Lecture 3

Review of Software Processes

Top-Down Design

- Break down a big idea into small digestible parts
  - Eating an Elephant
    - Cut into small pieces and chew
- Deductive reasoning is employed in our reductionism
- Can be fatally flawed from a bias or mistake at the top level

Bottom-up Design

- Piece together small understood components to build a larger system
  - Often as we go along
  - Think “building with Legos”
  - More inductive reasoning employed with small examples implying a likely larger conclusion
  - Organic strategy can get messy and not point to a useful generalization

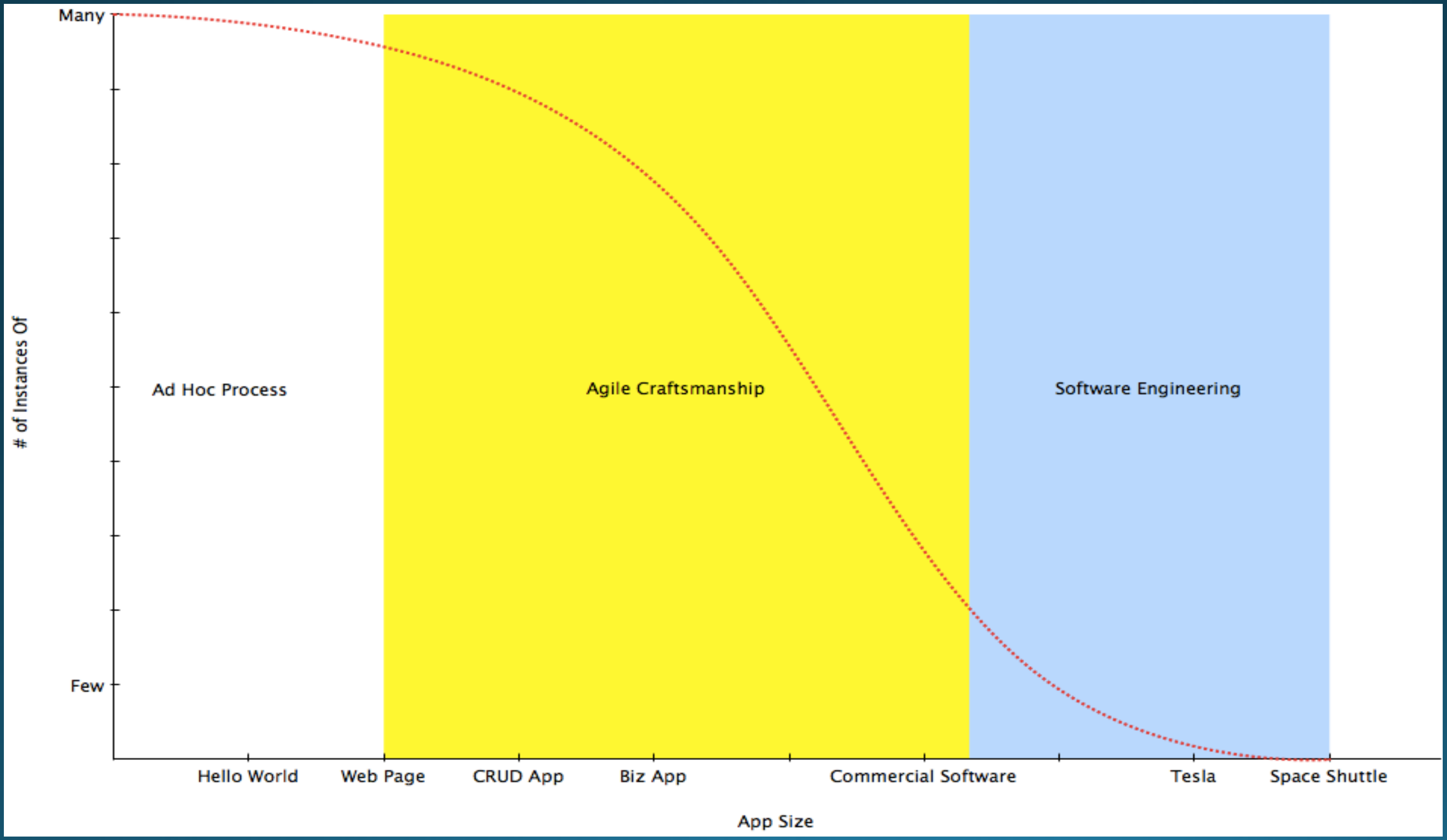
Top and Bottom

- Reality is when approaching problems you often need both approaches.
  - High level planning **<u>and</u>** work it out as you go!
    - Example: Writing my book(s)
- We will see a tension between the extremes in our SE models which suggest a continuum as opposed to an absolute blend

One Size Won’t Fit All

- Depending on situation, circumstance, type of product, etc. SE methodologies make no (or make perfect) sense
  - Airplane control software
    - “Maybe we should really plan this out”
  - My hot new social software for dogs
    - “My gut says this is a winner! We need to get it out there!!!!”

Meeting the Range of Needs



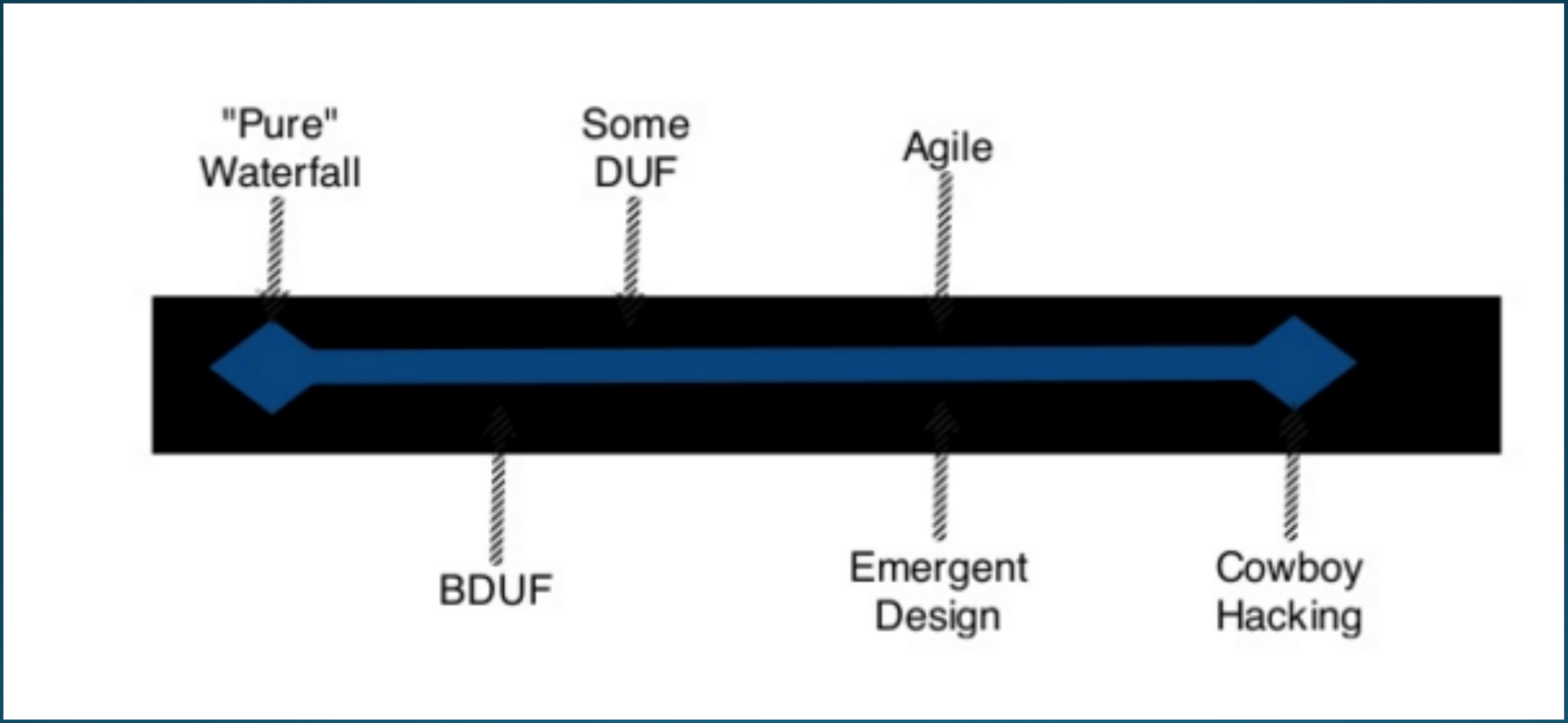
The Need for Process

- We need process to build software
  - reliably
  - on time
  - on budget
  - with quality
- ...and not have it be a miserable experience

Goal: Moving to Knowing

- Regardless of method utilized a successful development processes requires knowing where are in a project. How it is going, how long it will take to finish, etc.

Spectrum of Process



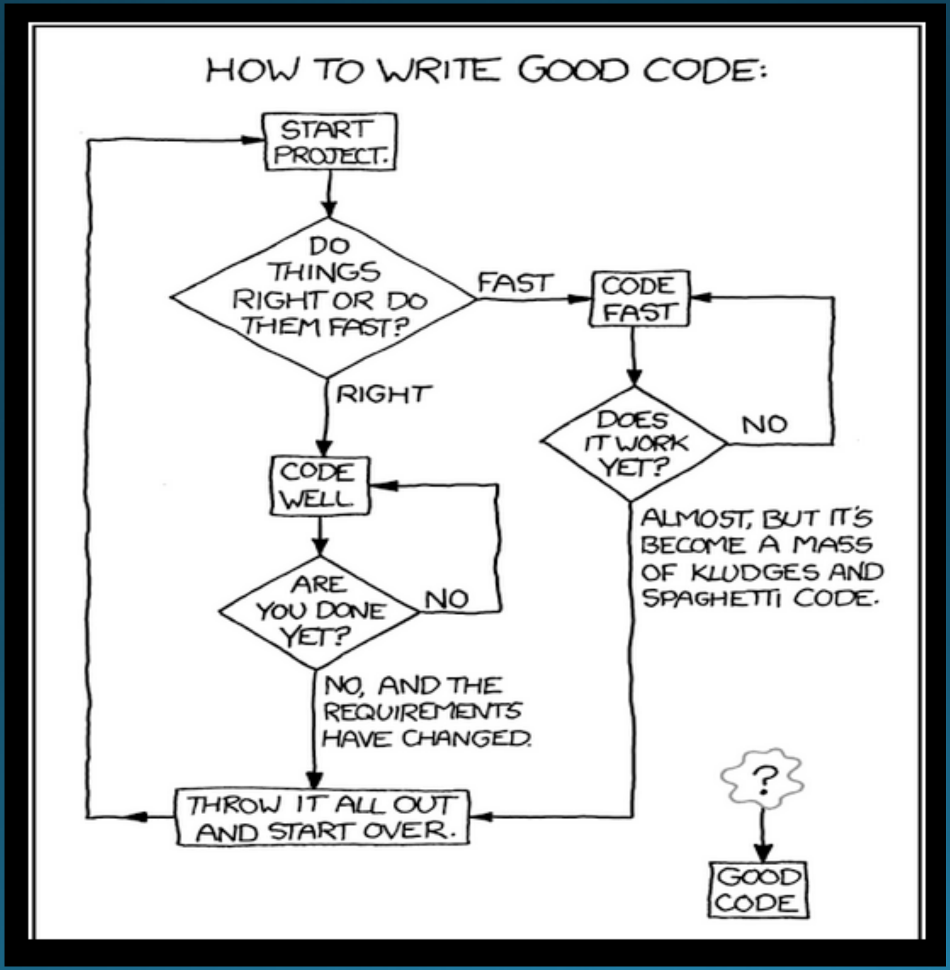
Code and Fix

- Little design, just start coding
  - Comes from naiveté - "This looks easy."
  - Schedule pressure - "It's due Monday!"
- Favorite method of Cowboy coders
- Q: Do we use this one at UCSD? Successfully?
  - Actually yes and yes*

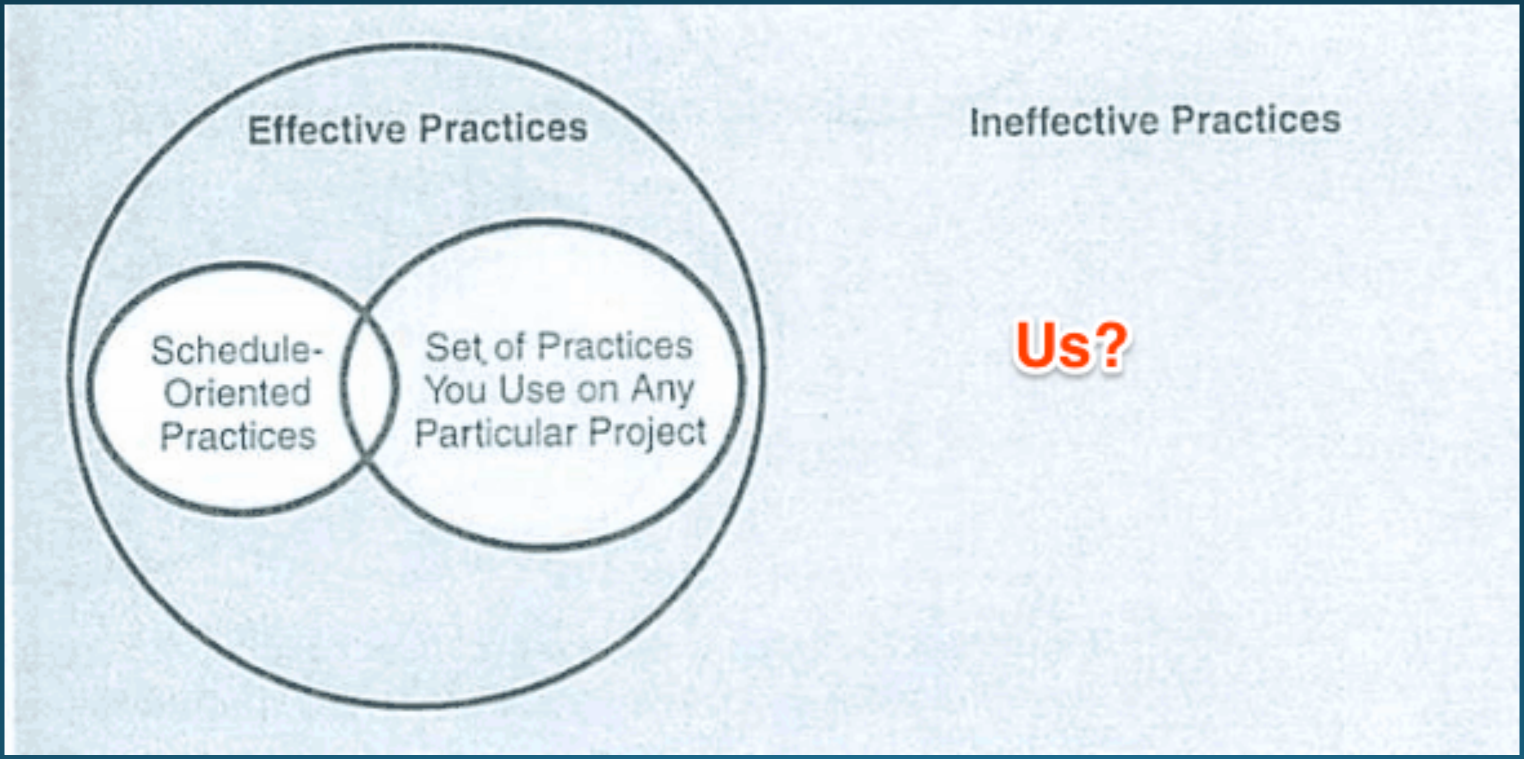
Cowboys are cool!?

- Code cowboys can be the hero <u>sometimes</u>
  - Freedom can inspire
  - No rules! - introduce new tools, langs, …
  - Border crossings can be useful
  - Sometimes faster - no boss blockers
- **<u>Lot's</u>** of room for failure though

The Infinite Loop of Doom


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
RIGHT
CODE WELL.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

We need to move to


Effective Practices
Schedule-Oriented Practices
Set of Practices You Use on Any Particular Project
Ineffective Practices
Us?

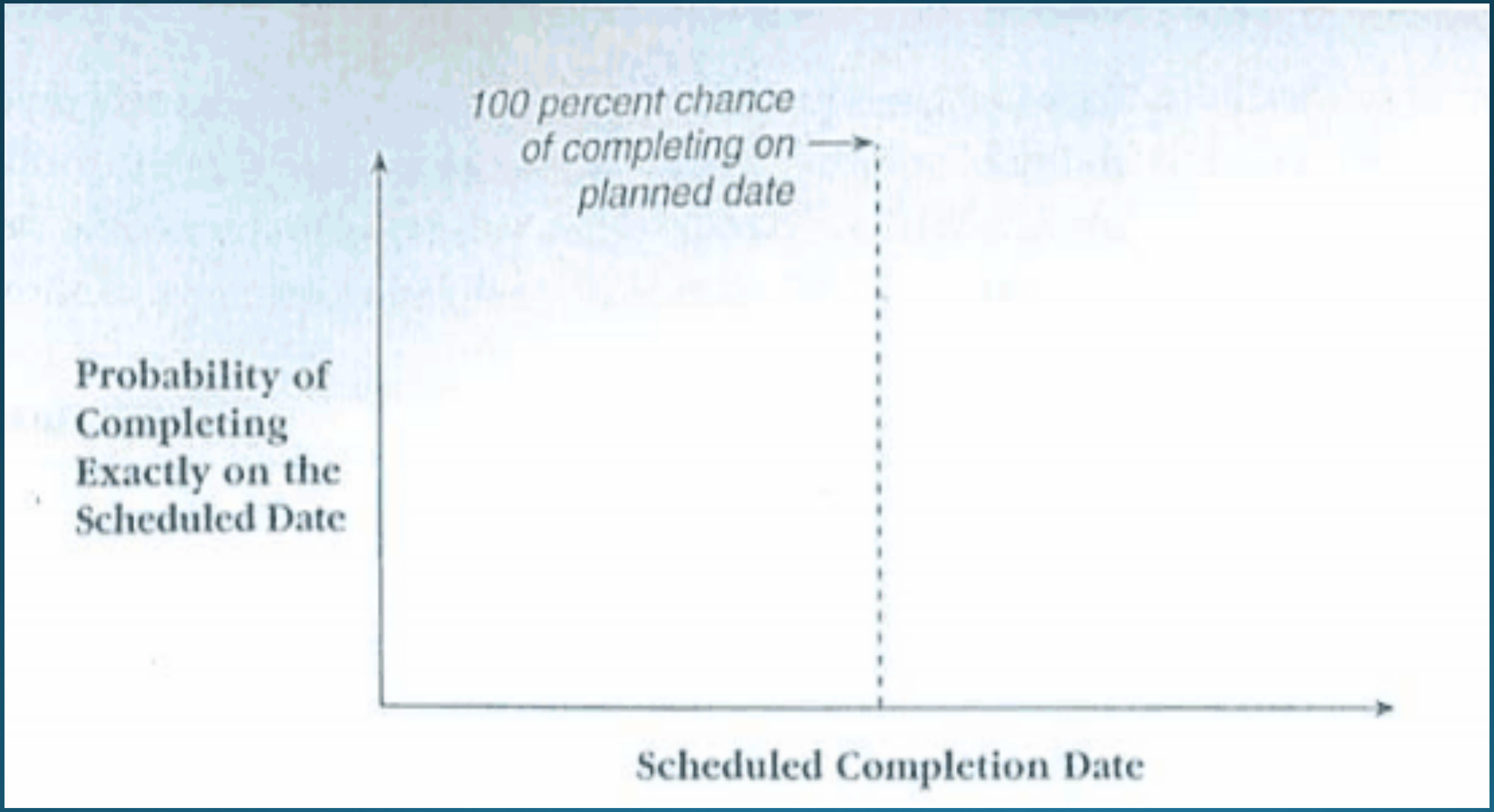
Ch 1, *Rapid Development*, Steve McConnell

It’s Done When It’s Done

- Blizzard approach
  - Nice if you have the capital and buy-in from stakeholders and customers to act like this
- Can lead to laziness and vaporware
  - http://en.wikipedia.org/wiki/Duke_Nukem_Forever
- Generally a no schedule approach just won’t work
  - …but loved to see you be in the position to do this (nice work if you can get it!)

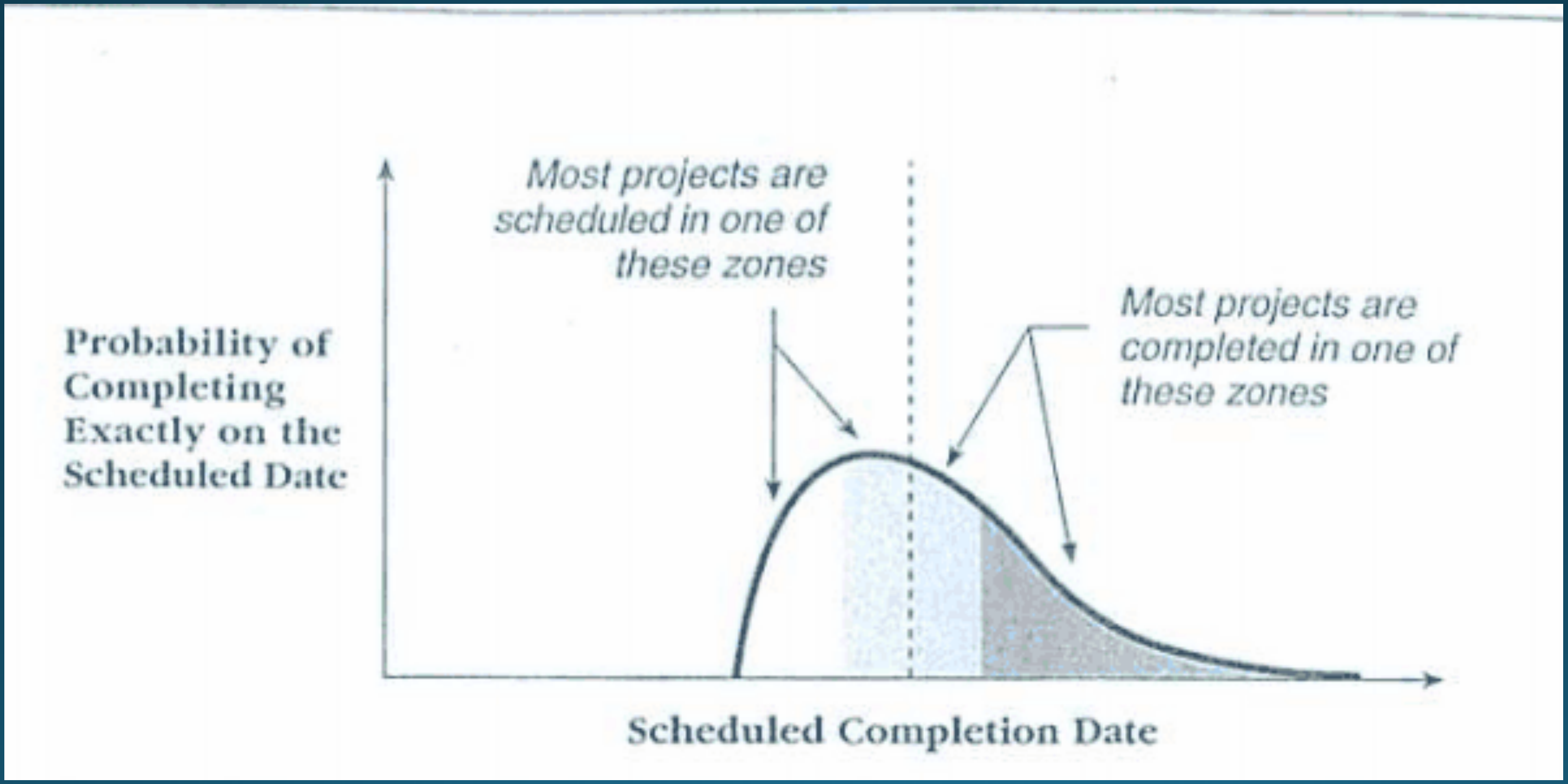
Scheduled Insanity

- It's never this...



