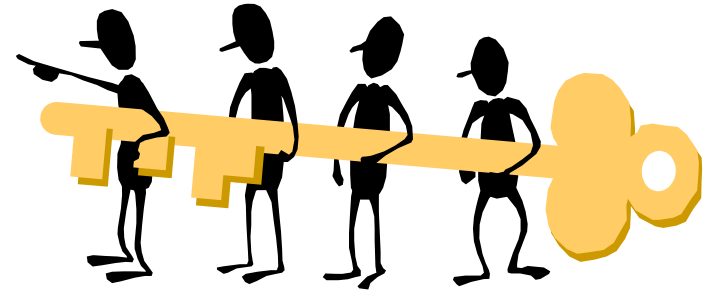
Wrong user interface

- Mitigate via:
 - **Prototyping**
 - » Pencil on paper
 - » Screen-flow diagrams
 - » Detailed screen mock-ups
 - » Interactive prototypes
 - Scenarios
 - User-task analysis
 - User characteristics
 - » User classes
 - » Work loads
 - » Work styles



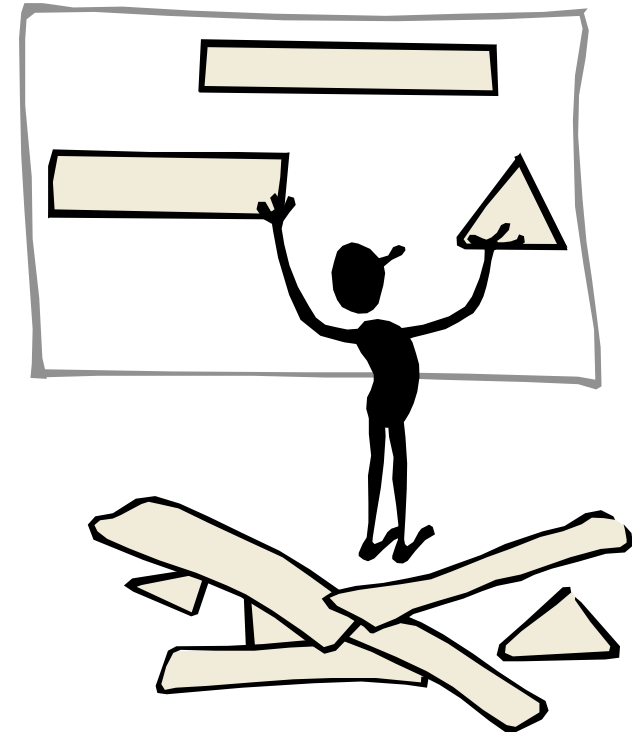
Gold plating

- Caused by:
 - Resume-driven development (RDD)
 - Team has nothing else to do
 - Misunderstanding of customer priorities
- Mitigate via:
 - Requirements scrubbing
 - Prototyping
 - **Cost-benefit analysis**
 - » “If we make the proposed change X to our project, will our customer be willing to pay (at least) our costs of performing X?”



Requirements volatility

- Mitigate via:
 - High threshold for changes
 - **Incremental development**
 - » Defer changes
 - **Information hiding:**
Ease changes, e.g., via:
 - » **All fields private**
 - » Most classes have their own **interface** type



Bad external components / libraries

- Caused by:
 - Used external library has unexpected bugs / problems
- Mitigate via:
 - Is library already used successfully elsewhere?
 - » What kind of projects are using the library and for what?
 - » Are there blog posts by big projects on why they are not (or no longer) using the library?



When to identify risks

- On an ongoing basis
 - During initial project planning
 - And **throughout the project**
- During the project, a particular risk may
 - Appear
 - Disappear
 - Reappear

Assumptions and Constraints

- Document & manage assumptions and constraints
 - Also implicit ones, e.g., in the results of risk identification
- **Assumption**
 - Something believed to be (or become) true but is not (or cannot be) currently verified
 - Example: Requirements will not be modified without adjusting the schedule
 - **Each assumption is a risk**
- **Constraint**
 - External condition imposed on the process or product
 - Example: Scheduled delivery date
 - **Each constraint is a risk**

Risk exposure & management

1. Calculate risk exposure for each identified risk R:
 - What is the **probability** p_R that risk R will actually occur?
 - What is the **effect** E_R of risk R becoming a reality?
 - **Risk exposure** $RE := p_R * E_R$
 - Example: 20% times “we will have to spend an extra 10 hours to fix the problem”
 - » Risk exposure == 2 (extra) hours
2. **Address risks with highest exposure early on**
 - Typically: Highest risk is not fully understanding what customers & users want

Or think of it in terms of grades

- Following is just one example
- What is the probability p_R that risk **R** will occur? E.g.:
 - R: Implement wrong features, when not talking with users
 - p_R : 95%
- What is the effect E_R of risk **R** becoming a reality?
 - E_R : Lose 15 points (e.g., get B grade instead of an A)
- Risk exposure $RE := p_R * E_R$
 - $RE = 0.95 * -15 = -14.25$