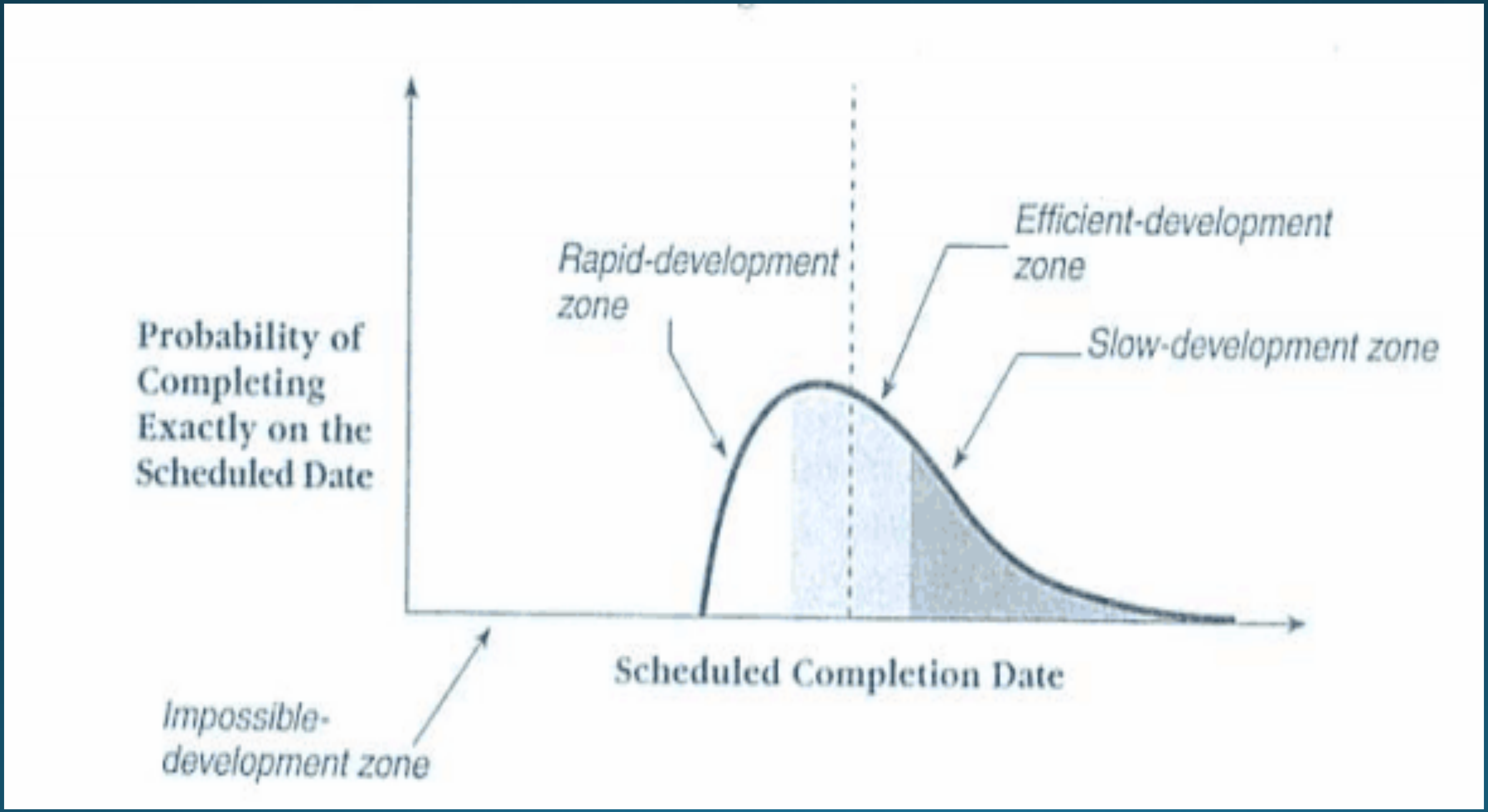
Ch 6, *Rapid Development*, Steve McConnell

More like this



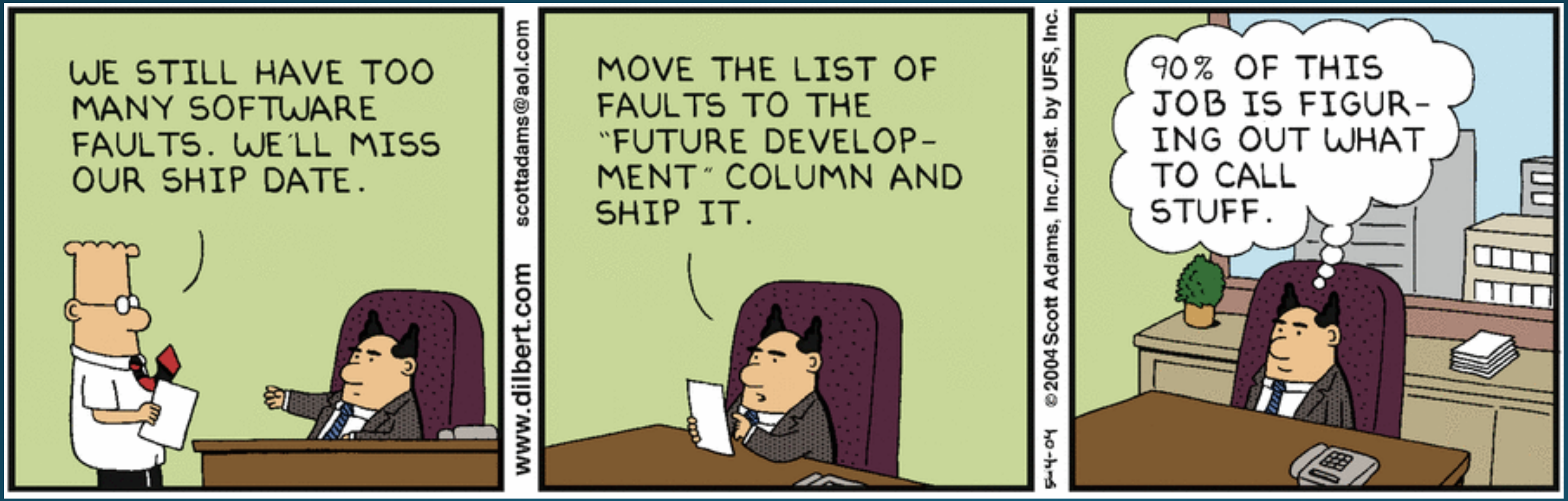
Ch 6, *Rapid Development*, Steve McConnell

What it means



Ch 6, *Rapid Development*, Steve McConnell

Common Software Project Management Technique for Deadlines


WE STILL HAVE TOO MANY SOFTWARE FAULTS. WE'LL MISS OUR SHIP DATE.
MOVE THE LIST OF FAULTS TO THE "FUTURE DEVELOPMENT" COLUMN AND SHIP IT.
90% OF THIS JOB IS FIGURING OUT WHAT TO CALL STUFF.
www.dilbert.com scottadams@aol.com
© 2004 Scott Adams, Inc./Dist. by UFS, Inc.
5-4-04

Brook’s Law

- Adding people to a late project doesn’t generally deliver desired results
  - On-boarding time is generally larger than assumed
  - Communication path multiplication
  - We lack the shared hive mind to recover history and details necessary to be productive quick enough

Some people have called the book the bible of software engineering. I would agree with that in one respect: that is, everybody quotes it, some people read it, and a few people go by it.

(Fred Brooks)

Flavors of Approach

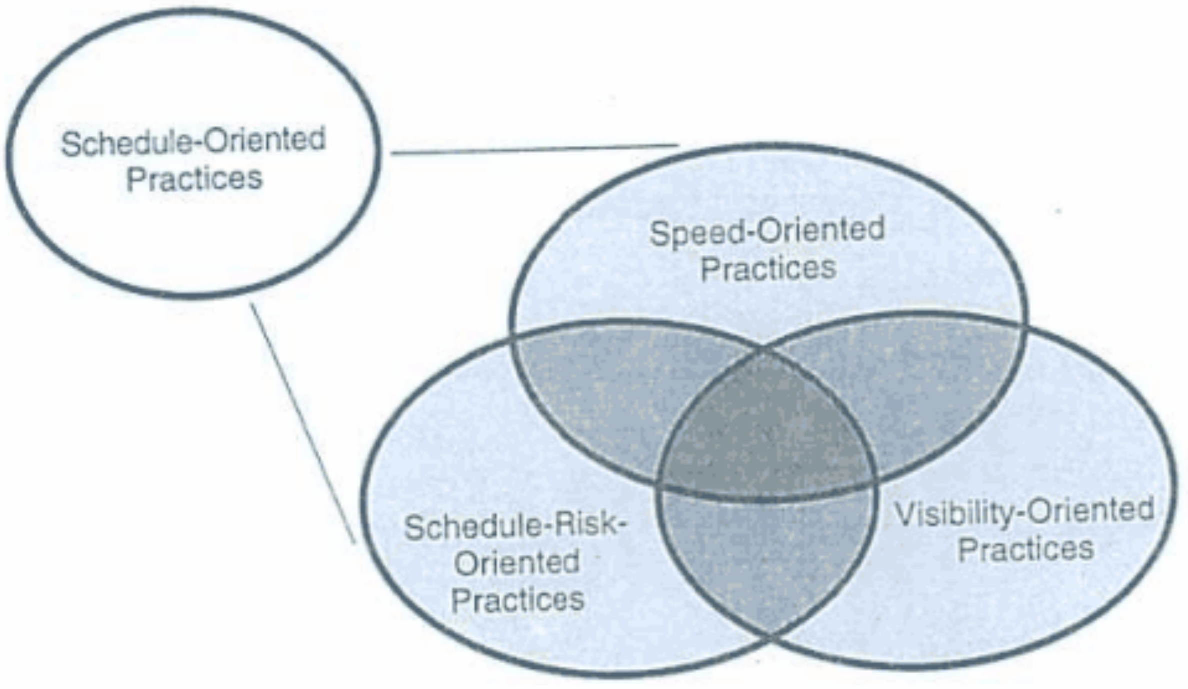
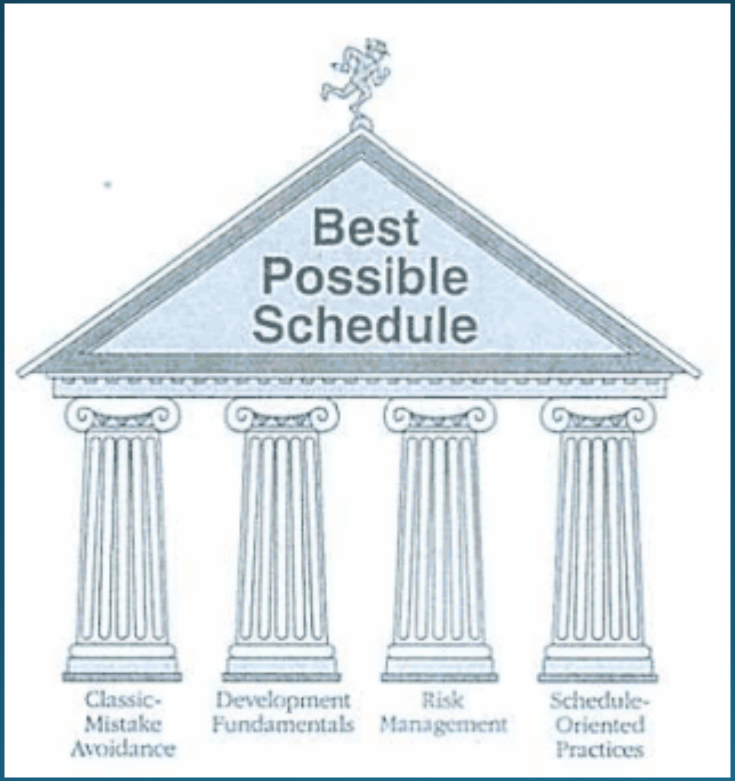

Schedule-Oriented Practices
Speed-Oriented Practices
Schedule-Risk-Oriented Practices
Visibility-Oriented Practices

Figure 1-2. *Schedule-oriented practices come in three kinds: practices that enable you to develop faster, reduce schedule risk, and make your progress visible.*

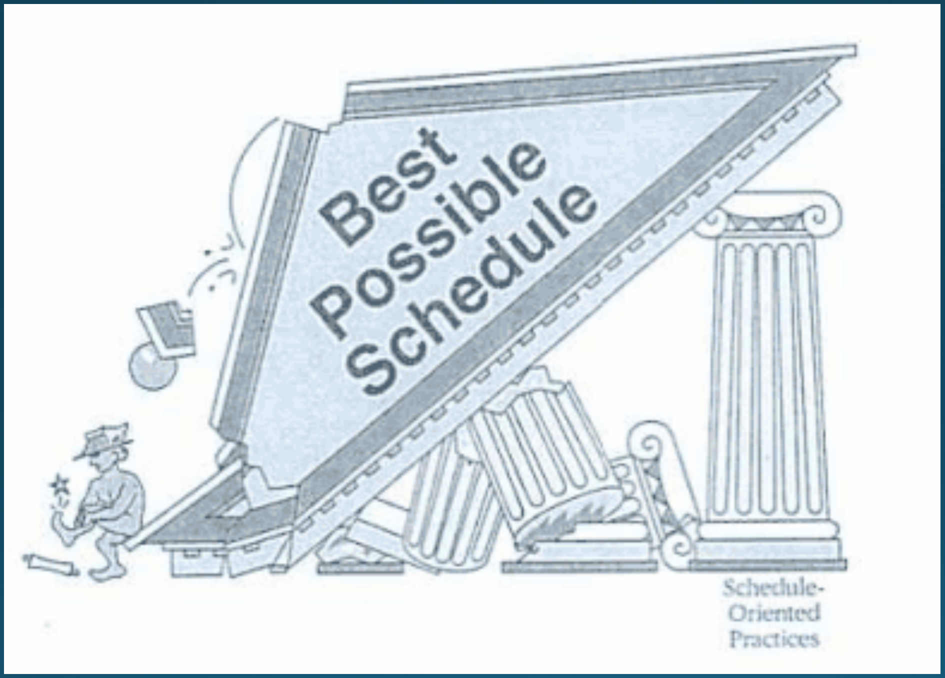
Ch 1, Rapid Development, Steve McConnell

We Hope for This



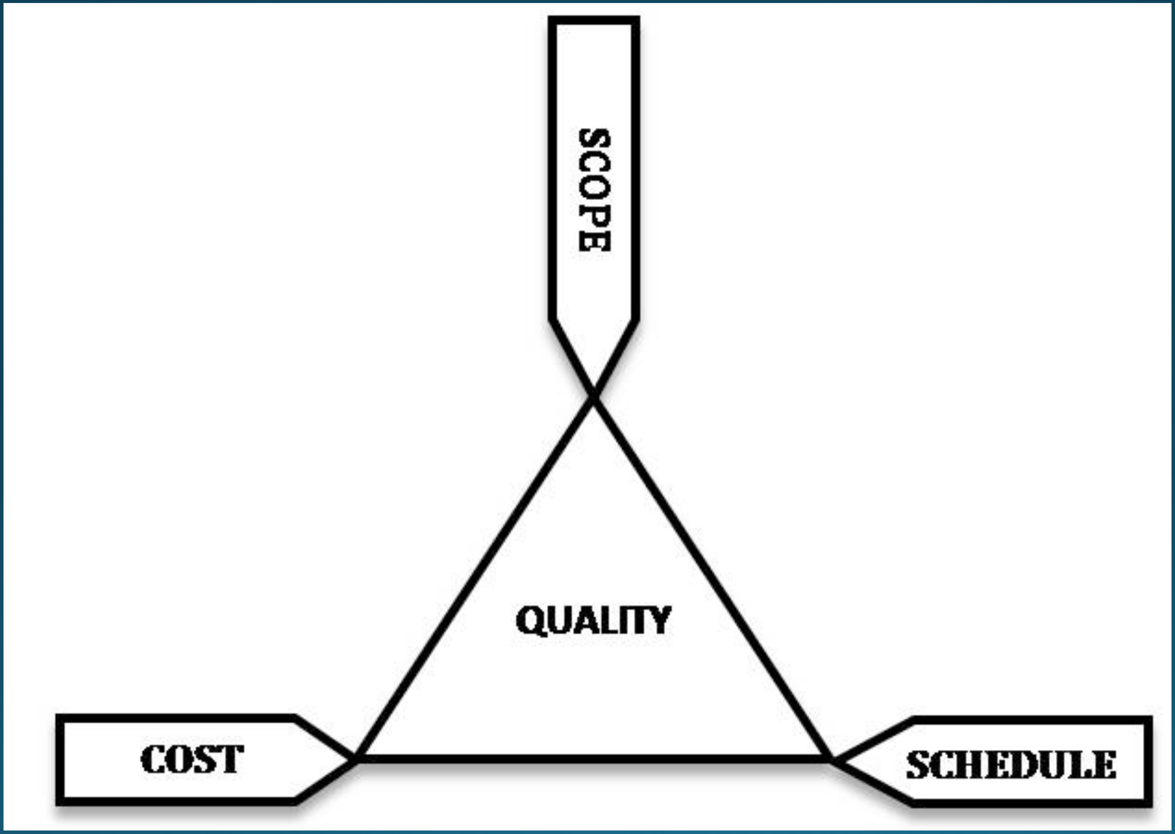
Ch 2, *Rapid Development*, Steve McConnell

Often Get…



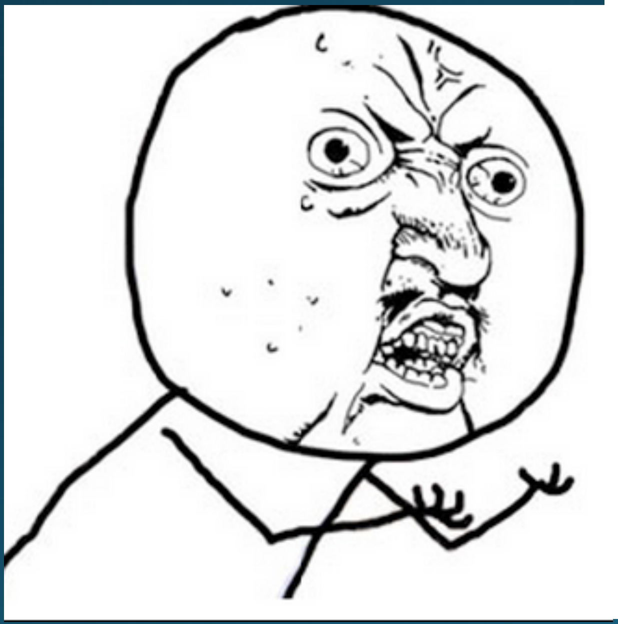
Ch 2, *Rapid Development*, Steve McConnell

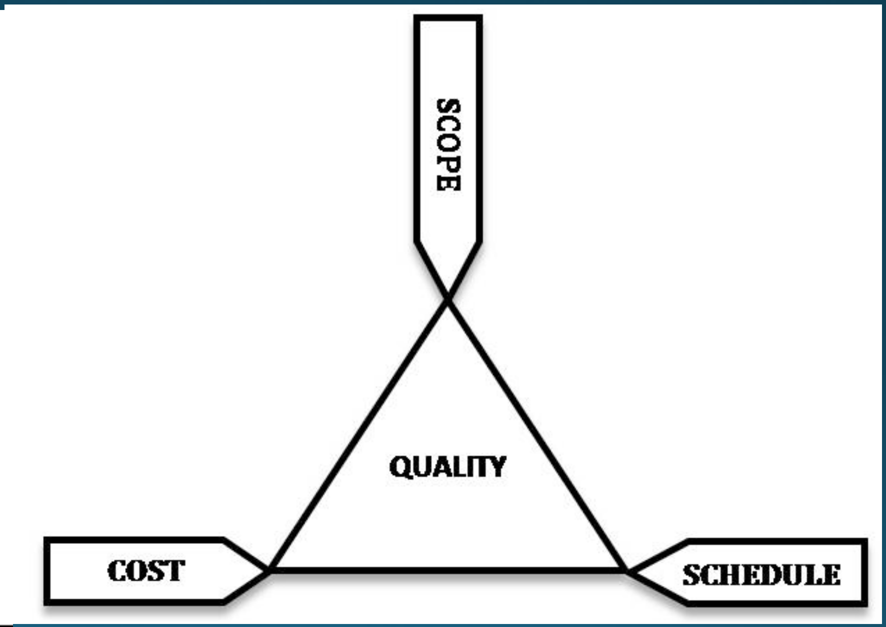
Tension Occurs



Pull a handle it effects other and of course interior

Yep for many…



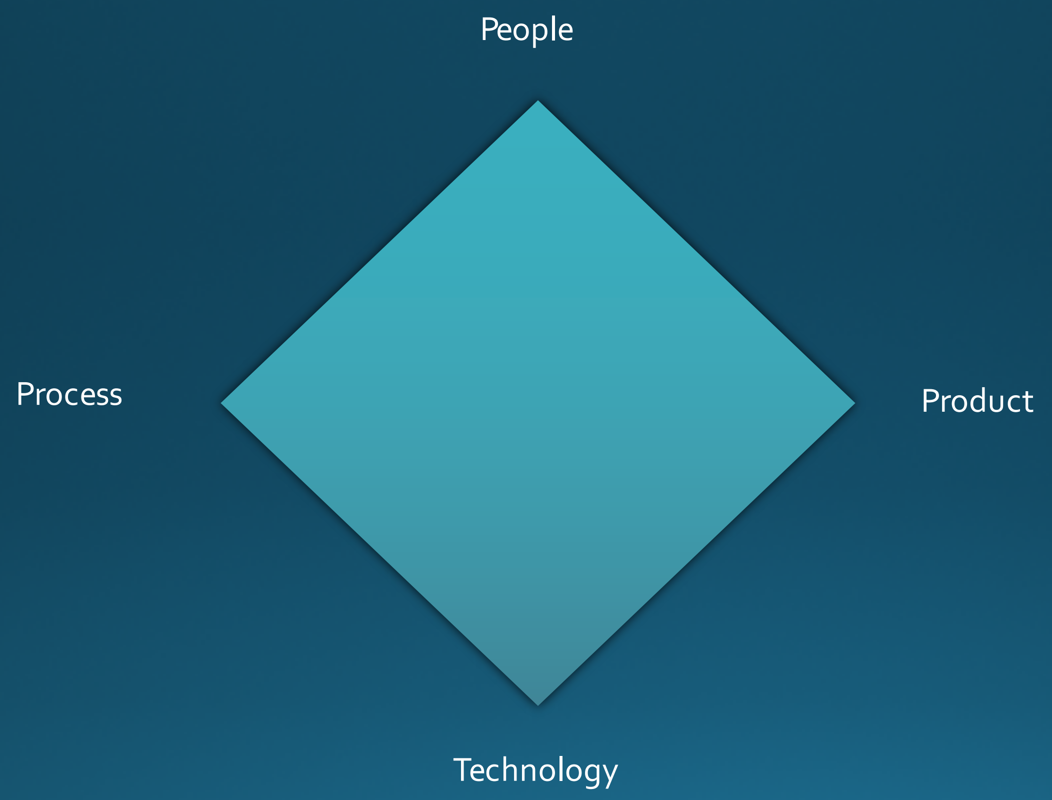


Y U No Give Me **ALL!?**

Professor’s Favorite Word

- The Professor’s favorite word that is the law above all law’s in SE (or maybe life)
- There are always ___________________
- Magical thinking
    - “Middle Earth Engineering” – thinking there is one master solution
    - Gandalf (and me) says “You shall not pass!!!”

Tension Always Present



Easy Question I hope: Which might be the hardest to deal with?

Example Process Risks

- Bad schedules (overly optimistic mostly)
- Not enough risk management
- Not enough planning and design
- Inefficient front-end time squeezing back-end time
- Planning abandoned due to time pressure
- Reduced QA efforts
- Not enough transparency (Boss doesn't know enough)
- Many more variations

Example Product Risks

- Unrealistic expectations
- Too many requirements up-front
- Feature creep
- Competitor reactions
- Developer reactions

Example Tech Risks

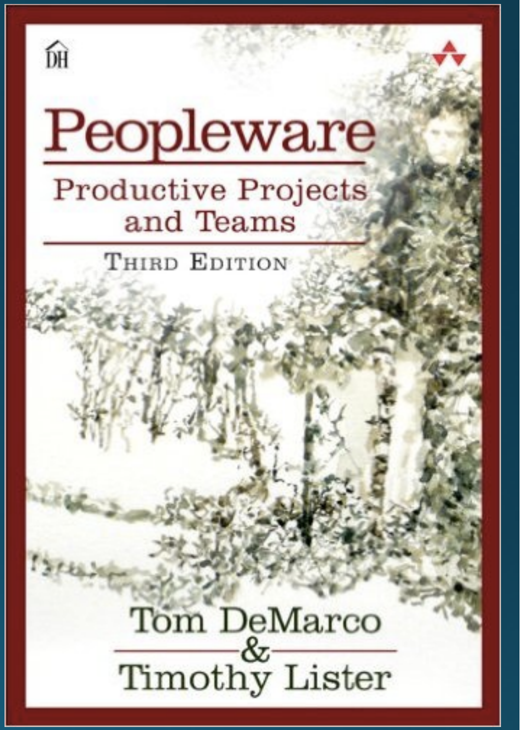
- Silver-bullet beliefs
  - Overestimating value of tooling or lang
- Tool switching costs
- Lack of appropriate tool availability or use
  - Source control, Bug tracking, Test tools, Automation/Deployment, etc.

Example People Risks

- Weak personnel
- Heroics
- Negative personalities
- Wishful thinking
- Politics
- Inappropriate work space
- Lack of buy-in, patrons, etc.

Peopleware

- Peopleware - a term describing role of people in dev and use of computer systems
  - productivity, teamwork, group dynamics, HCI, social effects, market effects
  - Plain old garden variety irrationality and inconsistency
- The biggest variable in your SE life, hard to learn this without experiencing it
- We've given you a flavor of this immediately in this class.



Team Organization Broad Concerns

- Should we all be equal partners?
- Should we all do same thing or different things?
- Should we have a lead developer with assistants?
  - Master chef, surgeon, etc.
  - Should the “big boss” just direct or work?

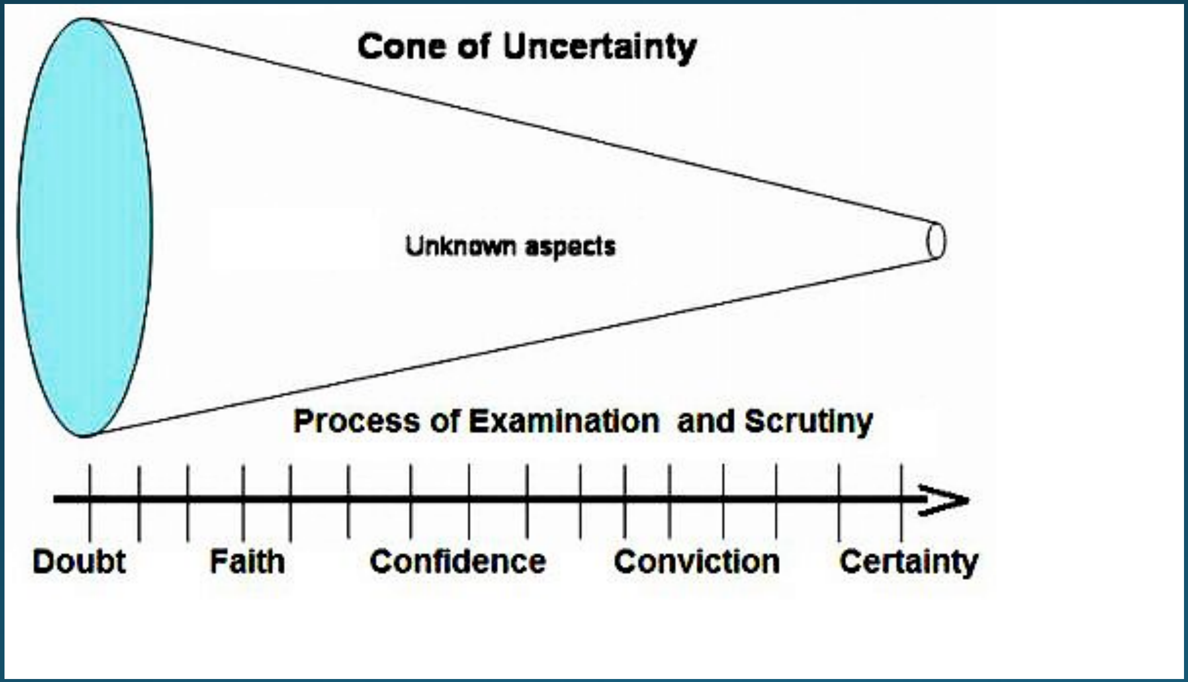
Yes lot's can go wrong

- Some of this can be learned
- Some of this truly must be experienced (sadly) even beyond the peopleware
  - At least know it is coming!
- Have realistic expectations for yourself and relative to others - "SE Nirvana is a myth"
  - Now let's see the two approach types

Big Design Upfront (BDUF)

- All these systems described are of the idea that we should do “Big Think” first
- Theory: If you big think up front and plan things out the investment will pay off
- Obvious Concern: Can we know what is not knowable?
- Obvious Concern: Time taken to plan may outweigh time taken to do or fix.

Facing the Cone of Uncertainty



- I like this one because it shows the general nature of the idea as opposed to some specific SE aspect to this point

Facts #8 and #9

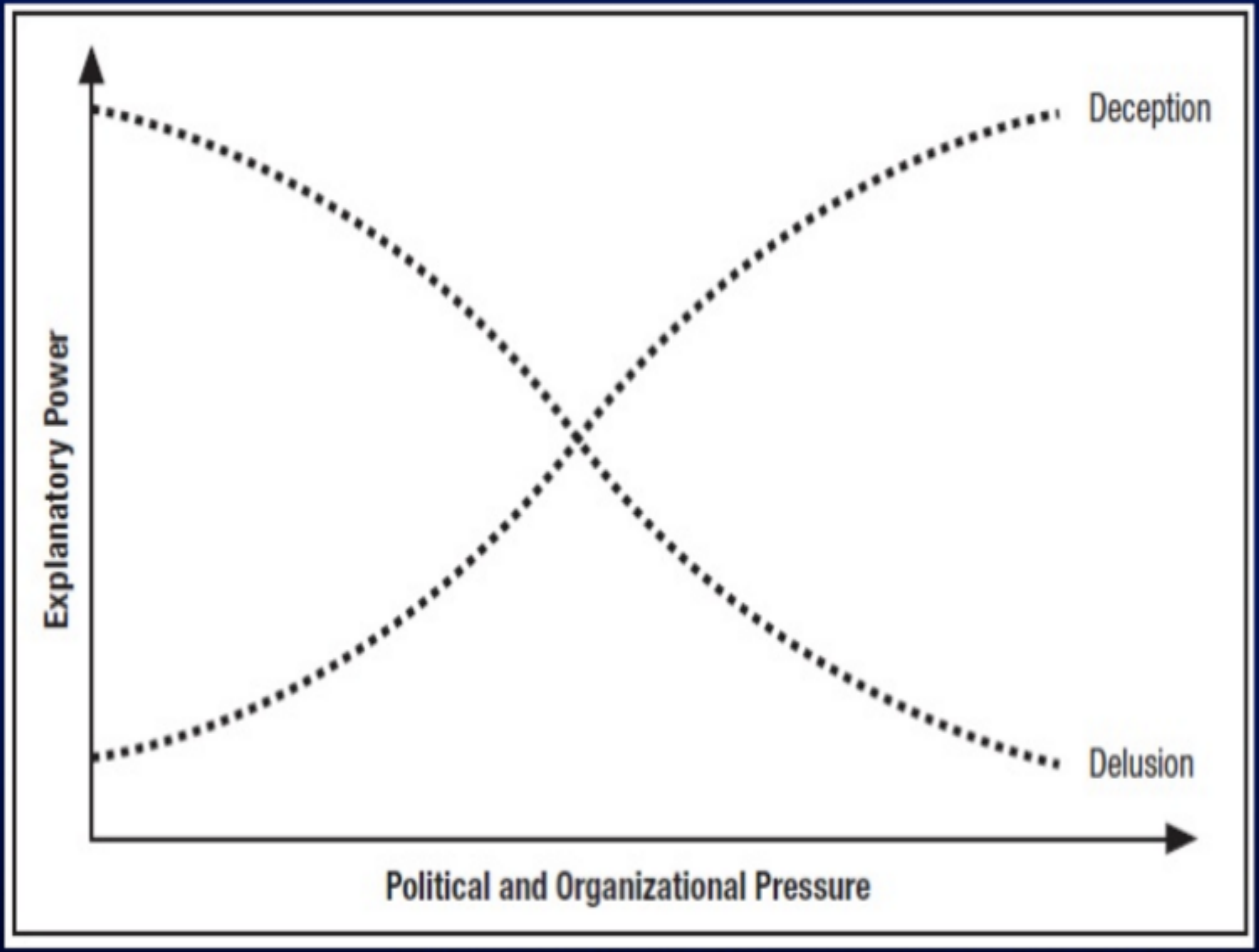
- Fact 8: One of the two most common cause of runaway projects is poor estimation
  - The other is changing requirements
- Fact 9 : Most software estimates are performed at the beginning of the life cycle. This makes sense until we realize that estimates are obtained before the requirements are defined and thus before the problem is understood. Thus estimation usually occurs at the wrong time
  - Solution? Spec phase / requirements phase with one budget and conclusion results in an another budget
  - Doesn't fit with know the answer now, cheapest bidder wins, etc. that we meet in real life. Again human nature for wanting things to not be as complicated as they tend to be leads to risk

Fact #10

- Fact 10: Software estimates generally made by the wrong people (not by the people who will build the software) so they operate with less info then they need for a quality estimate, often not knowing what they need to know

- Example: Joe bidding off Mark's plans
- Example: Peter nearly every class as a joke, not really

Cognitive bias vs. political behavior as cause of chronic underestimations



(Politics)

(Cognitive bias)

B. Flyvbjerg, From Nobel Prize To Project Management: Getting Risks Right, 2006

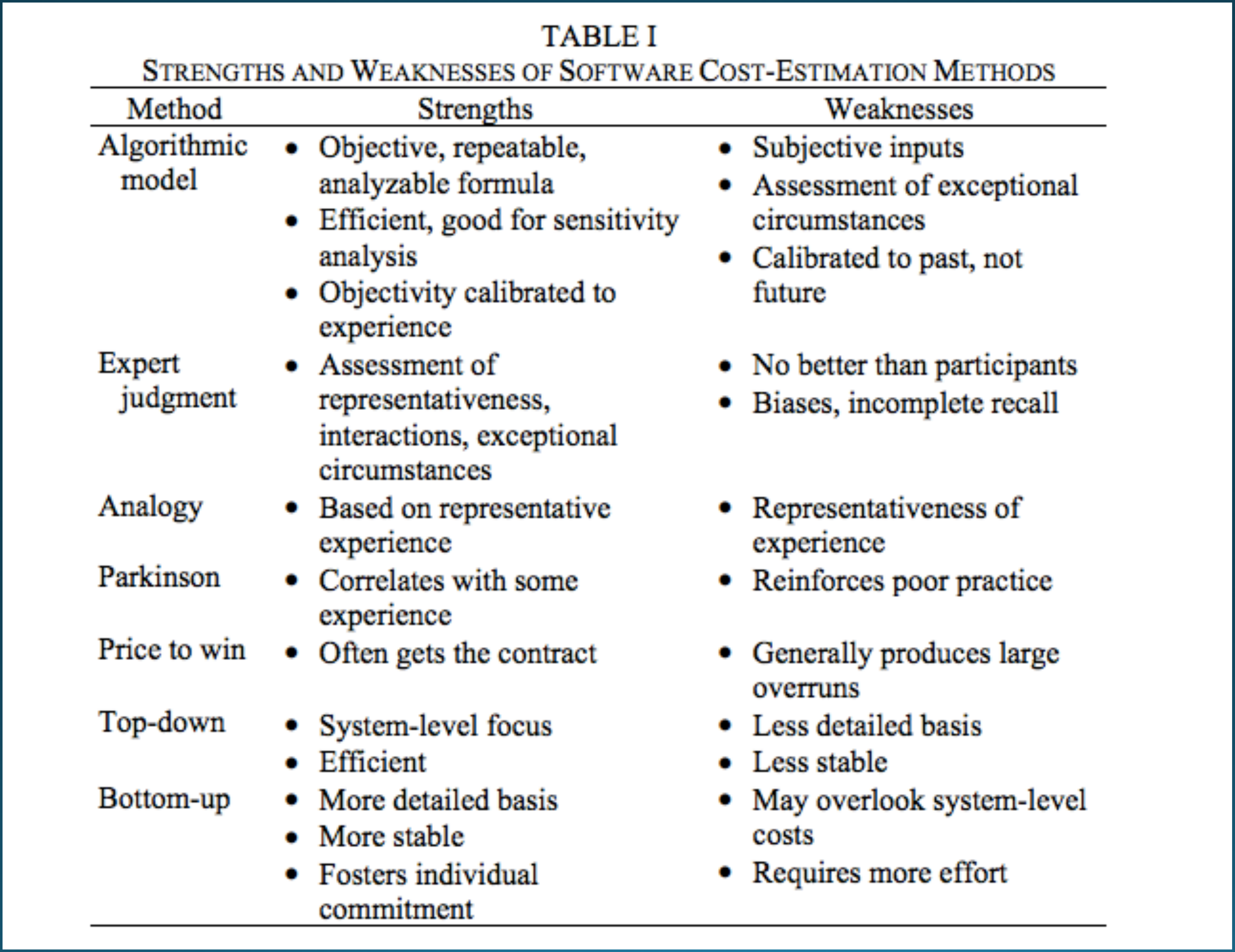


TABLE I
STRENGTHS AND WEAKNESSES OF SOFTWARE COST-ESTIMATION METHODS

Method	Strengths	Weaknesses
Algorithmic model	• Objective, repeatable, analyzable formula • Efficient, good for sensitivity analysis • Objectivity calibrated to experience	• Subjective inputs • Assessment of exceptional circumstances • Calibrated to past, not future
Expert judgment	• Assessment of representativeness, interactions, exceptional circumstances	• No better than participants • Biases, incomplete recall
Analogy	• Based on representative experience	• Representativeness of experience
Parkinson	• Correlates with some experience	• Reinforces poor practice
Price to win	• Often gets the contract	• Generally produces large overruns
Top-down	• System-level focus • Efficient	• Less detailed basis • Less stable
Bottom-up	• More detailed basis • More stable • Fosters individual commitment	• May overlook system-level costs • Requires more effort

Estimate Notes

- Software estimates are rarely adjusted as the project proceeds. Thus the estimates usually aren't corrected
  - Politics tends to get in the way!
- Estimates tend to be faulty, so not hitting time and $ target shouldn't be expected. However we get crazy about it anyway
- There is often a disconnect between management and programmers. A project that failed to meet estimates might be deemed by management a failure, but by the tech workers the most successful project (for a variety of reasons)

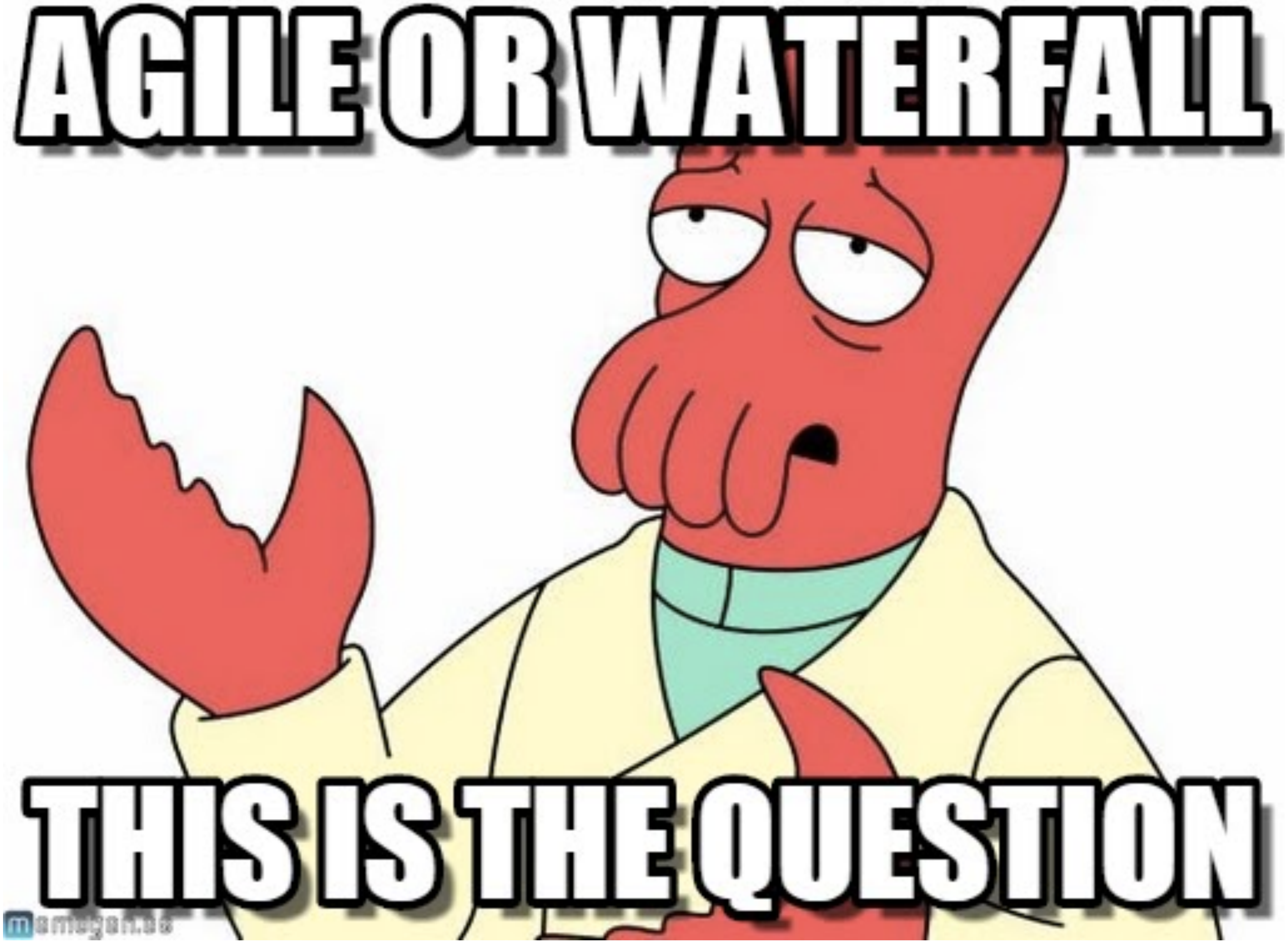
Feasibility - Yes

- The answer to a feasibility study is almost always “yes”
- We tend to like qualified yes, since no is politically toxic.
  - “Yes it is possible, putting a man on the moon in 1968 was possible given the right amount of effort, $ and TIME”
  - Need to make sure that No isn’t about attitude or insecurity but honest assessment. It is easy for the listener to rationalize and keep their delusions
    - Real Life: “The other guys said it can be done in 2 weeks for 2 thousand and you said it will be months and tens of thousands!!?”
- Prof: “Manned space flight to the moon using 1960s technology was feasible, near anything is feasible if the science and engineering supports it but is it reasonable to do and do you have commitment and money to do it?”

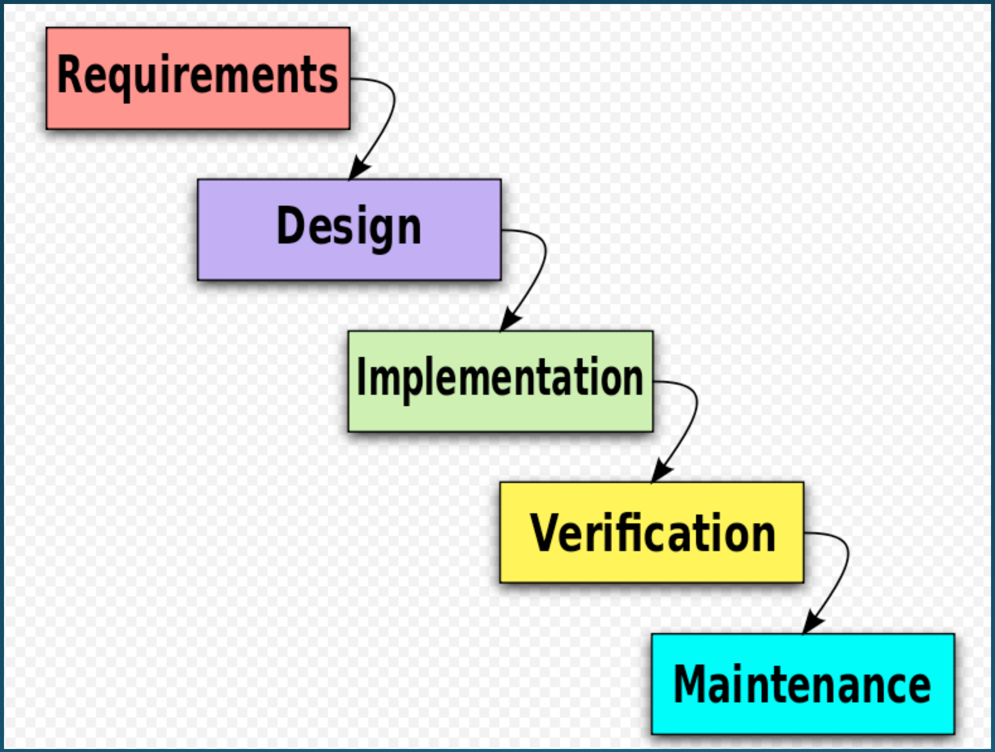
Which projects get done?

"We have found it isn't necessarily the best ones, but those projects for which proponents best succeed in conjuring a fantasy world of underestimated costs, overestimated revenues, undervalued environmental impacts and overvalued regional development effects"

Machiavellian Megaprojects, Bent Flyvbjerg, 2005


AGILE OR WATERFALL
THIS IS THE QUESTION
memegen.es

Waterfalls



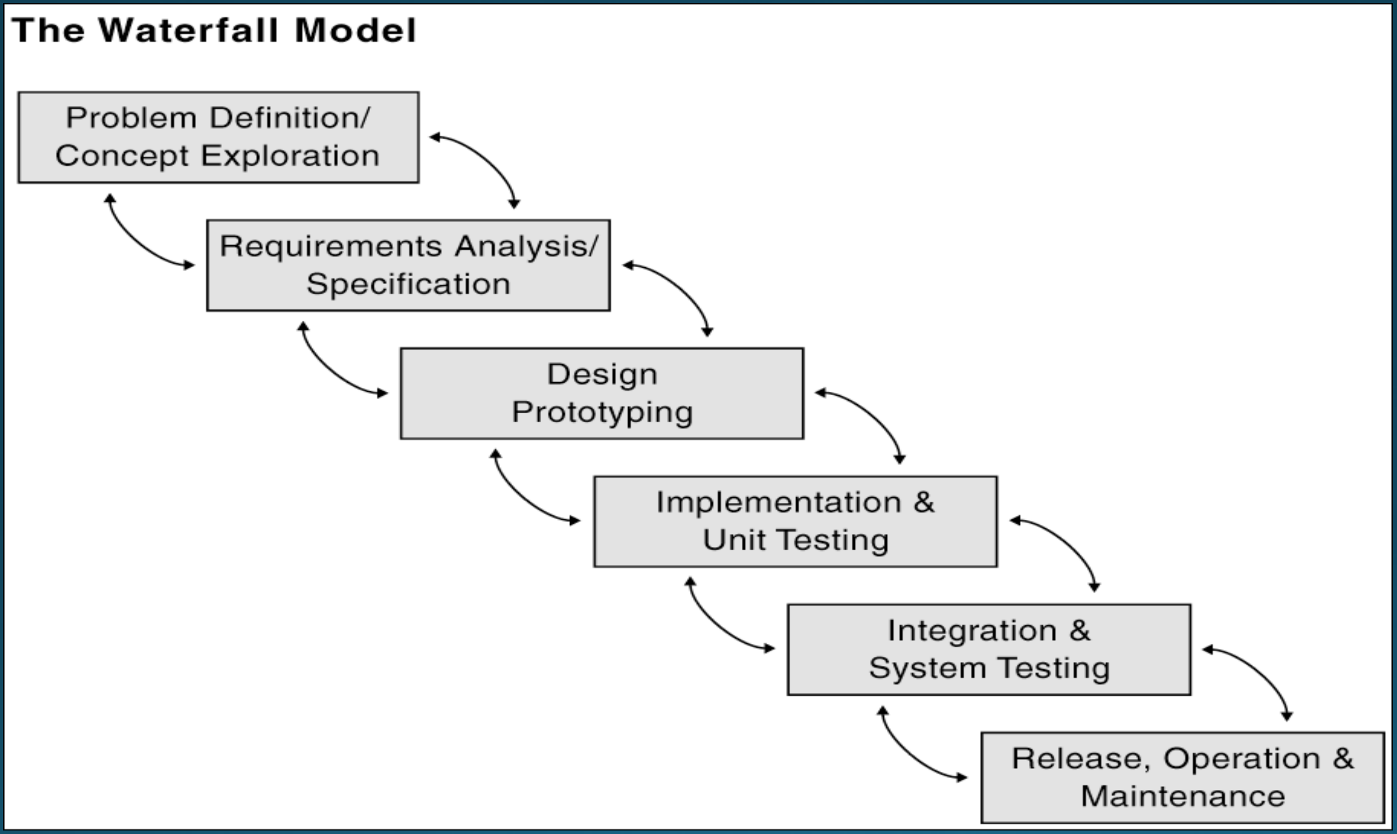
http://en.wikipedia.org/wiki/Waterfall_model

Waterfalls

- Royce’s Waterfall

1. Requirements Specification
2. Design
3. Construction aka “Coding”
4. Integration
5. Testing and Debugging
6. Installation
7. Maintenance

Waterfalls


The Waterfall Model
Problem Definition/ Concept Exploration
Requirements Analysis/ Specification
Design Prototyping
Implementation & Unit Testing
Integration & System Testing
Release, Operation & Maintenance

Web Site Engineering, Powell et al.

Waterfall Notes

- Number and name of steps is arbitrary
  - Unfortunately we will see discrete step boundaries not always clear
- Main point is a plan and document first before proceeding
  - Common methodology in "physical world"
  - Why?
  - In my experience with home building the idea of a pure waterfall is nonsense, there is plenty of "in field" modifications
  - Ike on the next slide says it all when it comes to plans!

"IN PREPARING FOR BATTLE I HAVE ALWAYS FOUND THAT PLANS ARE USELESS, BUT PLANNING IS INDISPENSABLE."
DWIGHT D. EISENHOWER

Waterfall Notes

- Yes it is simple to understand, that is the real reason I think it still works even in what you might consider in appropriate situations
  - It's a counter to the no process "make it up as we go along" and as such it is vast improvement
- Interestingly despite the agelistas thinking this feels right to those who cook, construct, etc. Maybe not it seems as if maybe they haven't seen the chaos in a kitchen or on a job site. So it might not be pure BTUF but at the very least ETUF

Towards Incrementalism

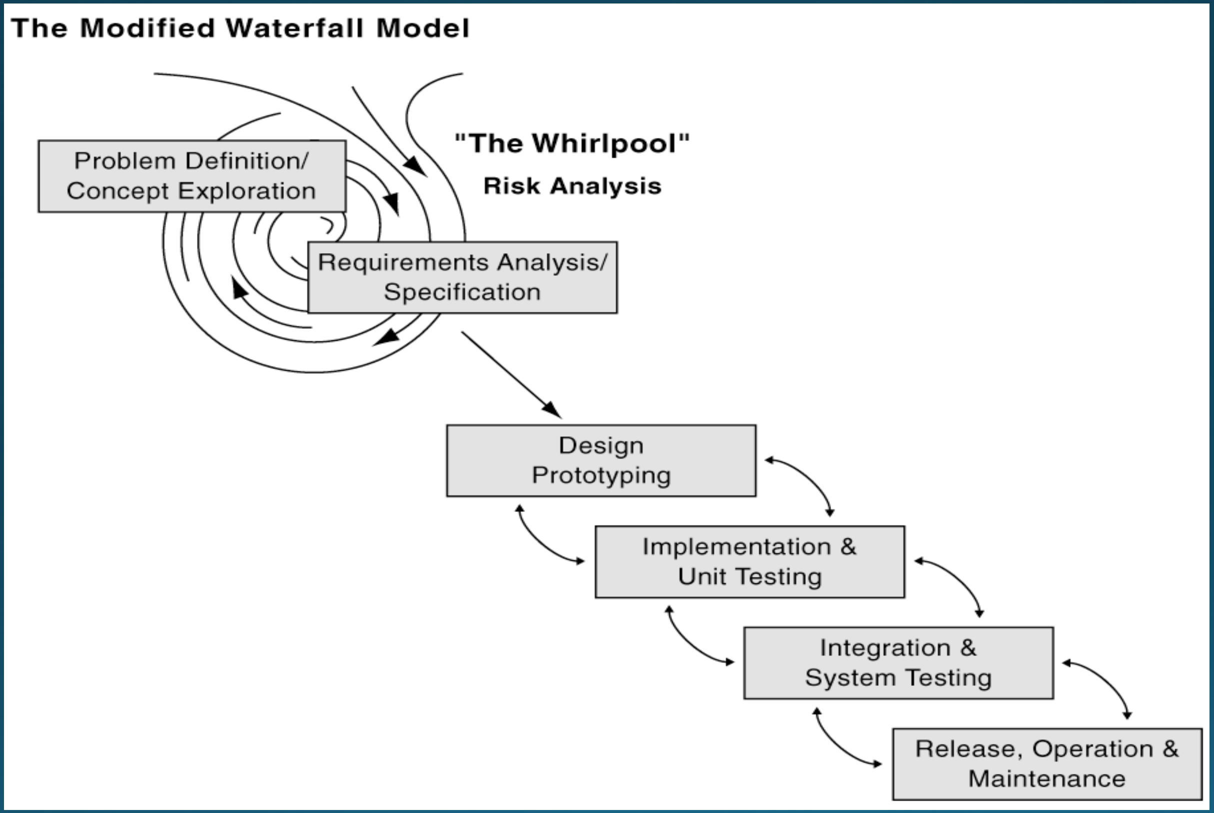
- Rough Design Up Front (RDUF) - plan enough to get going (or my EDUF)
  - What is enough? Temptation to start
- Clearly inching along seems safer than big plan on something with too many unknowns or just plain wrong
  - Unfortunately what happens if you inch your way into a corner?