IN-CLASS EXERCISE: IDENTIFY YOUR TOP PROJECT RISKS

Identify your top project risks

- Get together with your team
- **Identify the top 2 project-specific risks facing your project**
 - E.g., about certain feature or certain library you plan to use
 - Rank by risk exposure RE, recall:
 - **RE := (probability that risk will occur) * (effect of risk becoming reality)**
- Post your top-2 risks & their RE to class Teams chat

Top-N risk list ($N \approx 3$)

- Ranking by **consensus** of those affected by risks
- For each risk, include:
 - Short description of risk
 - Risk exposure (RE)
 - » Probability (p)
 - » Effect (E)

Finding actual users & customers and getting their feedback

USERS & CUSTOMERS

User & Customer Feedback is Key

- Top predictor for CSE 3311 project success
- Do you have access to actual people who can give you frequent (at least monthly) feedback on project?
- Feedback can use various channels
 - In-person
 - » Show them your pencil/paper or other UI prototypes
 - » Tryout (e.g., hand them your phone and observe them)
 - » Live demo
 - » Some teams stood outside UTA library & asked random students
 - Online via video conference, chat, ...
 - Email
 - Etc.

Amazon Obsesses Over Customers

- Following quotes are copy-paste from Amazon
 - I just formatted and **bolded** some parts
 - <https://aws.amazon.com/executive-insights/content/how-amazon-defines-and-operationalizes-a-day-1-culture/>

“There are many aspects of maintaining an agile, Day 1 start-up mentality. For us, **the most important one**—the bedrock upon which Amazon’s culture rests—**is customer obsession.**”

“Leaders **start with the customer and work backwards**”

“[Leaders] work vigorously to earn and keep customer trust. Although leaders pay attention to competitors, they obsess over customers.

Amazon Obsesses Over Customers 2/n

“We start with the customer and **work backwards from their needs**. By digging into their experiences and frustrations and deeply understanding the context behind them, we **avoid inventing in isolation**, or producing a solution in search of a customer.”

“**Customers can provide endless ideas** and inspiration to innovate, and **their needs and desires will drive you to invent on their behalf.**”

“As an example, at **AWS**, around **90% of the features developed come directly from hearing about what our customers need**. The other 10% comes from being close enough to customers that we can invent on their behalf when they don't, or can't, articulate those needs.”

Amazon Obsesses Over Customers 3/n

“This relentless customer obsession has been a central part of Amazon’s approach since the company’s literal Day 1 (one of the early names Bezos considered for the company was Relentless.com—type that into your browser and see where it takes you!).”

“By maintaining a customer-obsessed culture and working backwards from your customers’ needs, your innovations are better set up for success right from the start. **Being inspired by your customer’s needs is also likely to open yourself up to innovate in many more areas than you may have otherwise.**”

Your **innovations are not constrained by how you can accomplish them in your current environment**, but rather by how big they can be and how much they can delight your customers.”

Paul Graham in 2005 started Y Combinator, probably the most successful startup incubator to date




Paul Graham ✓
@paulg




The Y Combinator application deadline is today. I asked myself what's the most valuable advice I could give, per word, for getting into YC.

Explain what you've learned from users.

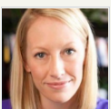
← → ↺ ycombinator.com

 "Y Combinator is the best program for creating top-end entrepreneurs that has ever existed."


Marc Andreessen, General Partner, Andreessen Horowitz


 "I doubt that Stripe would have worked without YC. It's that simple. Acquiring early customers, figuring out who to hire, closing deals with banks, raising money -- YC's partners were closely involved and crucially helpful."

Patrick Collison, Founder, Stripe (YC S09)

 "I've been fortunate to engage with the YC community at past events over the last few years, and always walk away impressed with the passion and caliber of talent that YC brings together."

Julia Hartz, Co-Founder & CEO, Eventbrite

 **Make something people want.**



ycombinator.com
Apply to Y Combinator | Y Combinator
To apply for the Y Combinator program, submit an application form. We accept companies twice a year in two batches. The program includes dinners every ...






9:16 AM · Sep 12, 2022 · Twitter Web App

khVM

<https://twitter.com/paulg/status/1569329072127238146>

Y Combinator Top Companies List

ycombinator.com/topcompanies/breakthrough

Top PrivatePublicFeaturedBreakthrough Companies					
NAME	RANK	COMPANY DESCRIPTION	STATUS	BATCH	HQ
 Stripe stripe.com	#1	Payments infrastructure for the internet.	—	S09	San Francisco, CA, USA; Remote
 Instacart instacart.com	#2	Same-day grocery delivery and pickup service.	—	S12	San Francisco, CA, USA; Remote
 DoorDash doordash.com	#3	Restaurant delivery.	Public	S13	San Francisco, CA, USA
 Coinbase coinbase.com	#4	Buy, sell, and manage cryptocurrencies.	Public	S12	Remote; San Francisco, CA, USA
 OpenSea	#5	The world's largest NFT marketplace.	—	W18	Remote; New York, NY, USA



Paul Graham @paulg · 13m

Replying to @paulg

If you find yourself thinking "I haven't learned anything from users; I have a lot of assumptions about them, but I haven't talked to or observed the behavior of any," bingo!