Modified Waterfalls

- Generally there is concern about risk reduction and dealing with future info
  - Try to avoid jumping in and doing the wrong thing
  - Try to incorporate things you learn on the way to adjust project
- A real truth

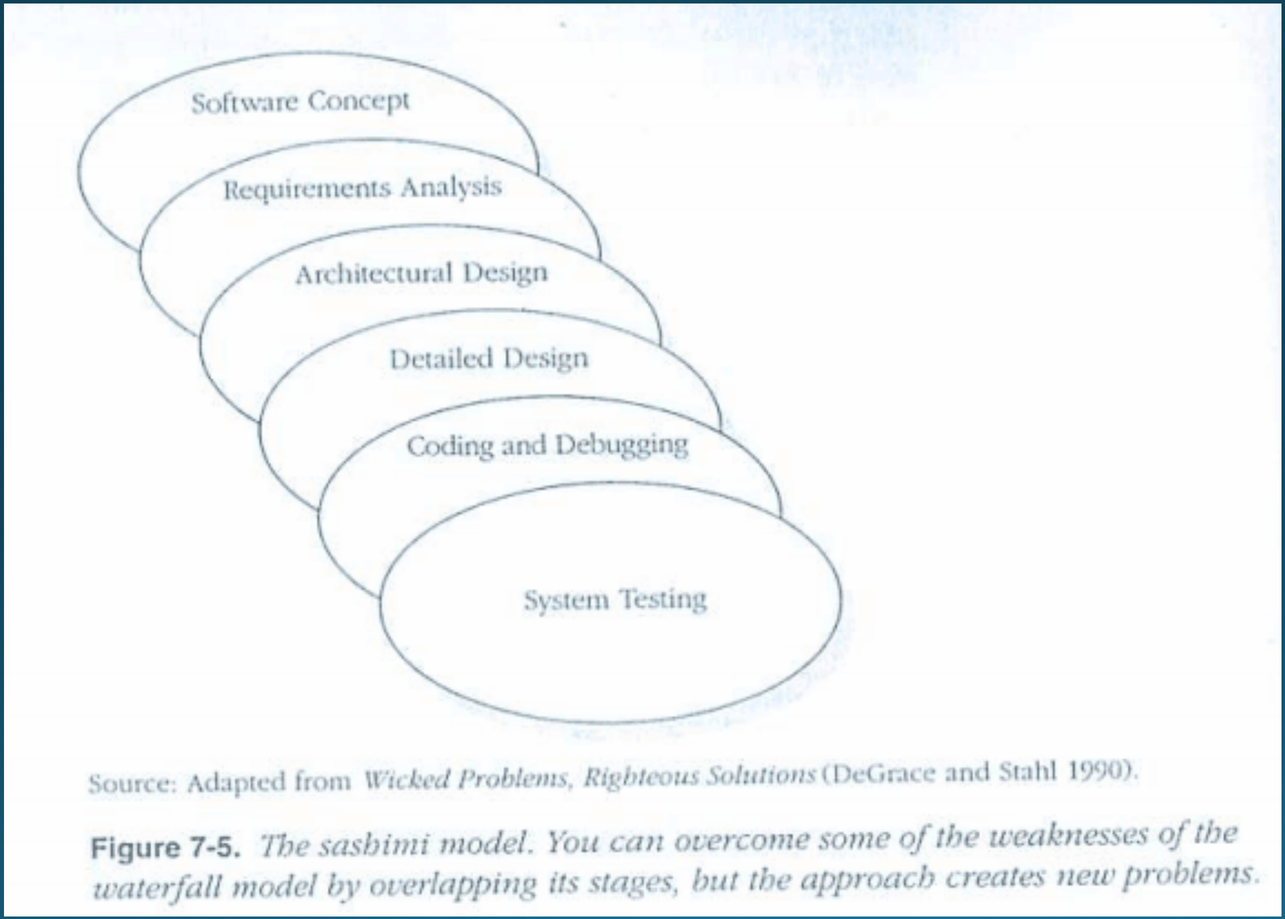
"You just don't know what you don't know until you know it."

Modified Waterfall #1


The Modified Waterfall Model
"The Whirlpool"
Risk Analysis
Problem Definition/ Concept Exploration
Requirements Analysis/ Specification
Design Prototyping
Implementation & Unit Testing
Integration & System Testing
Release, Operation & Maintenance

- *Web Site Engineering* Powell et al.

Modified Waterfall #2



Source: Adapted from *Wicked Problems, Righteous Solutions* (DeGrace and Stahl 1990).

Figure 7-5. *The sashimi model. You can overcome some of the weaknesses of the waterfall model by overlapping its stages, but the approach creates new problems.*

Ch 7, Rapid Development, Steve McConnell

Modified Waterfall #3

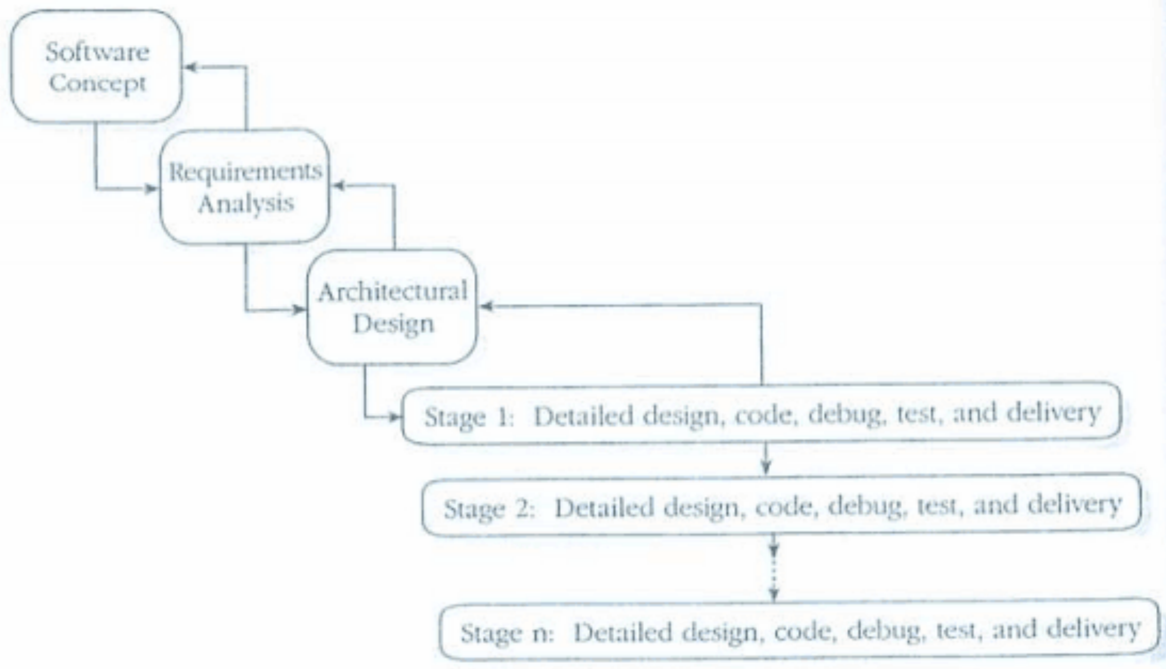

Software Concept
Requirements Analysis
Architectural Design
Stage 1: Detailed design, code, debug, test, and delivery
Stage 2: Detailed design, code, debug, test, and delivery
Stage n: Detailed design, code, debug, test, and delivery

Figure 7-9. *Staged-delivery model. Staged delivery avoids the waterfall model's problem of no part of the system being done until all of it's done. Once you've finished design, you can implement and deliver the system in stages.*

Ch 7, *Rapid Development*, Steve McConnell

Modified Waterfall #4

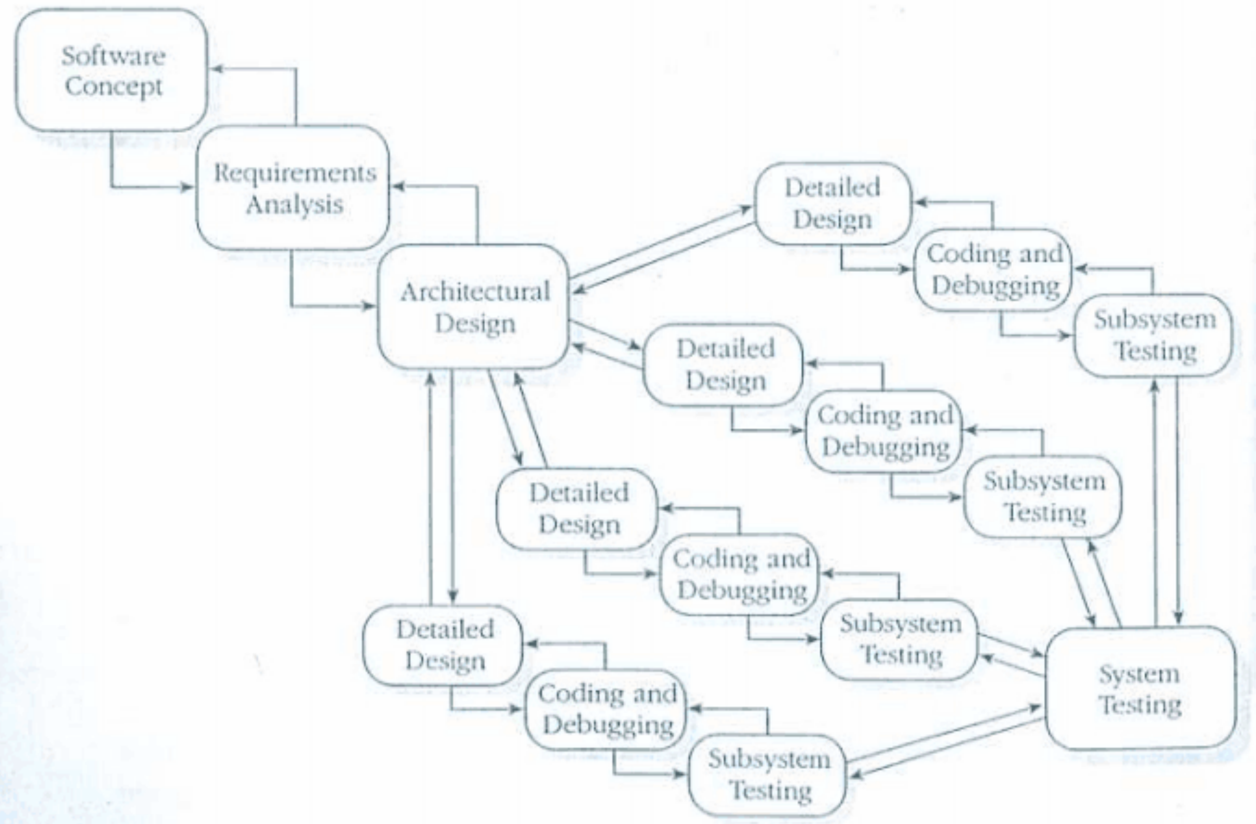


Figure 7-6. *The waterfall model with subprojects. Careful planning can allow you to perform some of the waterfall's tasks in parallel.*

Realization

- It seems as we might want to do some exploration before we dive in
- We can adjust our goals, desires, approaches based upon what we find
- Is there a good way to do this in a guided manner?

Prototyping

- Evaluate an idea without full commitment
- UX in particular benefits greatly from prototyping
- Tempting thought - take a prototype and refine it to production quality
- Challenges
  - Boss: Pretty pixels on screen = near done

Prototype Tradeoffs

Pros

- Reduced time
- Reduce cost
- Better User Involvement
- Work out details and make pretty

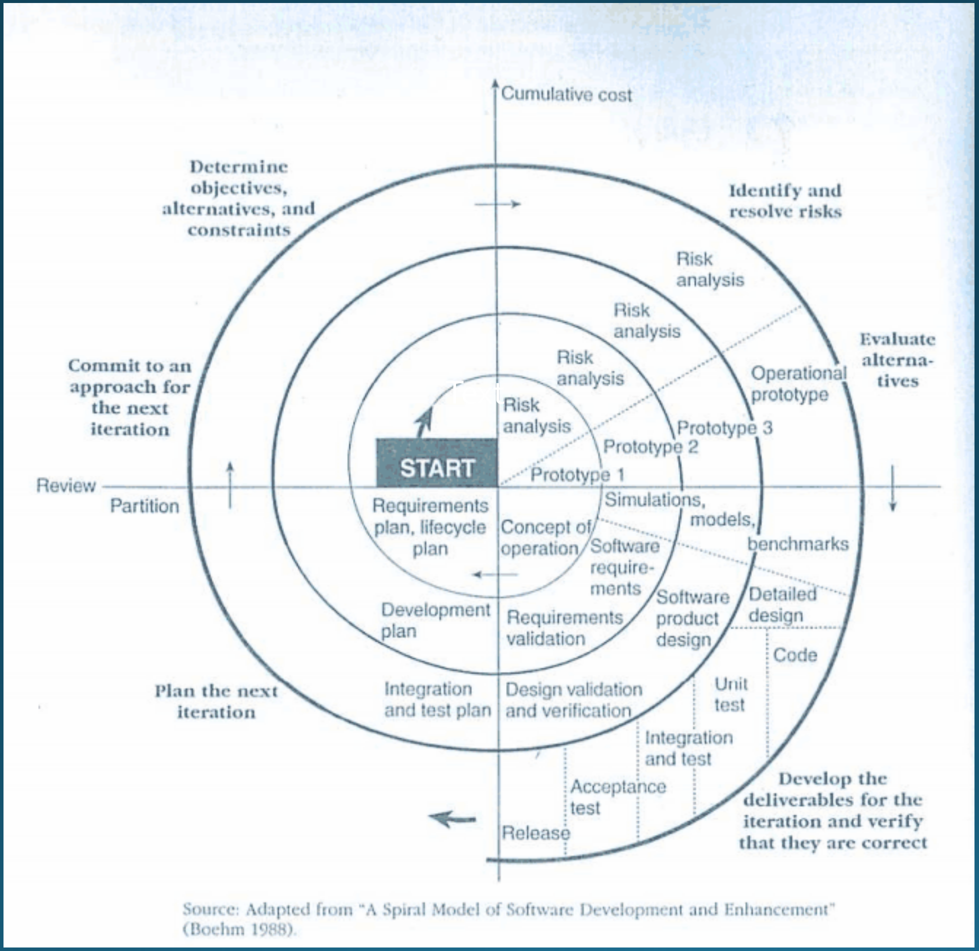
Cons

- User misunderstanding
- “Ugly kids”
- Longer time
- Bigger cost

Extreme Prototyping

- Static prototype
  - PSD Mock-ups and/or HTML mocks
  - Feedback and adjust
- Simulated service layer
  - Full HTML/CSS/JS with mock calls
  - Feedback and adjust
- Services implemented
  - Working app
  - Hope feedback changes focus on service layer after up front work

Spiral Model



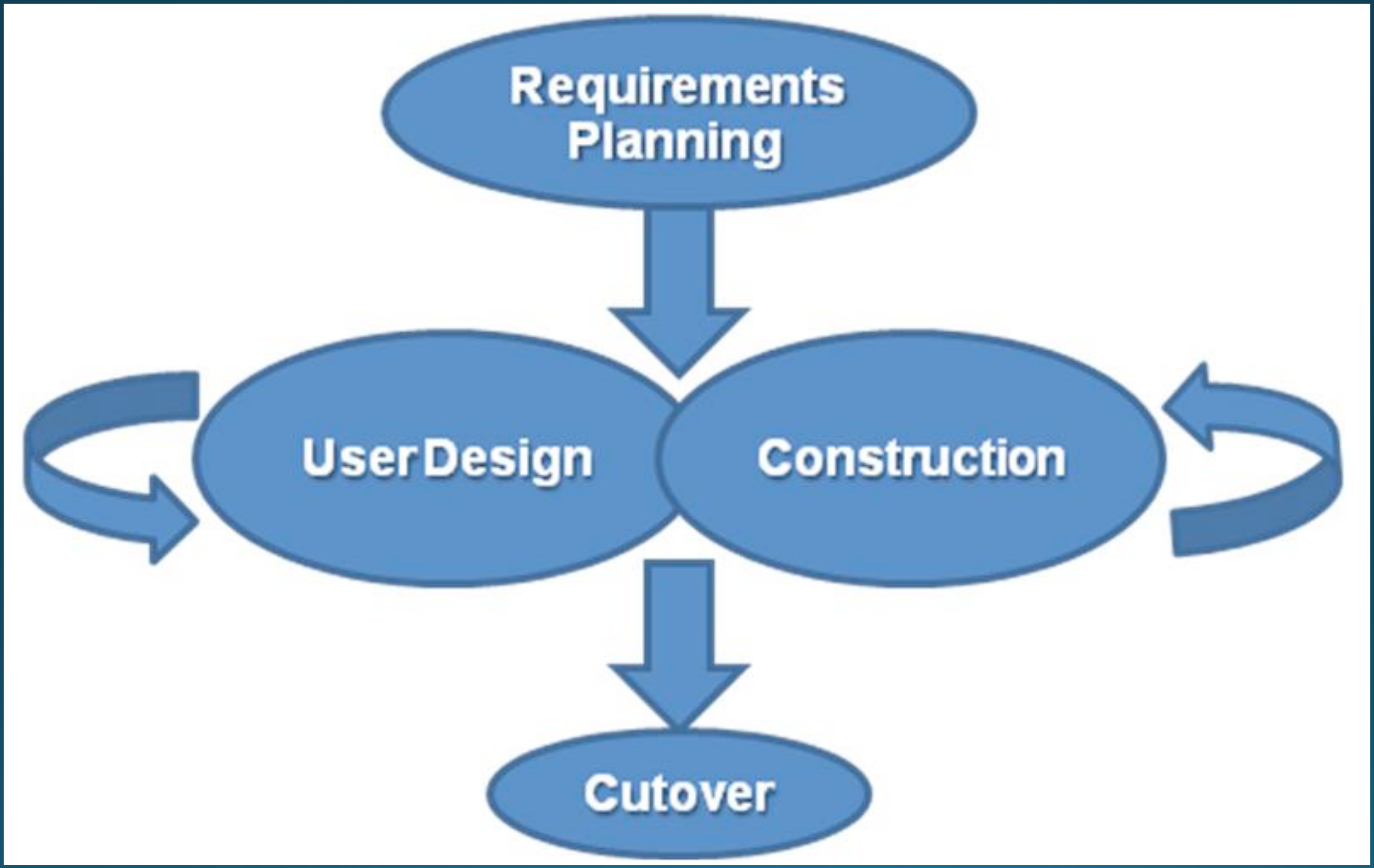
Source: Adapted from "A Spiral Model of Software Development and Enhancement" (Boehm 1988).

Iterative prototyping spiraling outward from small to big

Rapid Application Development

- Rapid Application Development (RAD) favors rapid prototyping over planning
- Like name aims for speed and tries to address the reality of change via iterative prototyping
- We change this name a lot over time: Agile, Extreme (XP), Lean, etc.

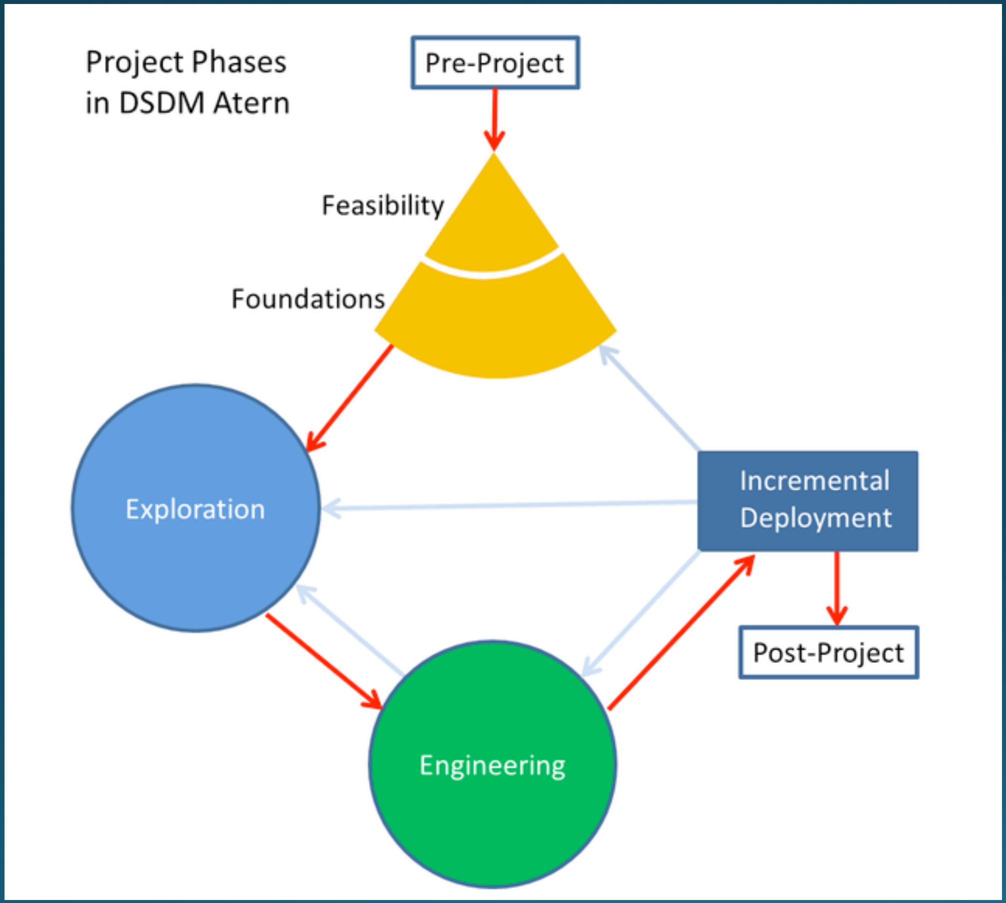
RAD Visualized



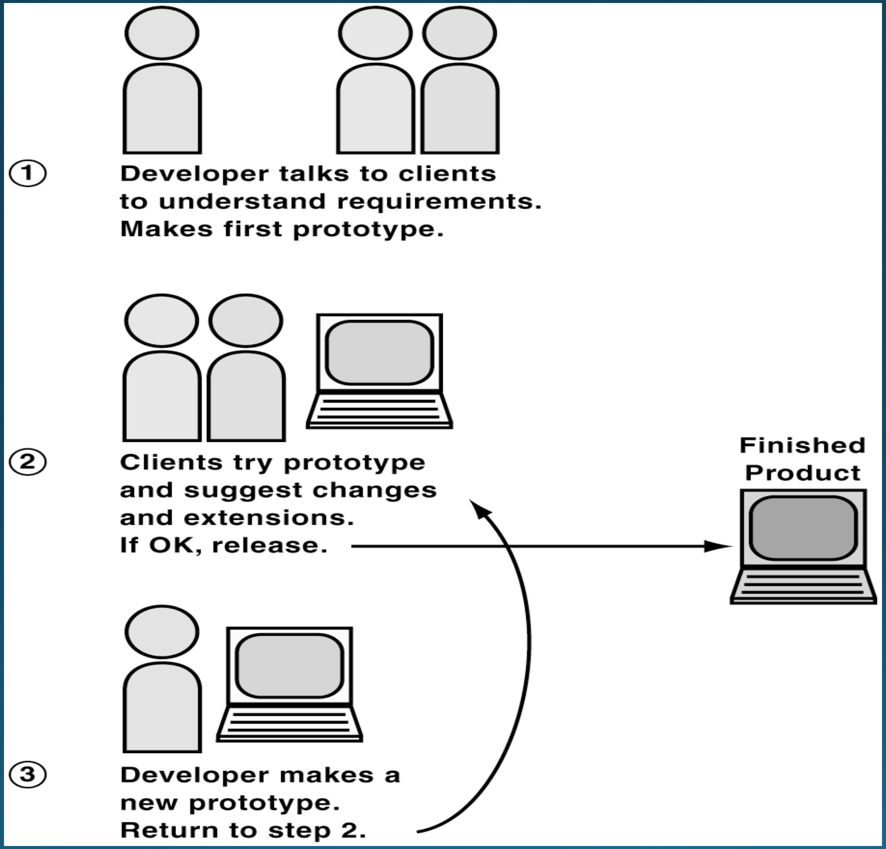
Joint Application Design

- JAD (Joint Application Design) - method to collect app requirements early on in iterative fashion with end users
  - Came from exploration in DSDM (Dynamic Systems Development Method) but is often applied wholesale to the whole project

DSDM Visualized



Sad JAD



① Developer talks to clients to understand requirements. Makes first prototype.
② Clients try prototype and suggest changes and extensions. If OK, release.
③ Developer makes a new prototype. Return to step 2.
Finished Product

Agile Manifesto

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

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

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

12 Principles of Agile Manifesto

1. Customer Satisfaction by rapid delivery of software
2. Welcoming changing requirements even late in development
3. Working software is delivered frequently (weeks not months)
4. Working software is the principal measure of progress
5. Sustainable development, able to maintain a constant pace
6. Close, daily cooperation between business people and developers
7. Face-to-face conversation is the best form of communication (co-location)
8. Projects are built around motivated individuals, who should be trusted
9. Continuous attention to technical excellence and good design
10. Simplicity - the art of maximizing the amount of work not done - is essential
11. Self-organizing teams
12. Regular adaptation to changing circumstances

SCRUM!

Theory based on Rugby move!?

Scrum Details

- Scrum team size - 7 (really 7+/-2)
  - Enough diversity for roles not too big
- Three scrum roles:
  1. product owner
  2. scrum master
  3. team member

Product Owner

- Directs team to maximize ROI on project
  - Pick valuable work, avoid less valuable
  - Control development order
- Records requirements in form of "user stories"
  - As a <type of user>, I want to <do something>, so that <some value is created>

Product Owner Role

- Holds product vision
- Represent biz interest
- Represent customer interest
- Balance these (I added this)
- Owns product backlog and prioritizes it
- Answers and owns team questions

Scrum Master

- Kind of the sub-boss (but they claim not)
- Serve as coach, champion, facilitator, road-block remover, etc.
- Primary job is keep team efficient

Scrum Master Role

- Expert in all thing scrum
- Coach (and cheerleader)
- Impediment remover
- Facilitator
- Politician? (I added this)

Team Member

- Teams are self-organizing (people can switch roles, do many, etc.)
  - Team members get to pick work
  - Team members may not be homogenous and have specialities or they may be more interchangable
- Team members ~~do the work~~ get things done

Team Member Role

- Responsible for completing user stories to incrementally improve product
- Self-organizes to get work done
- Creates and owns estimates
- Owns the how to do the work decision

Product Backlog

- Cumulative ordered list of desired deliverables of the product
  - Features, fixes, docs, etc.
- Should be ordered in priority
- Top items tend to be detailed (ex. Add underlining button) and and bottom items are more amorphous (ex. Make drawing routine faster)

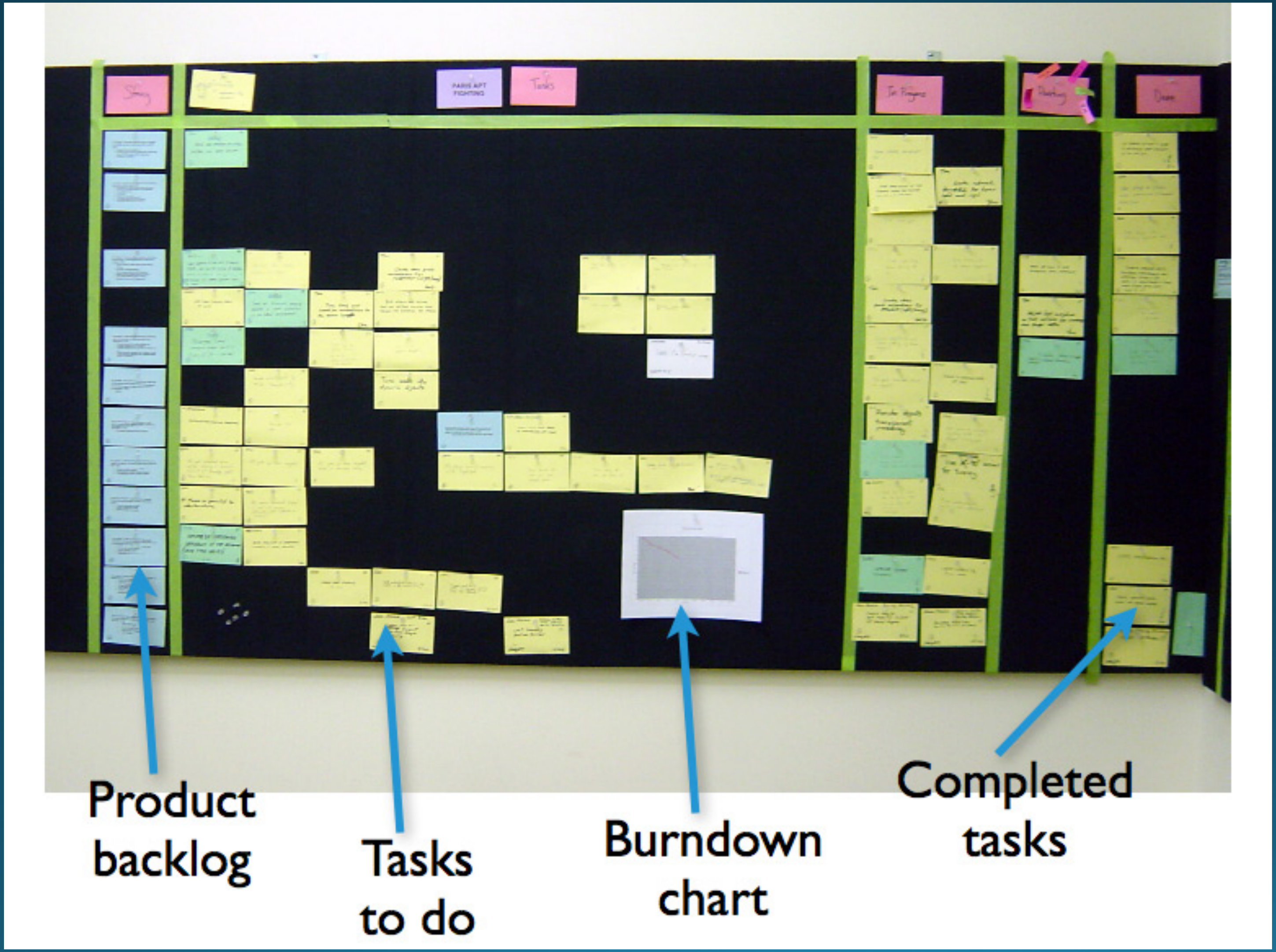
Prioritization of ideas

- There is nothing new here
- We try to formalize the process, but this trade-off effort is part of our famous triangle of features, time, cost
- Our choice of what makes sense is often quite reactive and inconsistent
- Many attempts are made to bring some order to this VERY important decision

MoSCoW Method* Not in Agile or Is It?

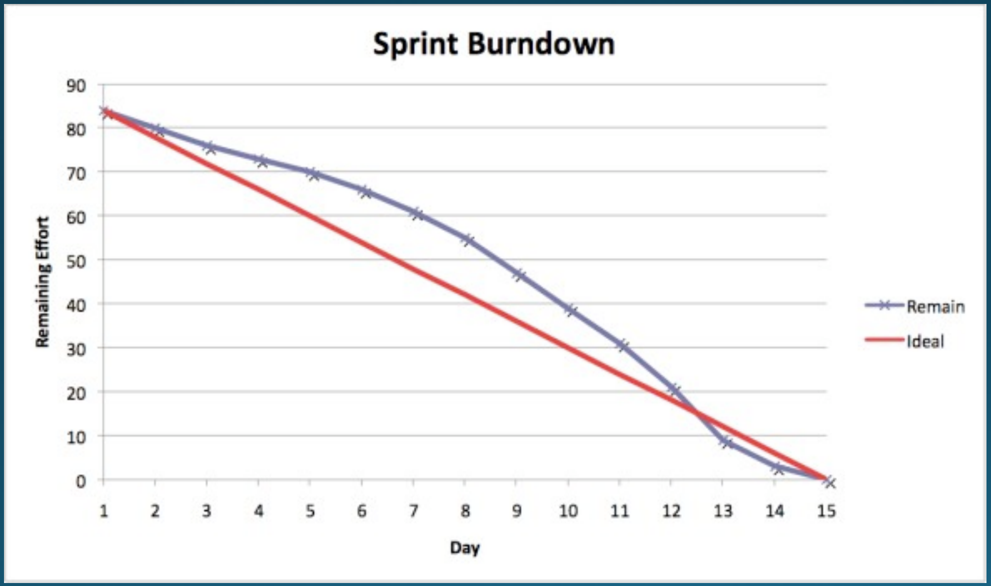
- M - Must: Requirement that must be in final solution
  - MUST - Minimum Usable SubseT
- S - Should: High priority item that should be in final solution
- C - Could: Desirable item but not mandatory
- W - Won't: Item agreed not to be done

Post-It Time!?

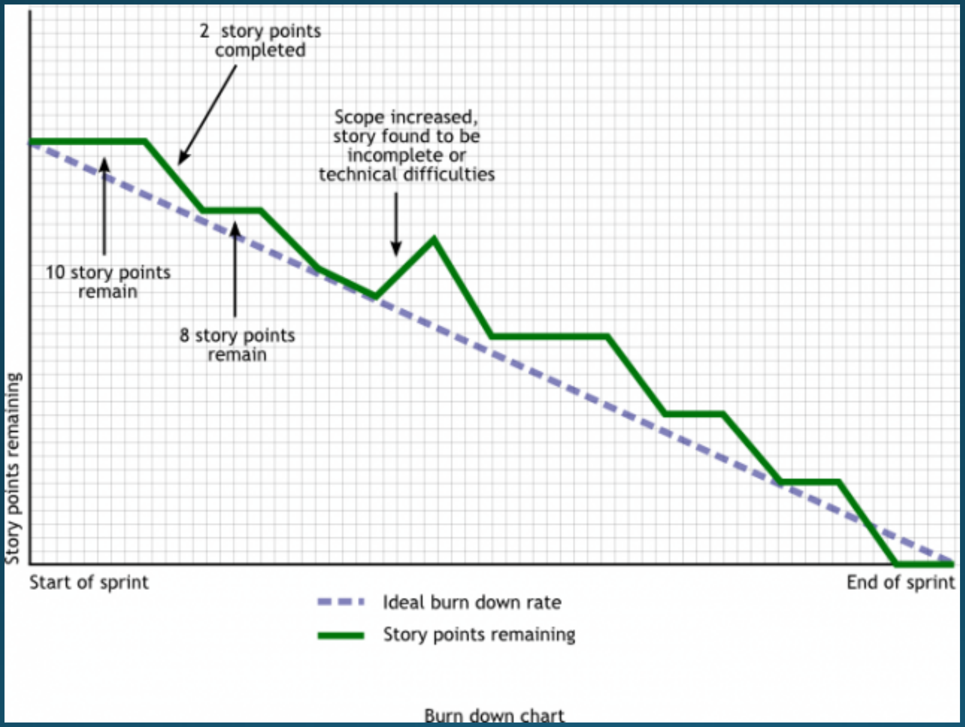

Product backlog
Tasks to do
Burndown chart
Completed tasks

Burn Down Chart

- A burn down chart plots work left to do versus time.

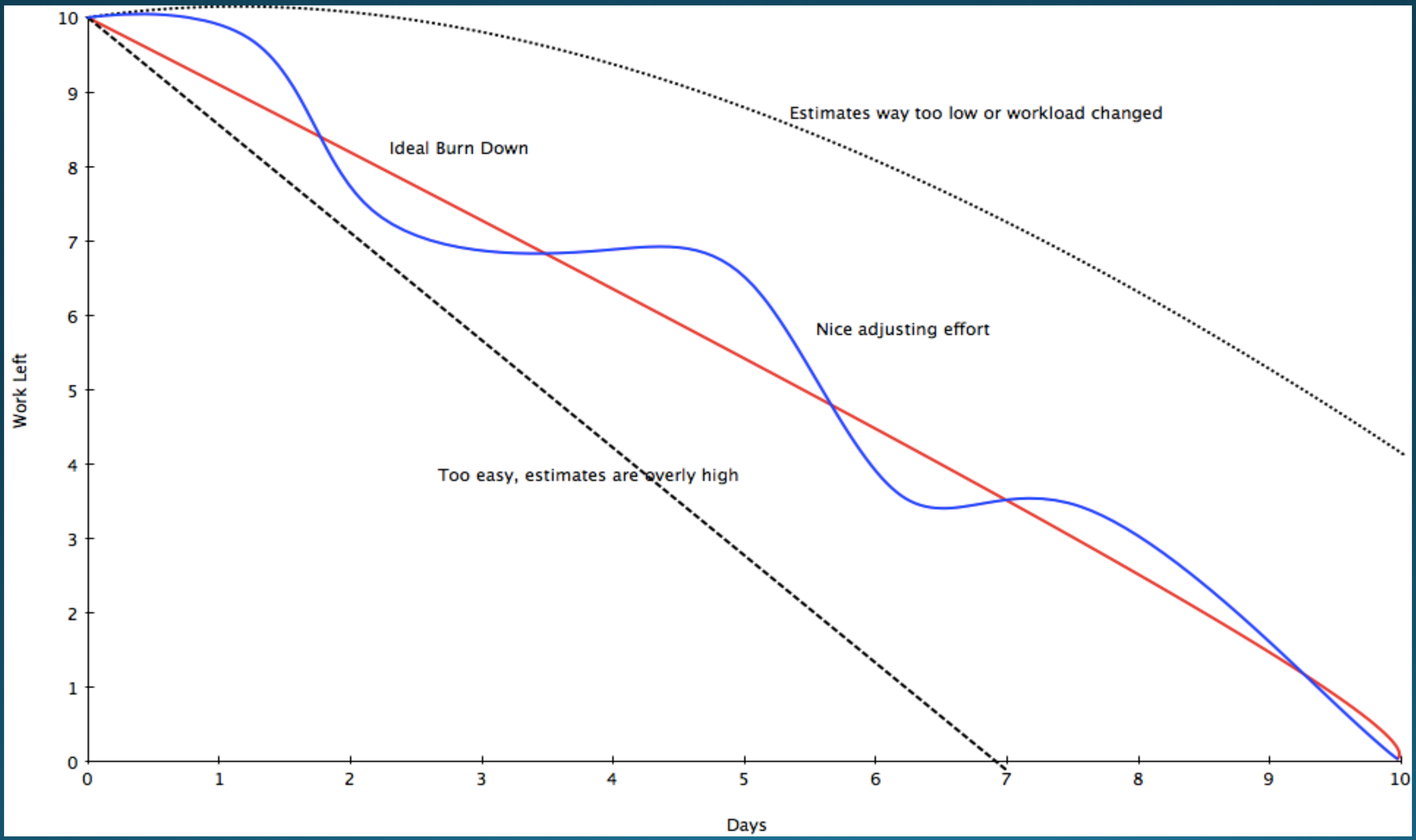


Note - Burn Down Chart

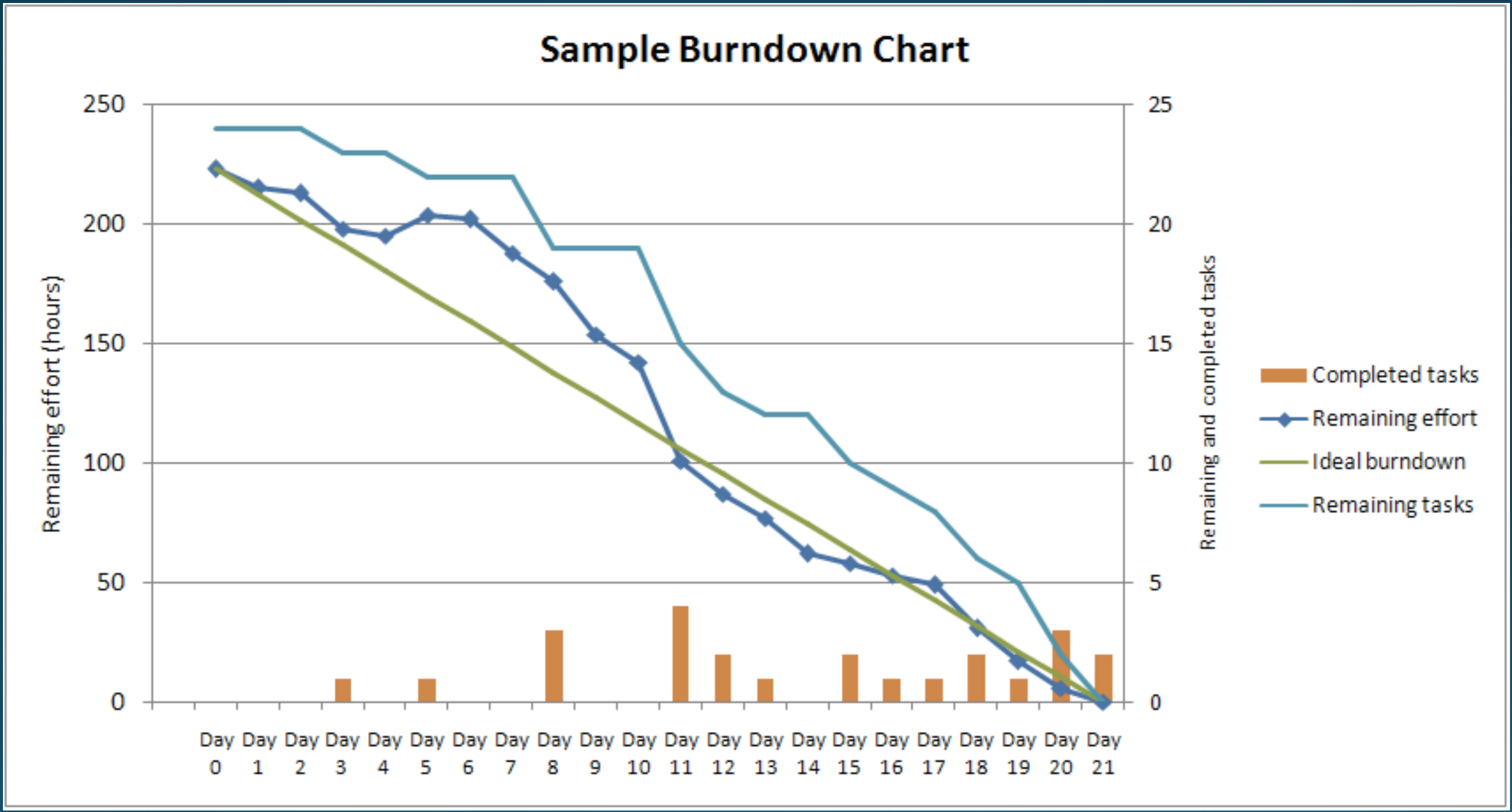


- Q: Are we getting closer?
- Helps keep you going running "1yard plays"
- Note all the examples show charts like this … slightly right ward and then magically leftward at the sprint end…hmmmm

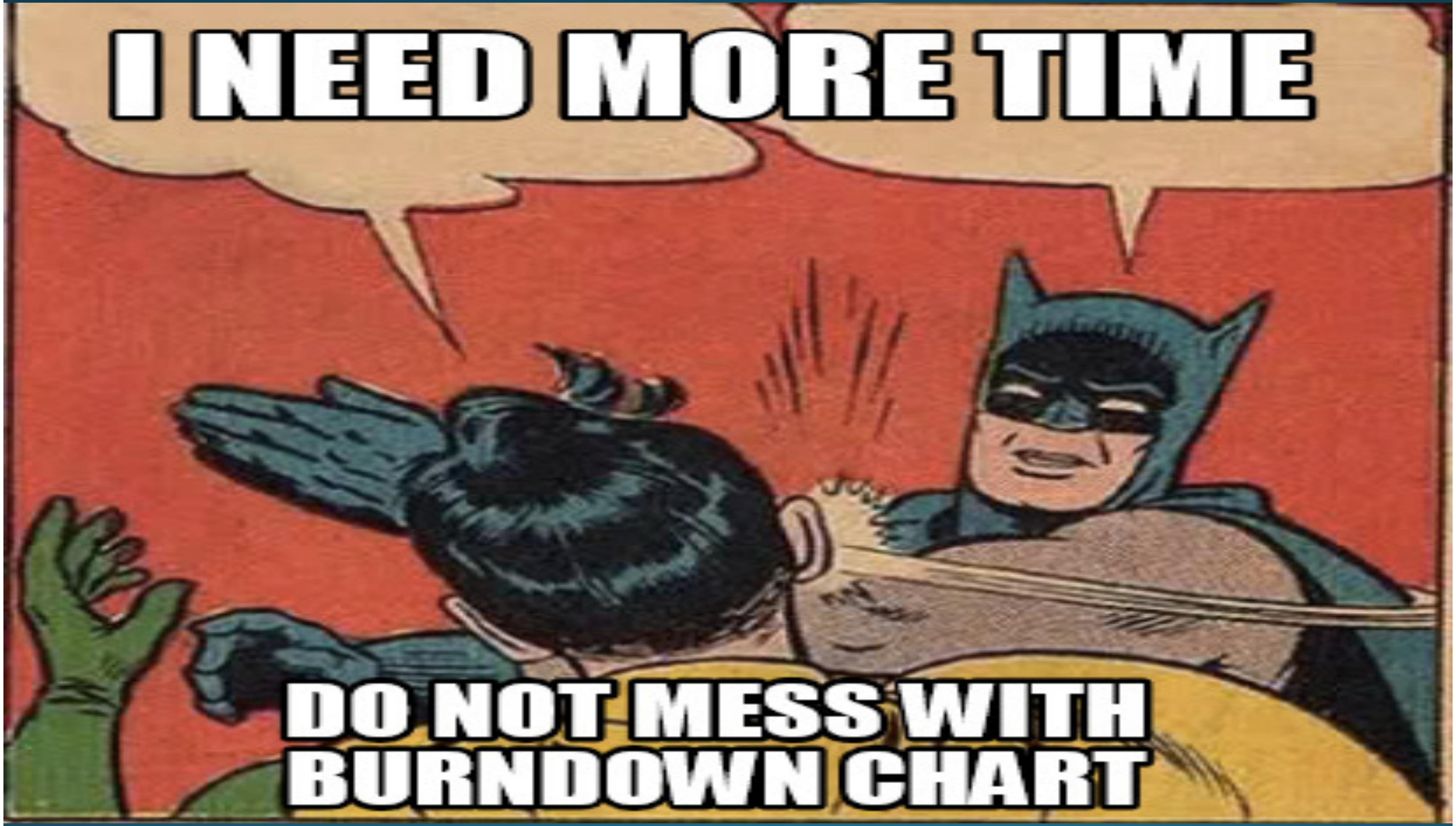
Burn Down Ranges



Burning More Complexity In



What's the point unless you are underserved by your BD chart?


I NEED MORE TIME
DO NOT MESS WITH BURNDOWN CHART

Iteration Design

- Don't make them too long
  - 1 week, 2 week, 3 week at outside
  - Short length helps you deal with change so you can switch focus if something comes up the next iteration
  - Progress of completing an iteration keeps us motivated
- Keep the iteration content in balance
  - Not just bugs, but features too for example
  - Variety keeps the interest up

The Need to Account for Velocity

- Given X days, some value X-O Is how much is actually spent on work
  - O for overhead maybe? Whatever the point there is some sense that you can't have perfect days
- Maximum productivity as we know is a myth

- Novice Programmers schedule utopian days
- Experienced Programmers schedule in real world days

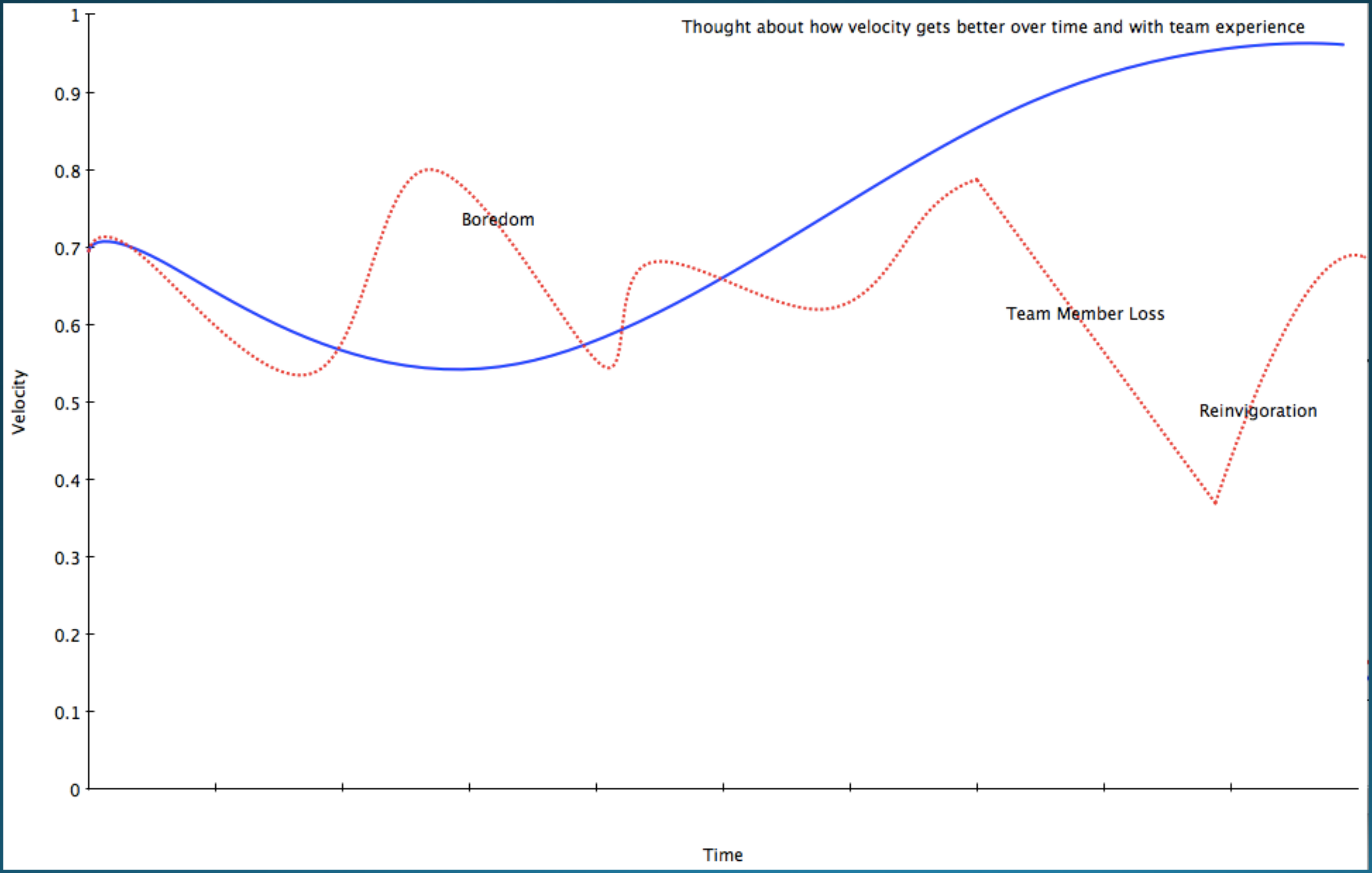
Velocity

- Some factor that indicates what % of time is actually working during a sprint

- Commonly 0.7 is used as a starting velocity value and we adjust it as we move along to account for reality