Not that anyone in that position is ever so aware of it.

4

3

57



Aaron Slodov @aphysicist · 12m

Replying to @paulg

Any hardtech founders curious about YC can send me a DM, happy to answers questions!



3



culpepperwilliams @cwhello · 21m

Replying to @paulg

Is there a Y combinator for social impact projects?

1



1



Paul Graham @paulg · 19m

You can apply to YC with nonprofits.

1



2





Paul Graham ✓
@paulg



Even if you only thought of the idea yesterday, you can still talk to users. In fact, that's the first thing I'd do after working out the first version of most ideas: run it by some potential users to see how they reacted.

9:44 AM · Sep 12, 2022 · Twitter Web App



FP Lee @0x_fp · 1h



Replying to @paulg

how does one find users to talk to?



1



Paul Graham ✓ @paulg · 9m



If you can't answer that, you should probably build something else.



Bottom line: You need access to actual users (and customers) & be able to get feedback from them frequently

<https://twitter.com/paulg/status/1569329072127238146>

“Someone May Need X”

- No clear path toward talking with an actual user or customer
- Built on what team assumes to be realistic requirements
 - Which unfortunately often turn out to be unrealistic
- These type of projects have often **struggled in previous semesters**



“Our User is a Family-member”

- This scenario has worked well several times in previous semesters
 - Easy to talk to such a user/ customer frequently
- Risk factor: One person with this personal connection drops out of the class
 - This has happened in a previous year and forced the other team members to pivot to a different project vision halfway through semester



“We Are Our Own Customer”

- In some cases the team decides to build something for themselves
- In previous semesters this has worked out well in most cases, e.g.,
 - Team had person who was into gaming
 - Person wanted to create a certain type of game
 - Got other team members excited (became a shared vision)
- Risk factor of this person dropping class still relevant

Finding (More) Users / Customers [BD]

- Start with people you know directly
 - Family & friends
- People you meet in dorm, on campus, etc.
- Consider contacts in your address books
 - Contacts on social media (LinkedIn, etc.)
- Consider attending meetups or other social events related to your project idea
- At the end of a conversation about your project, consider asking for additional contacts of people who may also be interested in your project



FINDING USERS / CUSTOMERS

Finding Users / Customers

- Get together with your team
 - Join Teams breakout room “Team N”
- Brainstorm whom your project will use as users / customers
 - Either use vision statement from last class or another one
 - Brainstorm names
 - Brainstorm concrete strategies for recruiting concrete people
- Post results to the meeting’s Teams chat

Develop a Problem Presentation [BD]

- Goals:
 - Capture your current hypotheses about users/customers
 - Elicit information from customers
- Ideally you never need to use this presentation with users or customers
 - **They should be the ones who are talking**, not you

Team's hypothesis about customer's problems	Current available solutions	Team's proposed solutions
1.	1.	1.
2.	2.	2.

Problem Meeting [BD]

- If needed: Describe team's hypotheses (column 1)
- Ask potential user/customer:
 - What they think the problems are
 - If you are missing problems
 - How would they rank the problems
 - Which are most-solve vs. nice-to-have
 - What is it we should have asked you
- Remember: **Goal is to encourage discussion**

Understand Current Solutions [BD]

- Goal: Understand
 - How they currently solve the problem
 - What they think how their peers solve the problem
- If needed: Describe current solutions (column 2)
- Ask potential user/customer:
 - What they think the solutions are
 - How they would rank the existing solutions
 - Who has similar problems

Describe Your Solution [BD]

- If needed: Show your current solution (column 3)
- **Watch for body language**
 - Can you describe your solution for user/customer?
 - Do you have to explain it for 20 minutes and they still do not understand what you are talking about?
- Ask user/customer
 - How does team's solution compare to existing solutions

To Summarize

- Project vision
- Top-3 risk list
- Competitors
- Core use-cases / screen shot flow
 - Other specification documents as needed
- Software development plan
- The users & customers you are talking with
- The documents should be in synch, e.g.:
 - Core use cases should match vision
 - Plan should support vision & attack biggest risks early on

Isn't That a Lot of Documentation?

- Create (only) those documents that add value
 - Feel free to add / remove documents as you go along
- But I recommend you start all of those documents
 - Encourages you to start planning
- Make sure you can access, rework, refine them
 - Key in iterative process
- “[P]lans are useless, but planning [is] indispensable”
 - General Eisenhower

References

- [BD] Steve Blank & Bob Dorf: “The startup owner’s manual”. K&S Ranch. 2012.
- [CL] Craig Larman: “Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development”. 3rd ed. PH, 2004.
- [CS] Carolyn Snyder: “Paper prototyping”. Morgan Kaufmann. 2003.