- Example:
  - January -> 31 days -> 20 work days actually 19 days (MLK)
  - Use 0.7 velocity and you get 14 work days to schedule!
  - The extra 5 days we can consider slack

Velocity Changes Over Time



Accepting Reality

- Ultimate Goal: Always tell customers and other stake holders what you know actually can be done as opposed to just what they want to hear
- Rule: Balance what customer wants ≈ what can be done
  - Degree of equality is related to customer reasonableness and your ability to be brutally honest with the customer
- If rule is out of balance then either the customer is going to be unhappy, the dev unhappy/over worked or both

SCOTTY SAYS ENGINEERS
SHOULD UNDERPROMISE
AND ALWAYS DELIVER
imgflip.com

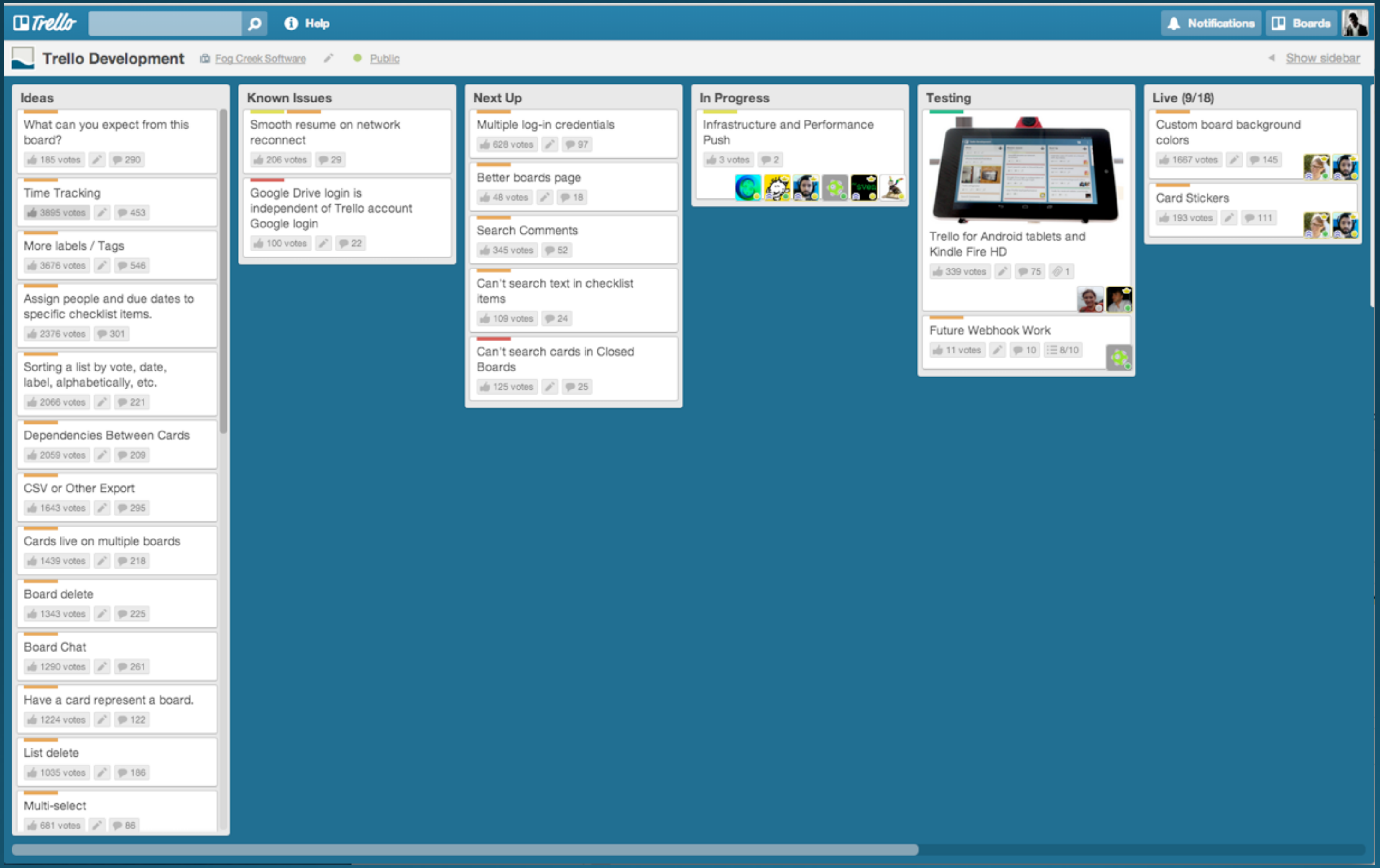
User Stories and Tasks

- Users stories != Tasks typically
- A User Story might be made of 1 to n tasks
- Each task should be a piece of work for a single developer and come with an estimate
- Tasks should be no smaller than .5 (4hrs) and 5 days (20hrs) assuming no velocity built in

User Story & Tasks Example

- User Story – Estimate 9
- Story is composed of 4 tasks
  - Task 1 – Estimate 2
  - Task 2 – Estimate 5
  - Task 3 – Estimate 3
  - Task 4 – Estimate 1
- Task estimate = 11 != 9 the original story estimate
- Now we should go back and adjust our user story to match the cumulative task estimates

Tooling Example



Trello Development · Fog Creek Software · Public · Show sidebar

Ideas

- What can you expect from this board? — 185 votes · 290
- Time Tracking — 3895 votes · 453
- More labels / Tags — 3676 votes · 546
- Assign people and due dates to specific checklist items. — 2376 votes · 301
- Sorting a list by vote, date, label, alphabetically, etc. — 2066 votes · 221
- Dependencies Between Cards — 2059 votes · 209
- CSV or Other Export — 1643 votes · 295
- Cards live on multiple boards — 1439 votes · 218
- Board delete — 1343 votes · 225
- Board Chat — 1290 votes · 261
- Have a card represent a board. — 1224 votes · 122
- List delete — 1035 votes · 186
- Multi-select — 681 votes · 86

Known Issues

- Smooth resume on network reconnect — 206 votes · 29
- Google Drive login is independent of Trello account Google login — 100 votes · 22

Next Up

- Multiple log-in credentials — 628 votes · 97
- Better boards page — 48 votes · 18
- Search Comments — 345 votes · 52
- Can't search text in checklist items — 109 votes · 24
- Can't search cards in Closed Boards — 125 votes · 25

In Progress

- Infrastructure and Performance Push — 3 votes · 2

Testing

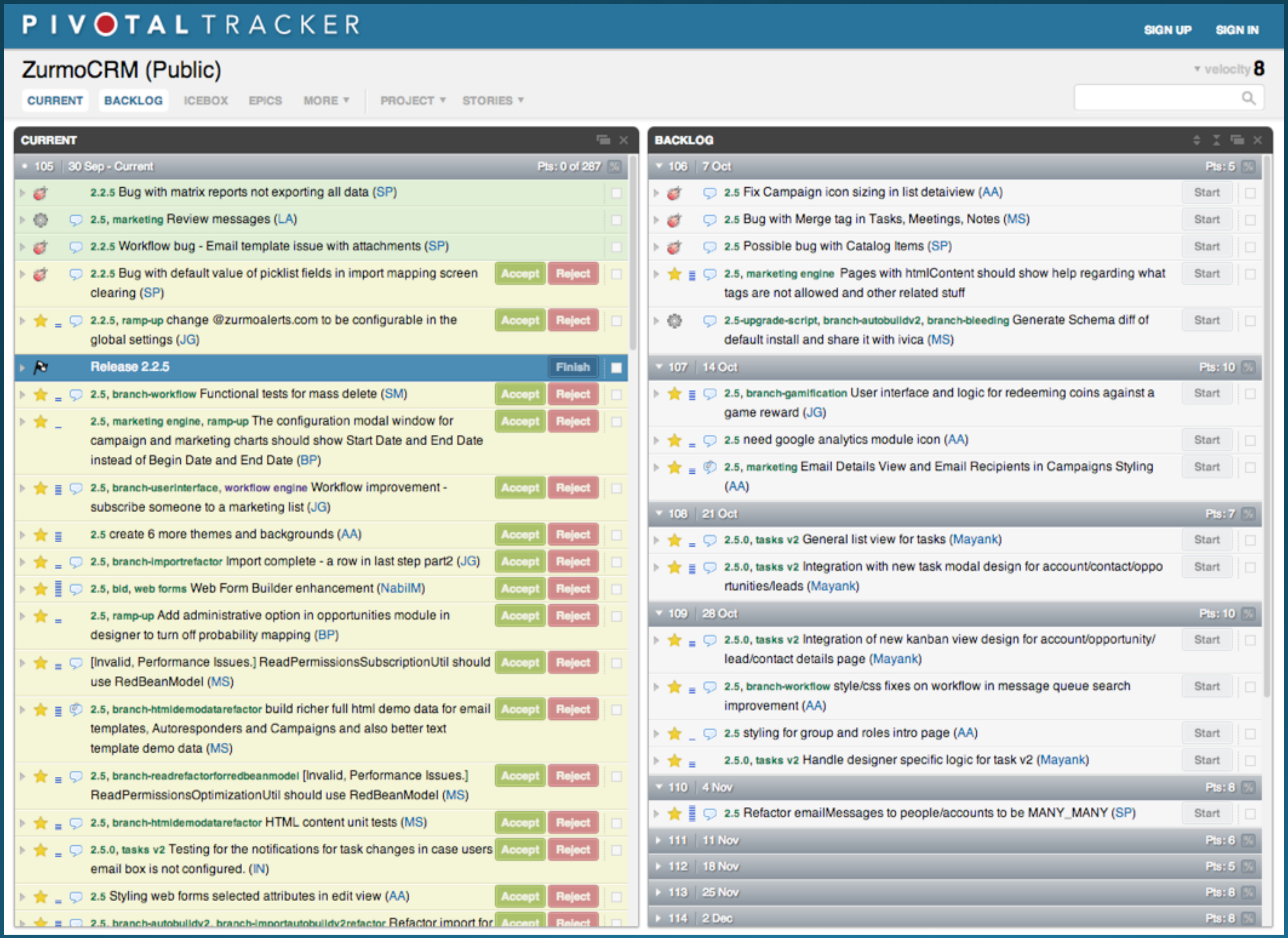
- Trello for Android tablets and Kindle Fire HD — 339 votes · 75 · 1
- Future Webhook Work — 11 votes · 10 · 8/10

Live (9/18)

- Custom board background colors — 1667 votes · 145
- Card Stickers — 193 votes · 111

http://www.trello.com

Tooling Example



ZurmoCRM (Public)

CURRENT

105 | 30 Sep - Current — Pts: 0 of 287

- 2.2.5 Bug with matrix reports not exporting all data (SP)
- 2.5, marketing Review messages (LA)
- 2.2.5 Workflow bug - Email template issue with attachments (SP)
- 2.2.5 Bug with default value of picklist fields in import mapping screen clearing (SP) — Accept Reject
- 2.2.5, ramp-up change @zurmoalerts.com to be configurable in the global settings (JG) — Accept Reject
- Release 2.2.5 — Finish
- 2.5, branch-workflow Functional tests for mass delete (SM) — Accept Reject
- 2.5, marketing engine, ramp-up The configuration modal window for campaign and marketing charts should show Start Date and End Date instead of Begin Date and End Date (BP) — Accept Reject
- 2.5, branch-userinterface, workflow engine Workflow improvement - subscribe someone to a marketing list (JG) — Accept Reject
- 2.5 create 6 more themes and backgrounds (AA) — Accept Reject
- 2.5, branch-importrefactor Import complete - a row in last step part2 (JG) — Accept Reject
- 2.5, bid, web forms Web Form Builder enhancement (NabilM) — Accept Reject
- 2.5, ramp-up Add administrative option in opportunities module in designer to turn off probability mapping (BP) — Accept Reject
- [Invalid, Performance Issues.] ReadPermissionsSubscriptionUtil should use RedBeanModel (MS) — Accept Reject
- 2.5, branch-htmldemodatarefactor build richer full html demo data for email templates, Autoresponders and Campaigns and also better text template demo data (MS) — Accept Reject
- 2.5, branch-readrefactorforredbeanmodel [Invalid, Performance Issues.] ReadPermissionsOptimizationUtil should use RedBeanModel (MS) — Accept Reject
- 2.5, branch-htmldemodatarefactor HTML content unit tests (MS) — Accept Reject
- 2.5.0, tasks v2 Testing for the notifications for task changes in case users email box is not configured. (IN) — Accept Reject
- 2.5 Styling web forms selected attributes in edit view (AA) — Accept Reject
- 2.5, branch-autobuildv2, branch-importautobuildv2refactor Refactor import for — Accept Reject

BACKLOG

106 | 7 Oct — Pts: 5

- 2.5 Fix Campaign icon sizing in list detaiview (AA) — Start
- 2.5 Bug with Merge tag in Tasks, Meetings, Notes (MS) — Start
- 2.5 Possible bug with Catalog Items (SP) — Start
- 2.5, marketing engine Pages with htmlContent should show help regarding what tags are not allowed and other related stuff — Start
- 2.5-upgrade-script, branch-autobuildv2, branch-bleeding Generate Schema diff of default install and share it with ivica (MS) — Start

107 | 14 Oct — Pts: 10

- 2.5, branch-gamification User interface and logic for redeeming coins against a game reward (JG) — Start
- 2.5 need google analytics module icon (AA) — Start
- 2.5, marketing Email Details View and Email Recipients in Campaigns Styling (AA) — Start

108 | 21 Oct — Pts: 7

- 2.5.0, tasks v2 General list view for tasks (Mayank) — Start
- 2.5.0, tasks v2 Integration with new task modal design for account/contact/opportunities/leads (Mayank) — Start

109 | 28 Oct — Pts: 10

- 2.5.0, tasks v2 Integration of new kanban view design for account/opportunity/lead/contact details page (Mayank) — Start
- 2.5, branch-workflow style/css fixes on workflow in message queue search improvement (AA) — Start
- 2.5 styling for group and roles intro page (AA) — Start
- 2.5.0, tasks v2 Handle designer specific logic for task v2 (Mayank) — Start

110 | 4 Nov — Pts: 8

- 2.5 Refactor emailMessages to people/accounts to be MANY_MANY (SP) — Start

111 | 11 Nov — Pts: 6

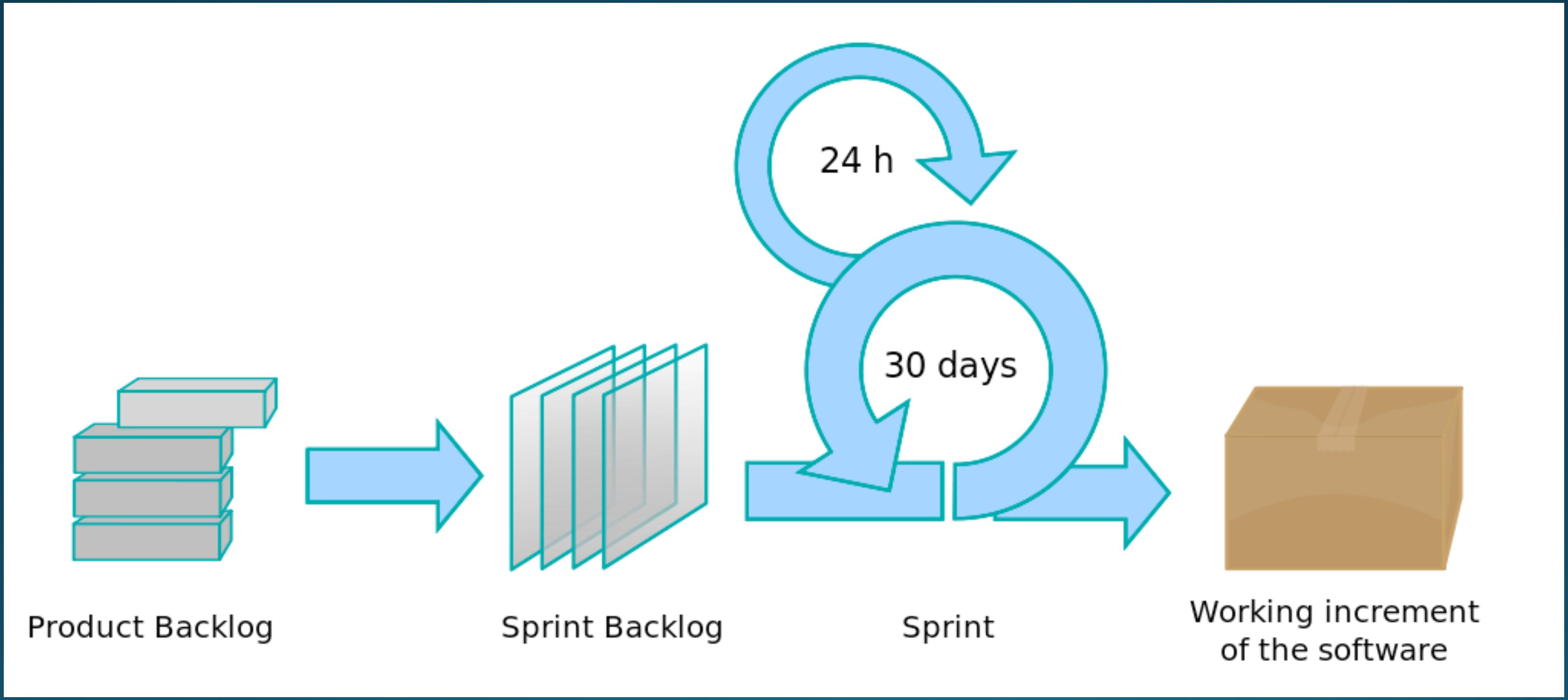
112 | 18 Nov — Pts: 5

113 | 25 Nov — Pts: 8

114 | 2 Dec — Pts: 8

http://www.pivotaltracker.com

SCRUM Visualized



Some Pot Shots

- Story points are like fake currency in games to me
  - I just spent X doughnuts
  - We just used X story points
- Intention is good
  - But it is somewhat window dressing on a cut and chew approach
- There have been a number of times I want to employ agile mechanisms but the nature of what needs to be done doesn't lend itself well even though the project itself is managed this way
  - Very often the tasks don't decompose well they are largeish but don't benefit well being broken into mini tasks. Ex: We are porting ~96 classes to a messed up way of OOP to a standard ES6 way. That is clearly not a single sprint effort but breaking the story cards into a single class at a time is nuts and just chunking it by sprint size is arbitrary. The person(s) doing this every report for ~2-5 weeks annoyingly get to say "working on the class refactor"
  - Conclusion: Some sort of religious adherence is not useful, you'll be made to force things and it limits how you might solve the problem. Remember: At the conclusion/ship time people don't buy your process they buy your result! The process has to help us get there and when it doesn't we should feel free to modify it or even dump parts of it

Some Other Thoughts

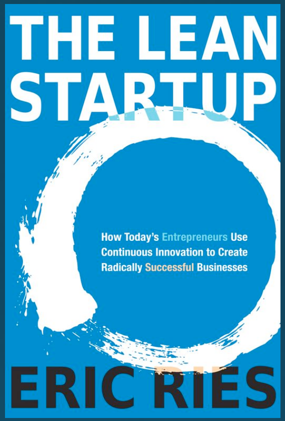
- Sprint length varies - 1 week, 2 weeks, 1 month
  - Too frequent overhead gets bad
  - Too infrequent - you're doing Waterfall even if you don't know it
  - Should it be static? Makes it simple but sometimes you might be making things arbitrary or awkward
- Product Owner is a key role, have a bad one and you're doomed.
- Q: What does a good product owner look like? Waterfall wise are they different?

Lean is the New Agile?

- Enter the whole Lean team with their modifications and applications

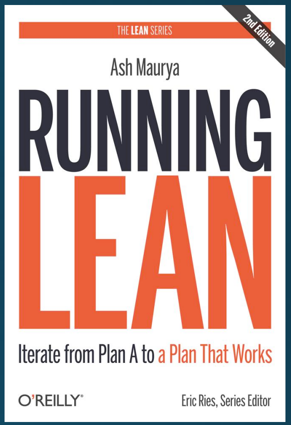
- Some really good ideas but as always watch the kool-aid

- Original for cars maybe it works for SE maybe also? Hmm…more borrowing!


THE LEAN STARTUP

How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses

ERIC RIES


THE LEAN SERIES

2nd Edition

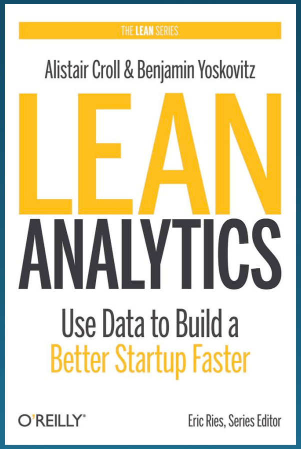
Ash Maurya

RUNNING LEAN

Iterate from Plan A to a Plan That Works

O'REILLY

Eric Ries, Series Editor


THE LEAN SERIES

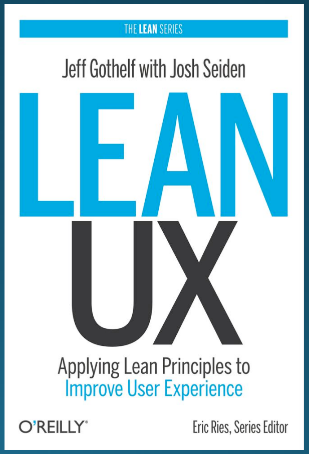
Alistair Croll & Benjamin Yoskovitz

LEAN ANALYTICS

Use Data to Build a Better Startup Faster

O'REILLY

Eric Ries, Series Editor


THE LEAN SERIES

Jeff Gothelf with Josh Seiden

LEAN UX

Applying Lean Principles to Improve User Experience

O'REILLY

Eric Ries, Series Editor

Origins of Lean

- Production practice that considers the expenditure of resources for any goal other than the creation of value for the end customer to be wasteful, and thus a target for elimination.
  - “preserving value with less work”
- This idea is being adopted in the Web/Start-up World

Seven Wastes

1. Transportation
2. Inventory
3. Motion
4. Waiting
5. Over-processing
6. Over-production
7. Defects

http://en.wikipedia.org/wiki/Lean_manufacturing

Do they apply?

- Transportation - the movement of code around various environments?
- Inventory - backlog of undeployed code
- Motion - harsh sprints, long sessions, etc.
- Waiting - work blocking, team inefficiency
- Over-production - algorithm polishing, corner case over focus
- Defects - Bugs

TPS SE?

- Continuous Improvement
  - Check: Data driven, Big Data, Metrics, Analytics, etc.
  - Check: CI (Continuous Integration), Agile, Sprints, etc.
- Respect for People
  - Check: Peopleware ideas

Quote #1

*"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including **all** the interfaces to **people**, to **machines**, and to **other software systems**. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."*

Quote #2

"...the most important function that the software builder performs for the client is the **iterative extraction and refinement of the product requirements**. For the truth is, the **client does not know what he wants**. The client usually **does not know what questions must be answered**, and he has almost **never thought of the problem in the detail necessary for specification**."

Quote #3

"Much of present-day software-acquisition procedure rests upon the assumption that one can specify a satisfactory system in advance, get bids for its construction, have it built, and install it. I think this assumption is fundamentally wrong, and that many software-acquisition problems spring from that fallacy. Hence, they cannot be fixed without fundamental revision--revision that provides for iterative development and specification of prototypes and products."

Quote #4

“Grow, don’t build, software.”

Nope, not Lean really surprise, all said by

Fred Brooks, age 83

"Mr. Mythical Man Month, Waterfall, etc."

Seems like we keep visiting the same ideas with new names

Lean Premise

- Three rough steps:

1. Document your Plan A
2. Identify the riskiest part of the plan
3. Systematically test the plan

All the while upload a strong vision with facts not faith

It’s so simple!

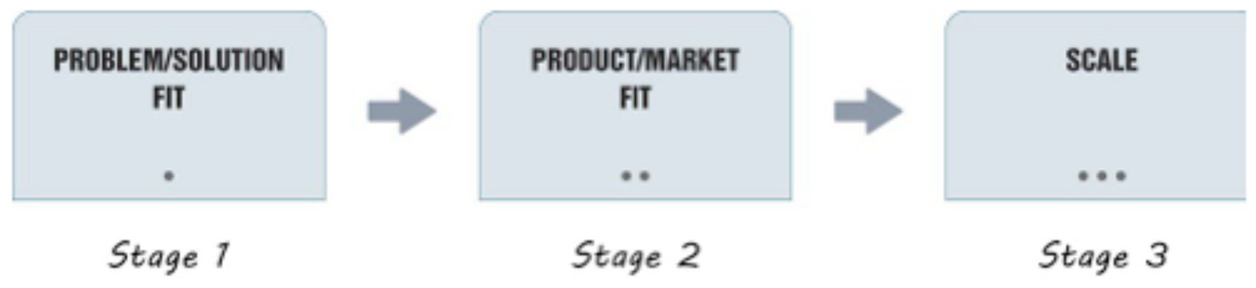


Figure 1-3. Three stages of a startup

Err…not so sure

Step 1

- Q: Do we have a problem worth solving?
  - *Tip: Watch out for confirmation bias here!*
    - Q: Is this something customers want? (demand)
    - Q: Will they pay for it? (viability)
    - Q: Can it be solved*? (feasibility)

Solidifying the Step

- Write down initial vision and share it!
  - Sounds like reqs, biz plan, etc.
  - Problem: Too long to write, nobody reads
  - Solution: Refined pitch, lean canvas, etc.

Lean Canvas

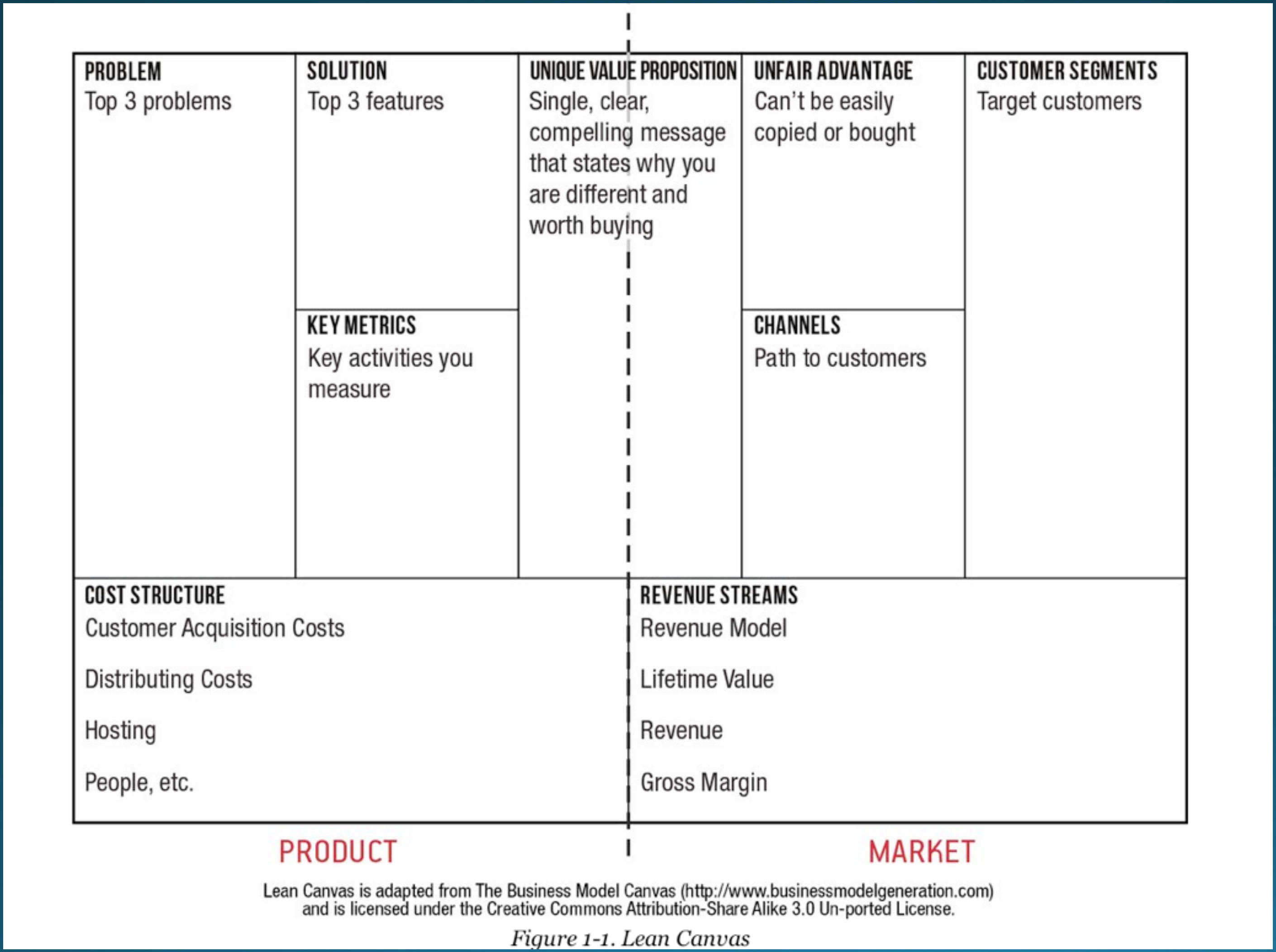
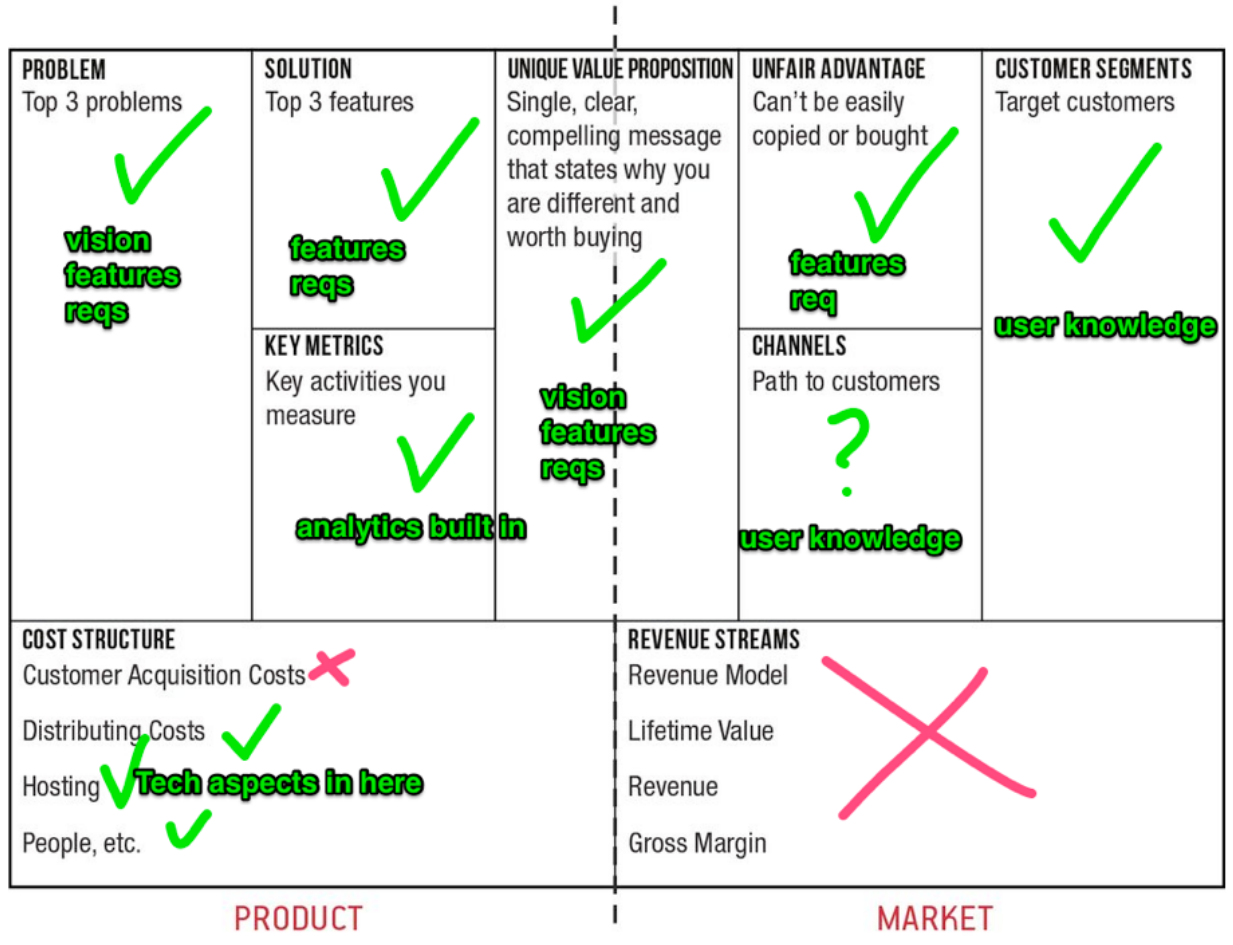

PROBLEM
Top 3 problems
SOLUTION
Top 3 features
KEY METRICS
Key activities you measure
UNIQUE VALUE PROPOSITION
Single, clear, compelling message that states why you are different and worth buying
UNFAIR ADVANTAGE
Can't be easily copied or bought
CHANNELS
Path to customers
CUSTOMER SEGMENTS
Target customers
COST STRUCTURE
Customer Acquisition Costs
Distributing Costs
Hosting
People, etc.
REVENUE STREAMS
Revenue Model
Lifetime Value
Revenue
Gross Margin
PRODUCT
MARKET
Lean Canvas is adapted from The Business Model Canvas (http://www.businessmodelgeneration.com) and is licensed under the Creative Commons Attribution-Share Alike 3.0 Un-ported License.

Figure 1-1. Lean Canvas

Not our problem

- Yes indeed portions of this are simply not our problem
  - We should worry if there aren't answers here. If no new job, next client, etc.
  - Reality is that many areas involve us and we can't abdicate them otherwise don't complain about being a code pusher for a bad idea

Involvement?



Lean Canvas is adapted from The Business Model Canvas (http://www.businessmodelgeneration.com) and is licensed under the Creative Commons Attribution-Share Alike 3.0 Un-ported License.

Figure 1-1. Lean Canvas

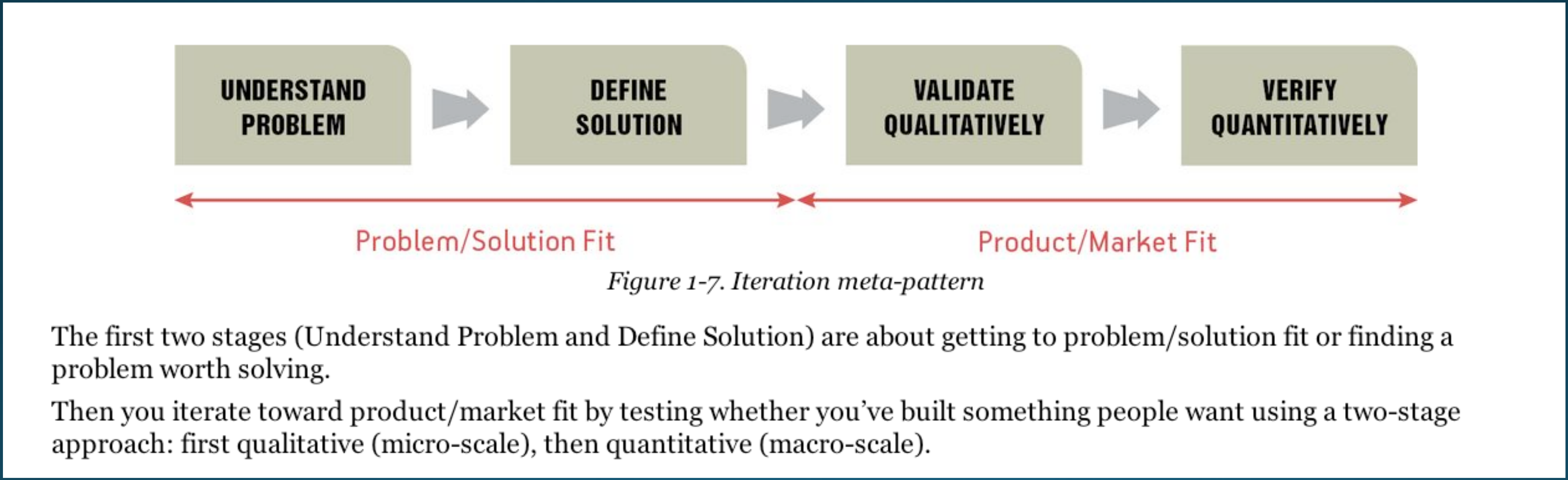
Step 1 Result

- From our better be tested ideas we derive the minimum viable product (MVP)
  - Sadly MVP determination isn't that easy
  - MVP may not be that small
  - MVP may create a code corner
  - Timing can kill all this

Step 2: Product / Market Fit

- We somehow have to come up with an understanding if this MVP is really a good idea before doing all this work
- Take the MVP out see if it works, talk to customers, ask questions and iterate the idea and “pivot” to other approaches as they become apparent

Step 1 and 2 Iterations



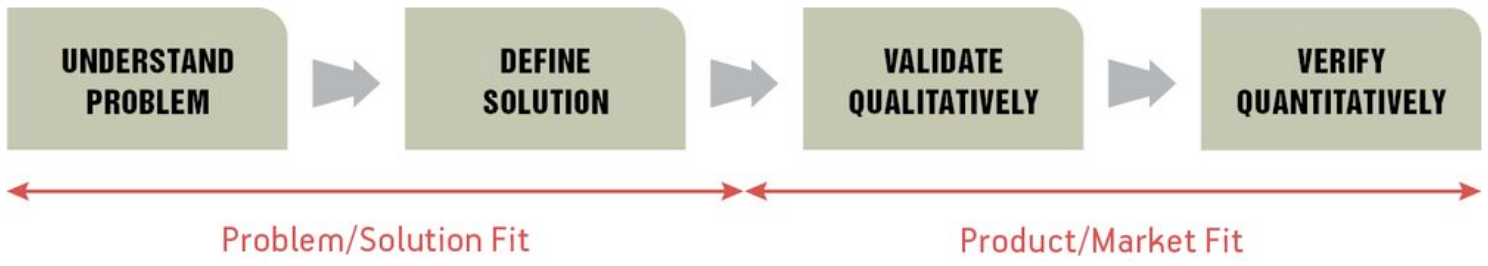


Figure 1-7. Iteration meta-pattern

The first two stages (Understand Problem and Define Solution) are about getting to problem/solution fit or finding a problem worth solving.

Then you iterate toward product/market fit by testing whether you've built something people want using a two-stage approach: first qualitative (micro-scale), then quantitative (macro-scale).

Lean Loop

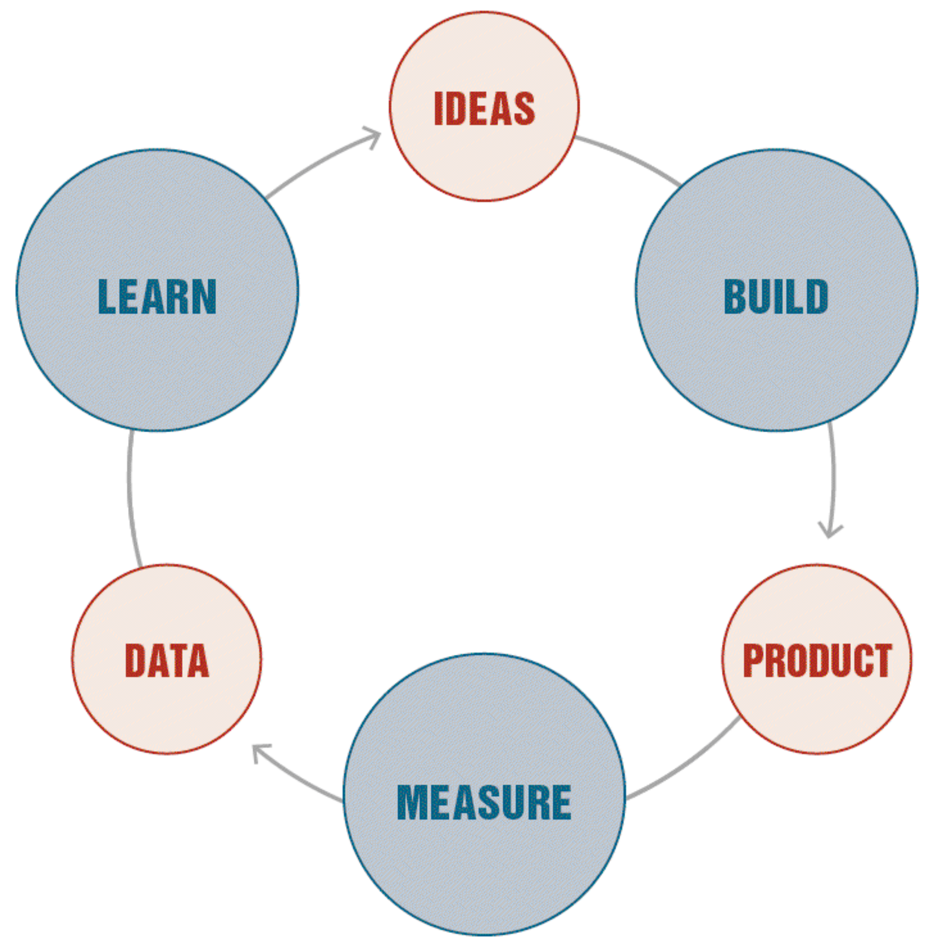
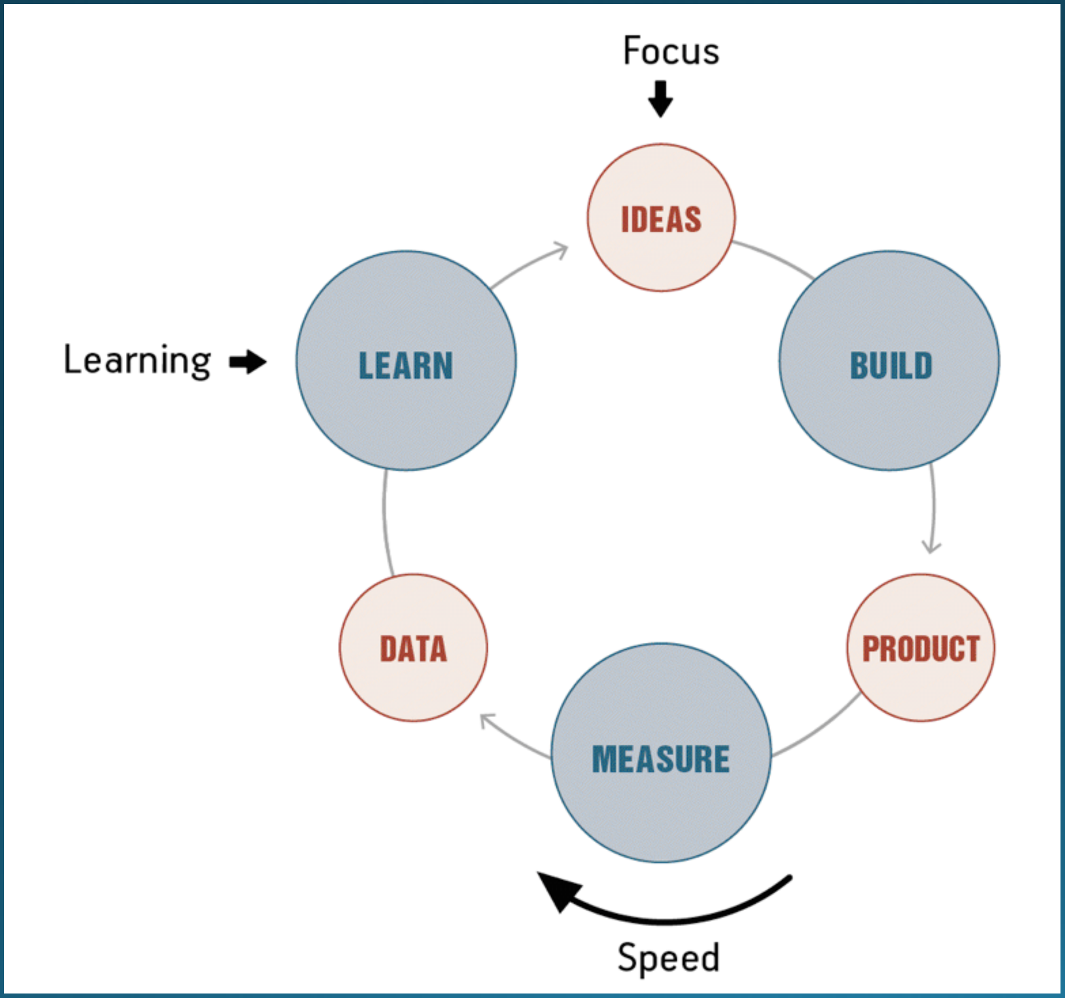


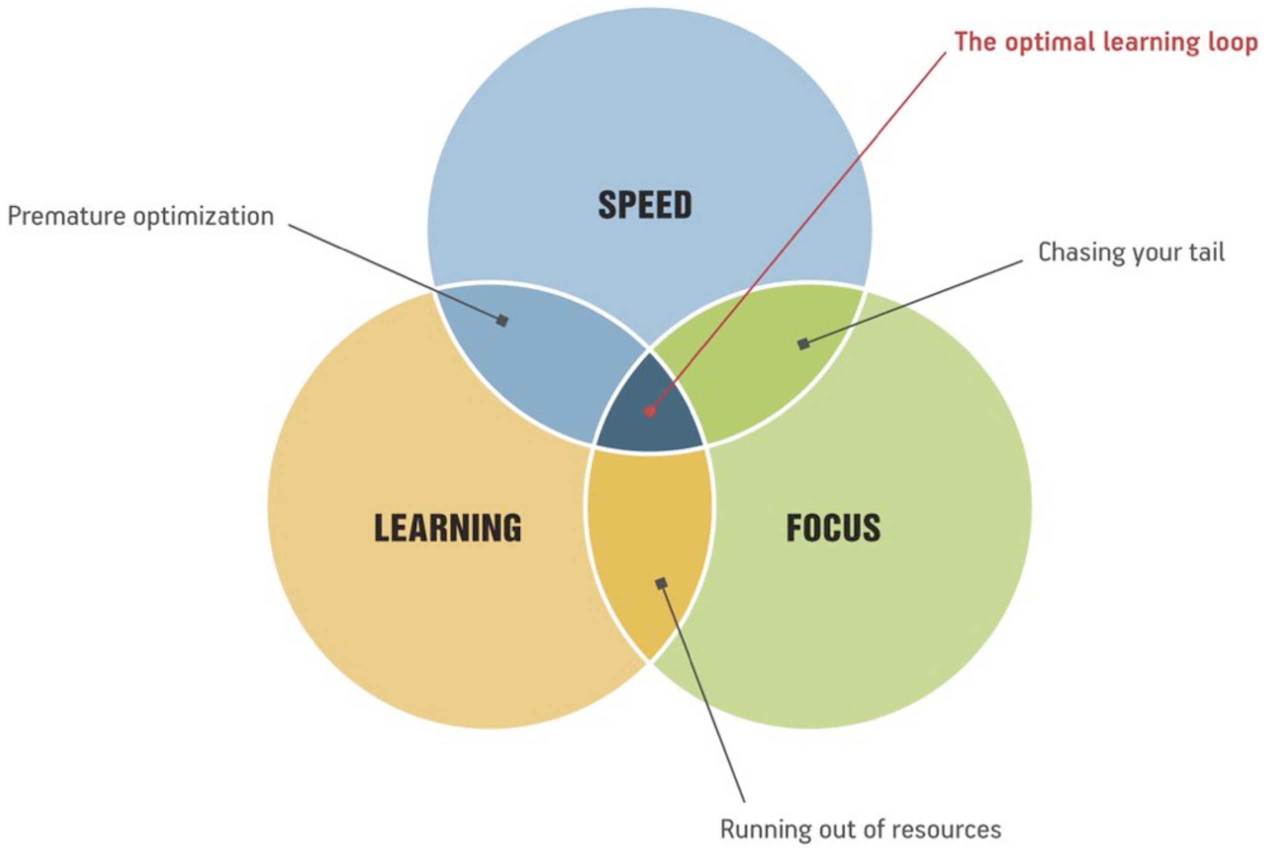
Figure 1-6. Build-Measure-Learn loop

Loop Thoughts

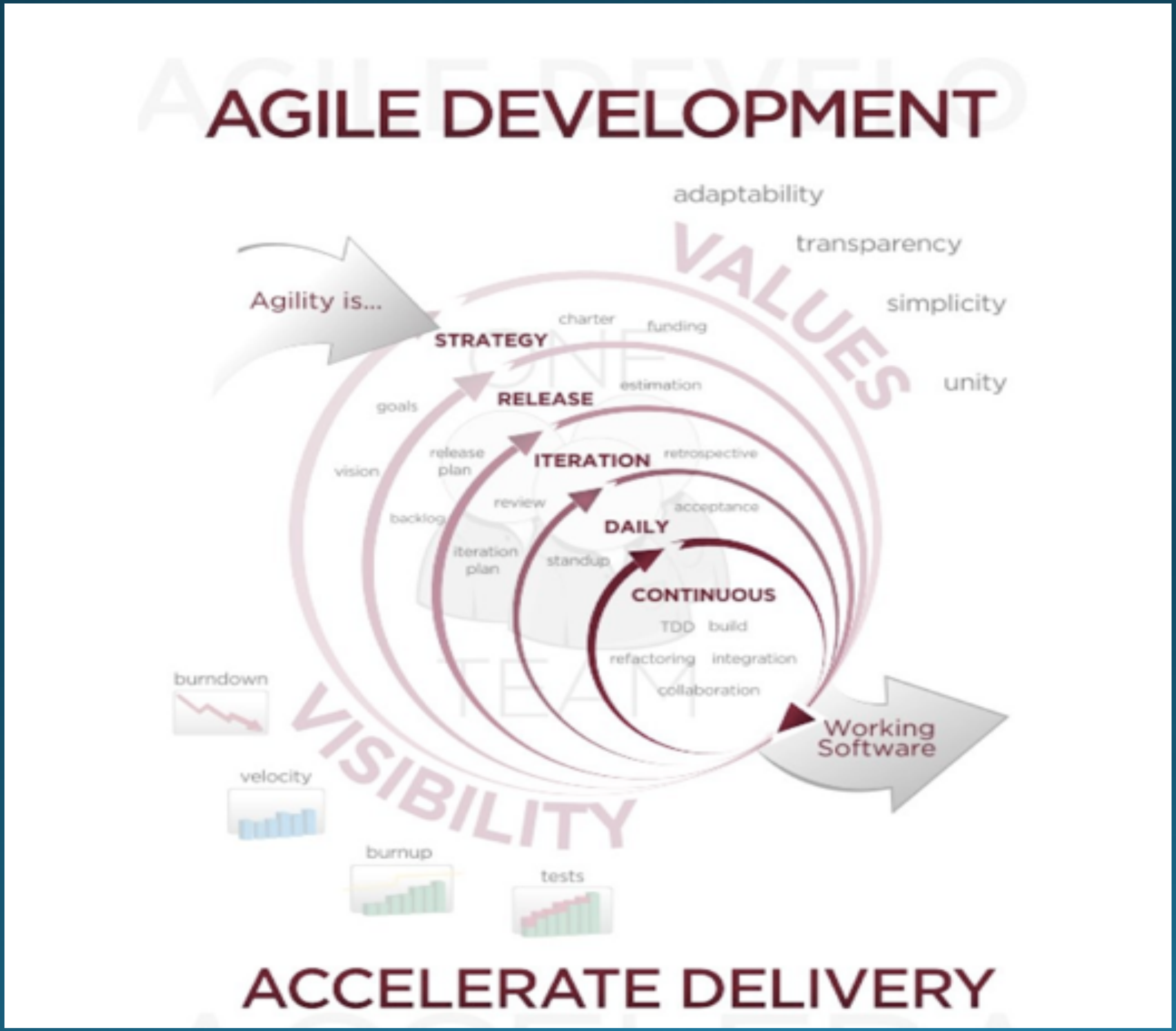


Optimal Lean Loop

Figure 5-1. Speed, learning, and focus



Agile Visualized Seems Lean-y



Continuous Deployment is Lean?

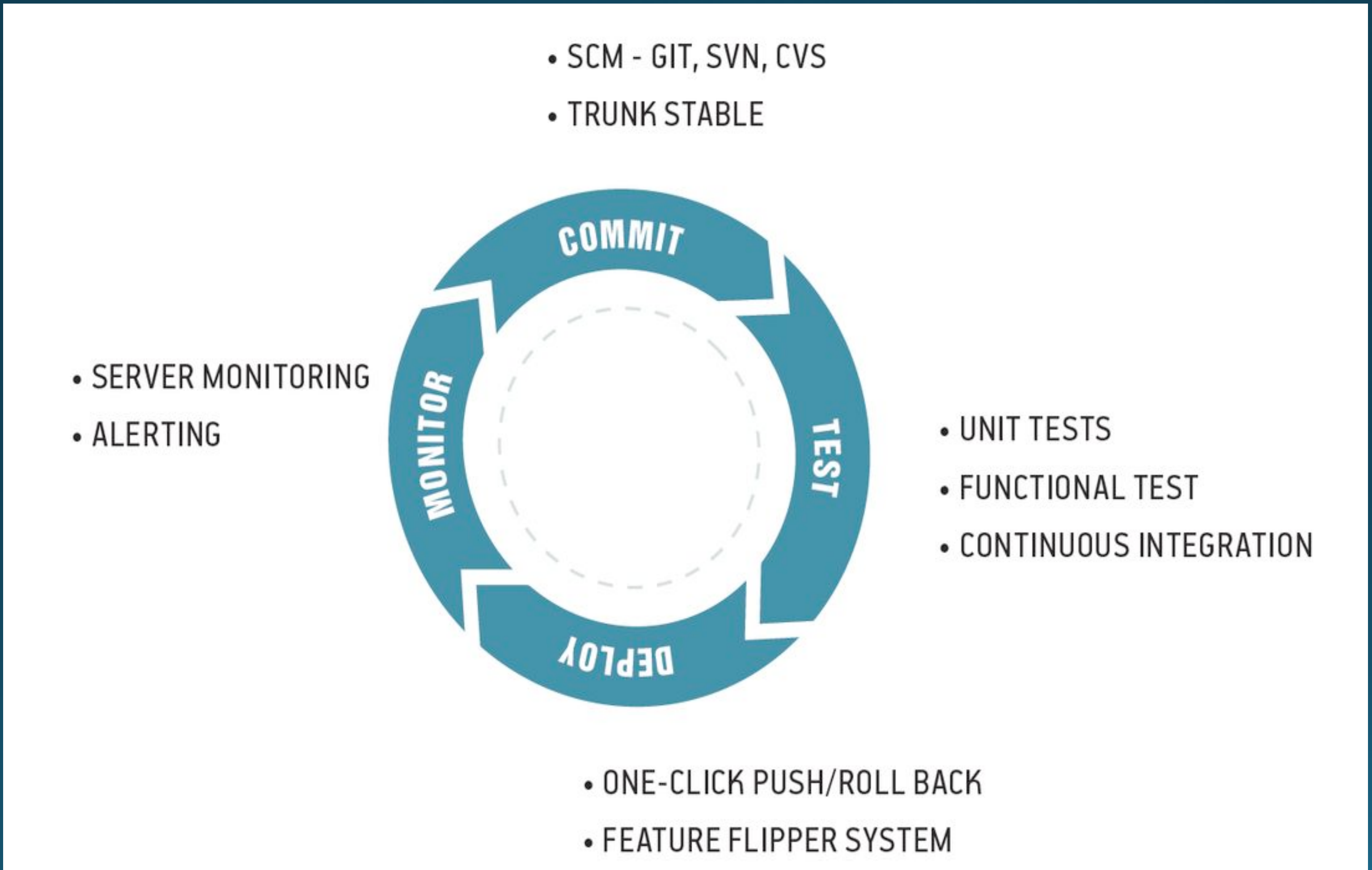


Figure A-1. Continuous deployment cycle

What to Build?

- Lean says less new feature

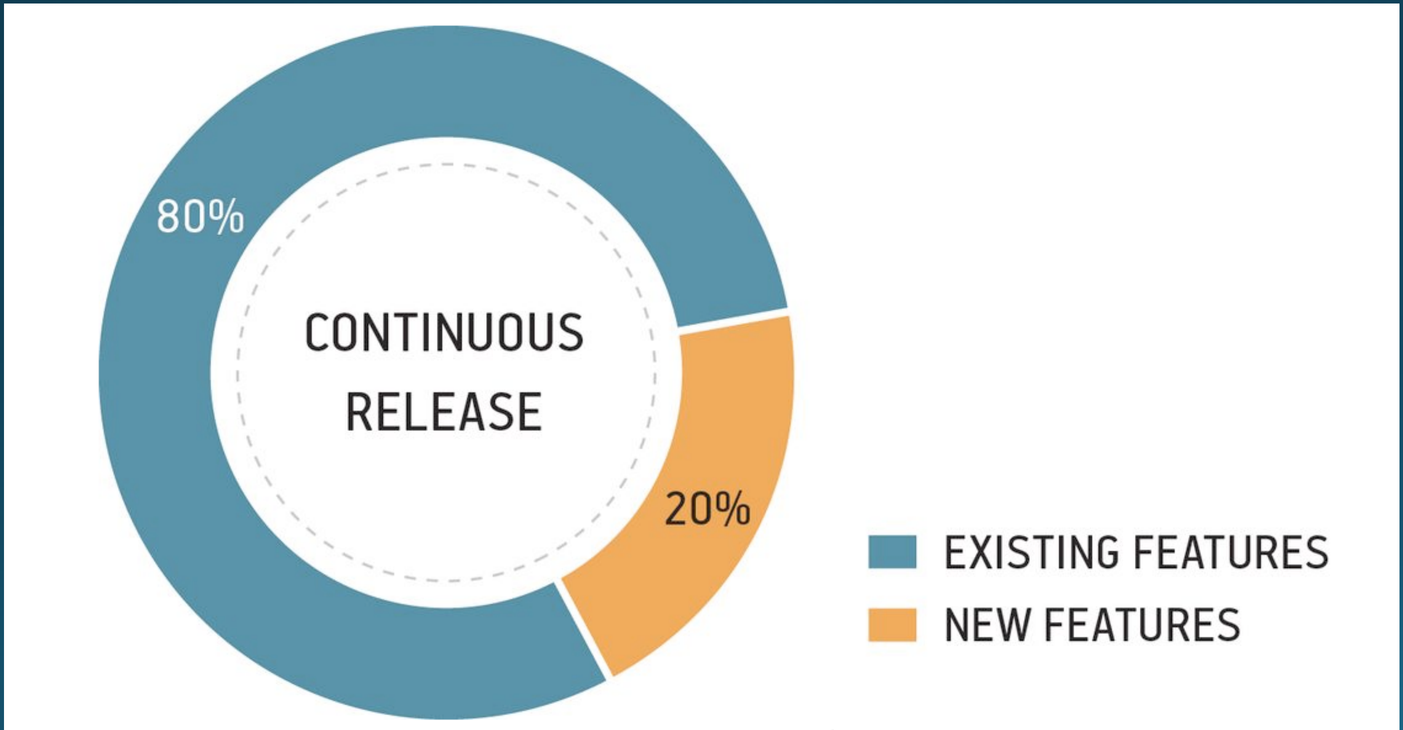
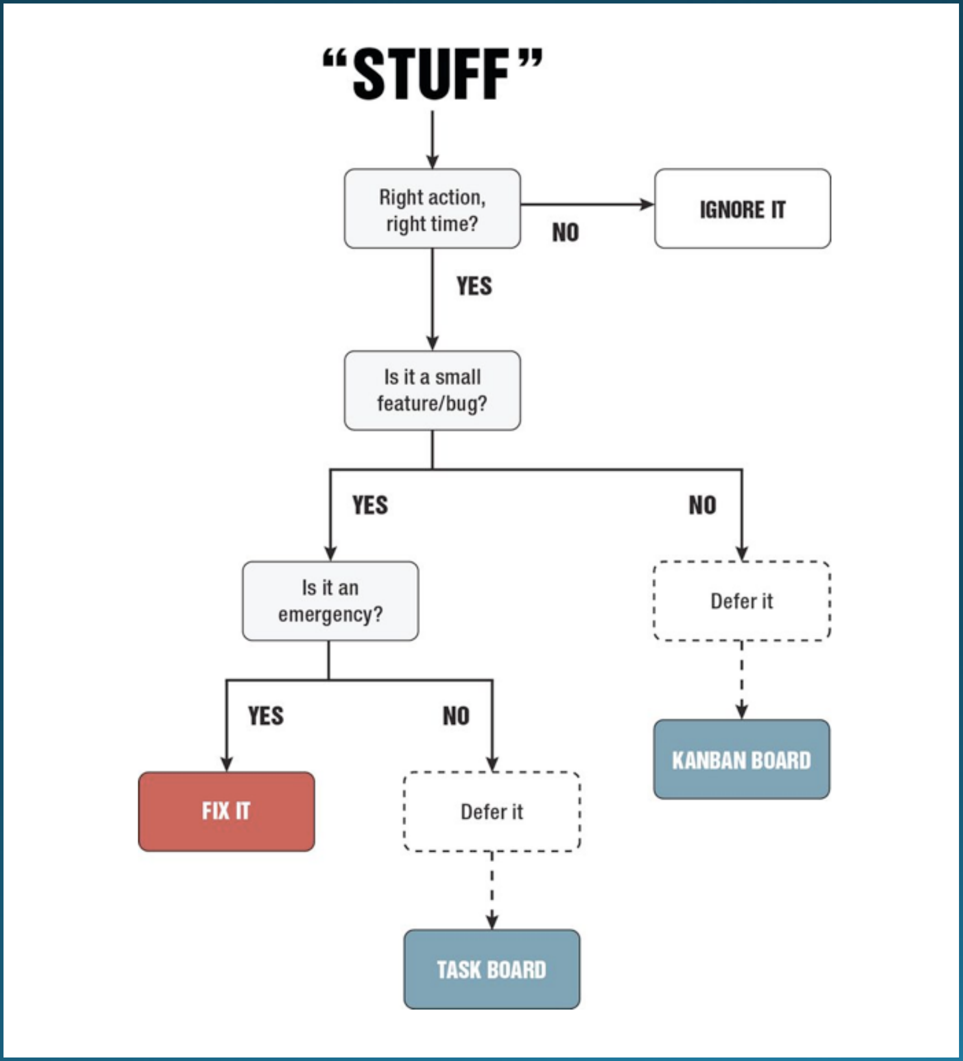


Figure 13-1. 80/20 Rule

Pull don't push

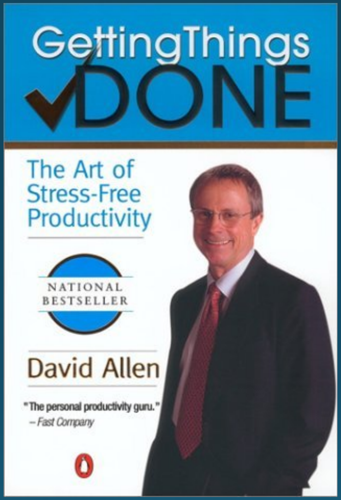
- Lean Premises: "Don't be a feature pusher"
  - Adding features based upon customer demand and interest
- Whoa! This flies in the face of customers not knowing what they want. Furthermore this might get you in trouble with market effects as well

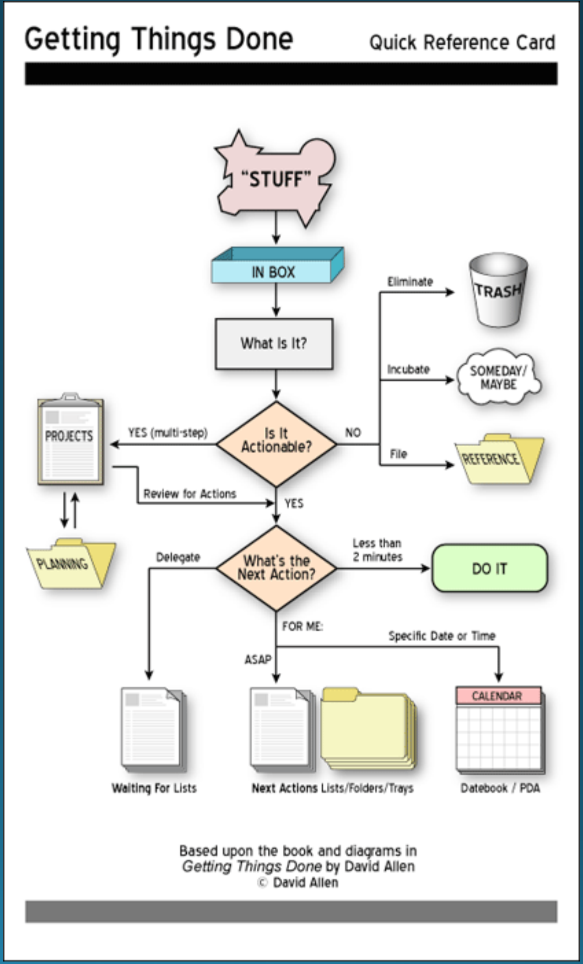
Task Prioritization



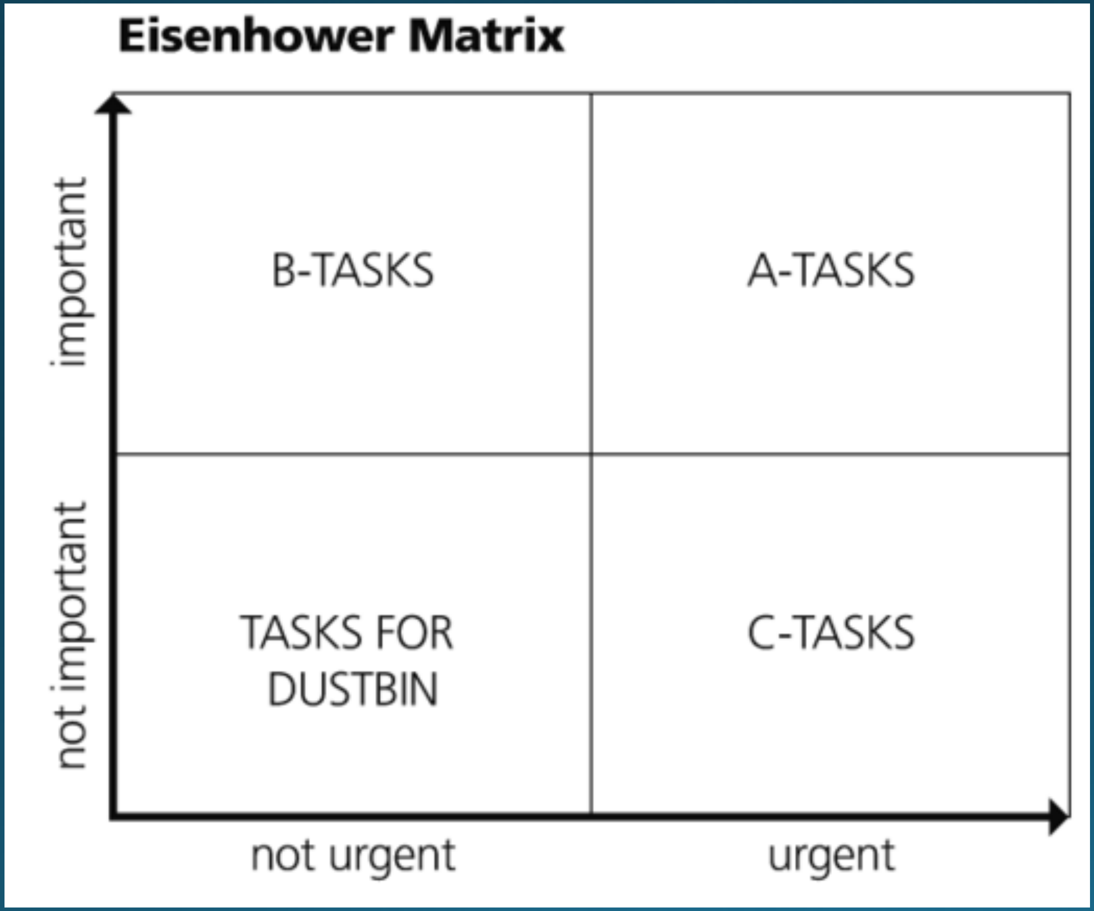
Just GTD for Programming?

- This seems to remind me of productivity schemes like *Getting Things Done* methodology
- Ok ok ok…we get it now!


Getting Things DONE
The Art of Stress-Free Productivity
NATIONAL BESTSELLER
David Allen
"The personal productivity guru." – Fast Company



Ike did this before them?



It seems there really isn't much new under the sun in terms of process management, but then how come we keep pushing here?

Step 3: Scale

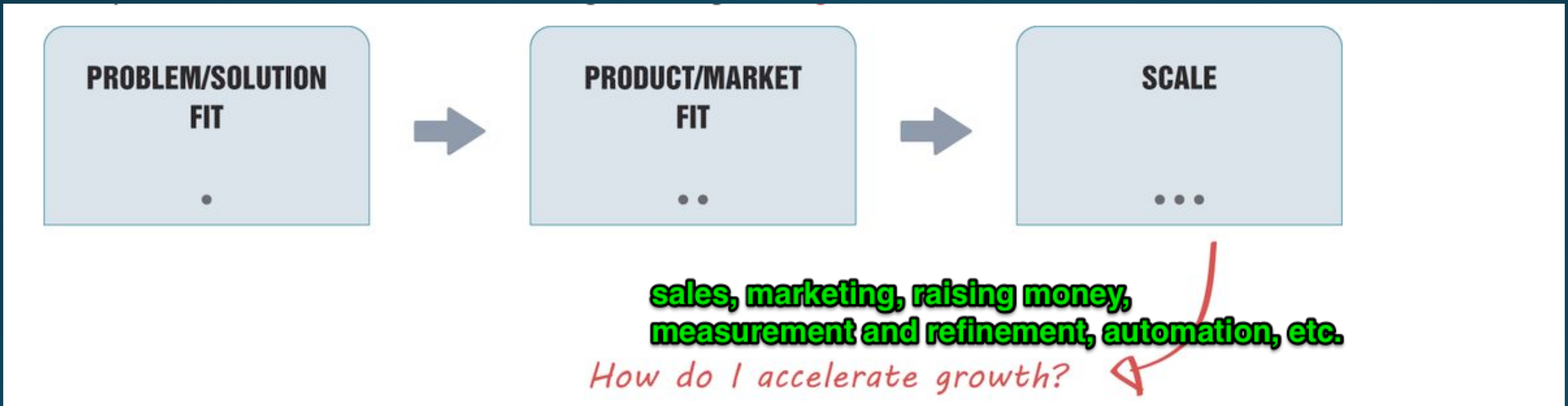

PROBLEM/SOLUTION FIT
•
PRODUCT/MARKET FIT
••
SCALE
•••
sales, marketing, raising money,
measurement and refinement, automation, etc.
How do I accelerate growth?

Figure 15-1. After product/market fit

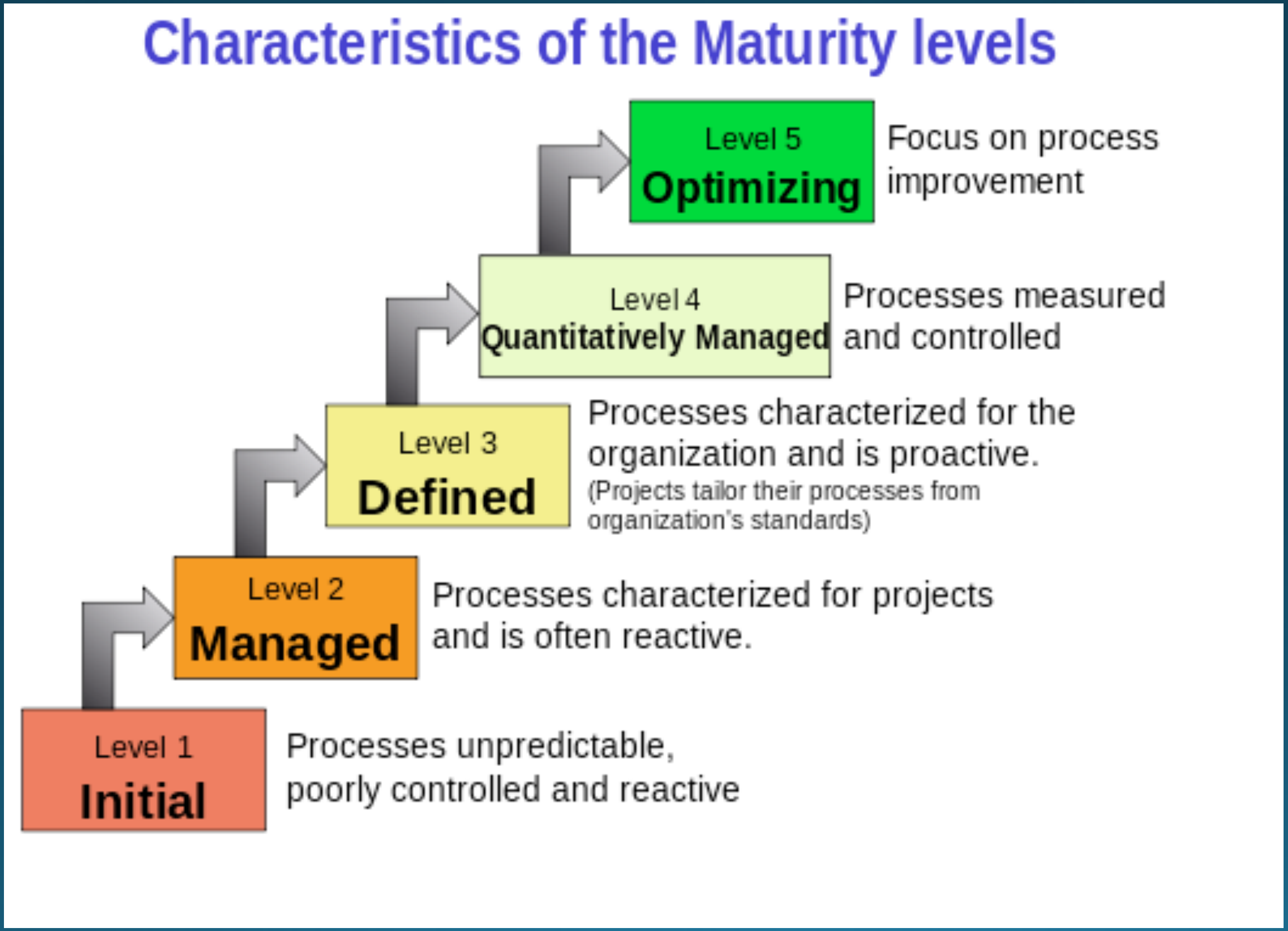
_AD, Lean, Agile Truths

- While JAD, RAD, Lean, Agile… can deal with change and unclarity of thought it may…
  - Build wrong thing many times?
  - Suffer from "marry" prototype problem
  - Damage us with early releases
  - Be difficult to cost control or budget
  - Suffer from difficulty of org buy-in
  - Have its own arbitrary adherence problems like any other process model (see my example earlier)

Process Maturity

- CMMI - Capability Maturity Model Integration
  - Required by many (ex. DoD, US Govt.)
- Basically talks about having formalism and repeatability to your SE processes with eventual focus on improvement

CMMI Visualized


Characteristics of the Maturity levels
Level 5
Optimizing
Focus on process improvement
Level 4
Quantitatively Managed
Processes measured and controlled
Level 3
Defined
Processes characterized for the organization and is proactive.
(Projects tailor their processes from organization's standards)
Level 2
Managed
Processes characterized for projects and is often reactive.
Level 1
Initial
Processes unpredictable, poorly controlled and reactive

Over-Engineered

- Watch for Architectural Astronauts
  - Afflicted with Analysis Paralysis
- Failure (or fear) to launch
- Just one more thing…
- “Spruce Goose” projects

Under-Engineered

- Quick and dirty
- Cowboy coding, magical thinking, no compromises
- Sometimes it makes sense for exploration
- Both these points are the extremist position to too much process or too little

Product over process

- Does process really matter how you did it if result is ok?
- To us or boss
    - Yes - if not repeatable
    - Yes - if it costs too much
    - Yes - if it was a miserable experience
- To customer
    - Who cares about sausage unless pricey or unavailable!

Worse is better?

- Counterintuitively sometimes this is true
  - Less functionality can be better quality wise
  - More functionality can be worse system wise, but may be better market wise?
- Consider the effect of feature/UX from market forces
- Human's employing simple heuristics to navigate the market will often introduce things which may sabotage the engineering purity
  - Ex: Having to jam in features for market acceptance only to see them unused

Always Be Pragmatic

- Principle of Good Enough
- 80/20 Rule - Pareto distribution, 80% of value from 20% "Law of vital few"
- KISS - Keep it simple, stupid
- MVP - Minimal Viable Product
- More to be created

Theory and Practice

- There has been some sense that theory is well just theory
    - I've already noticed the perk ups in various lectures
- So we take some minutes to tell war stories to see the patterns here

Story Time

- FieldLevel: Practice and premise issues
- 'Start-up X': The anti-FieldLevel
- Dokoni: Solving today's problem >10years ago
- ServerMask: the warm-up project with success?
- ServerDefender: 4 year MVP
- w3compiler - scratched an itch years early

Story Time

- pageXchanger - yeah nobody wanted it
- URLSpellCheck - MVP example no demand
- CustomError - Nice to have but …
- httpZip - better copies make more money
- ClickFormant - good idea, too early, didn’t stick it out

Story Time

- ZingChart - the pivot from ClickFormant, success but…
- ZingGrid – MVP then explode…recovery mode
- More in the hopper but…

Morale of Stories?

- There is no secret, other than everyone wants there to be one
  - Maybe we should sell that!!!
- Have a set of techniques to pull out per problem, roll with the challenges, be patient, be precise, and try to have fun