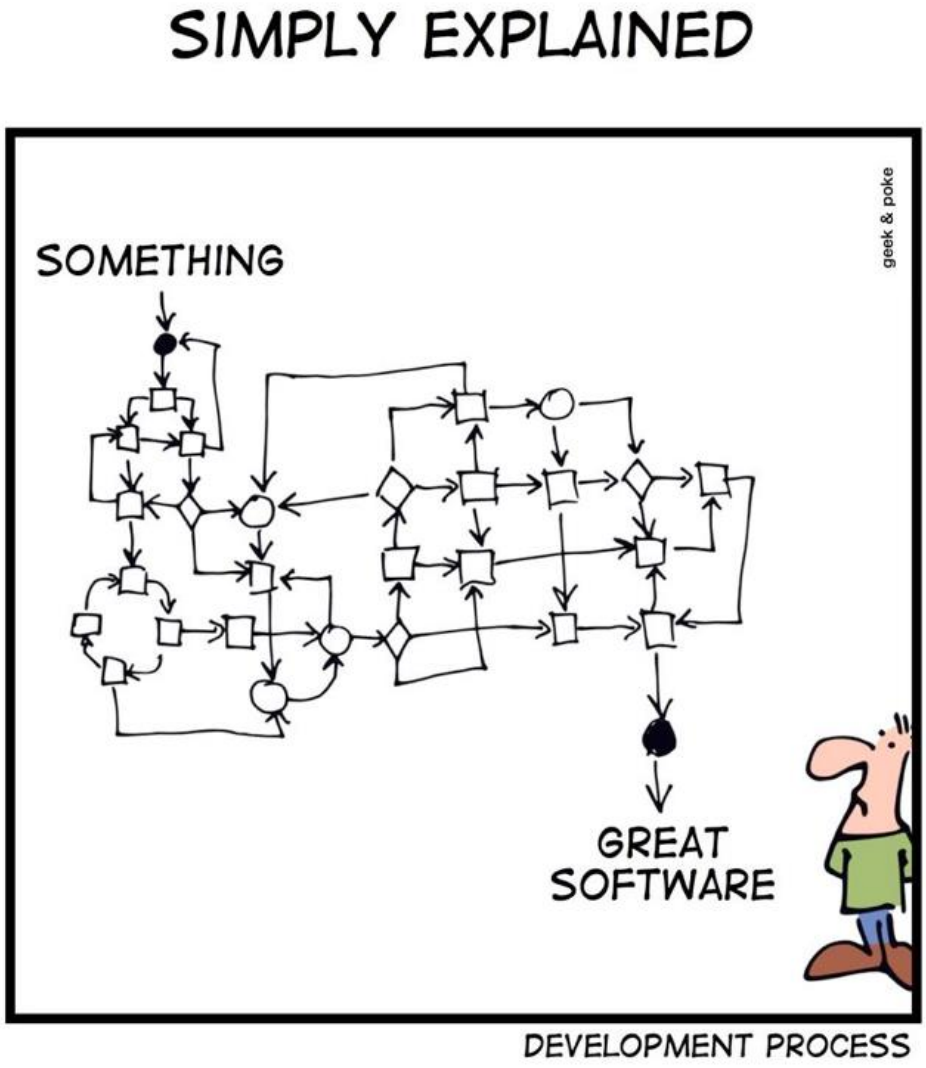
SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

Northeastern University
College of Computer and Information Science
440 Huntington Avenue . 202 West Village H . Boston, MA 02115 . T 617.373.2462 . ccis.northeastern.edu

UNIT OBJECTIVE

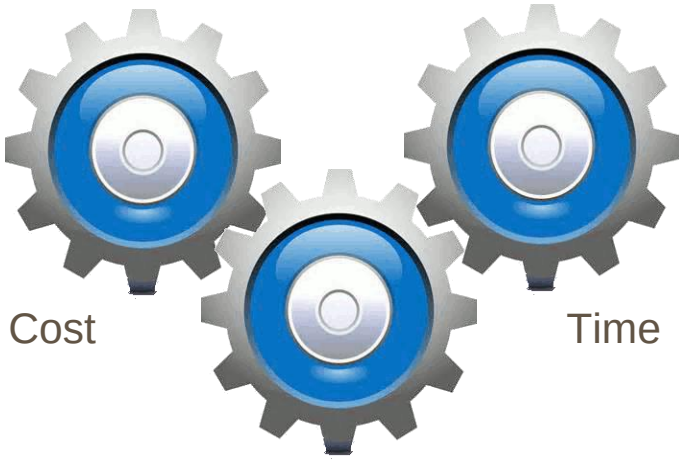
- Understand the influences on a project
- Understand what a software process is
- Understand two common models


SIMPLY EXPLAINED
SOMETHING
geek & poke
GREAT SOFTWARE
DEVELOPMENT PROCESS

WHAT EACH PARTY CONTROLS

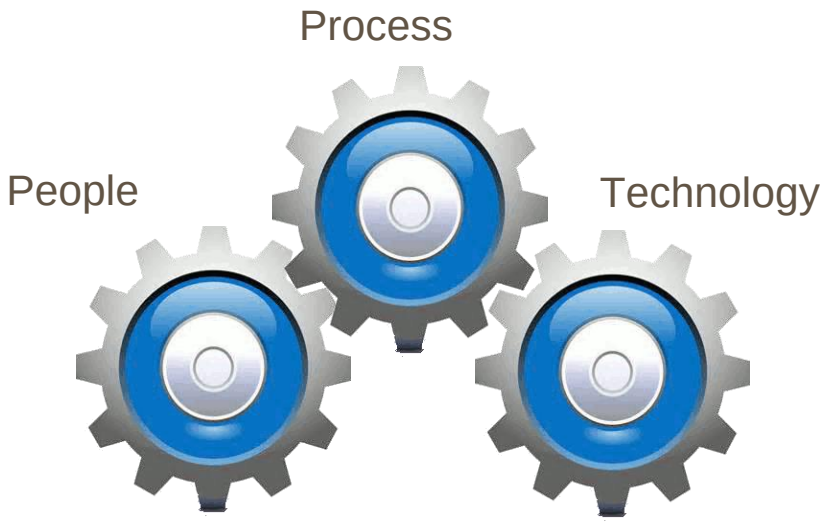
Client Side

Every software project has three client controls



Tech Side

The tech team has three controls



Software Engineering is about managing the client side and defining the tech side while managing risk.

MOST EVERYTHING INVOLVES TEAMS

- The effectiveness of the team relates directly to success
- Working with and within teams requires extra effort for
  - ***Communication***

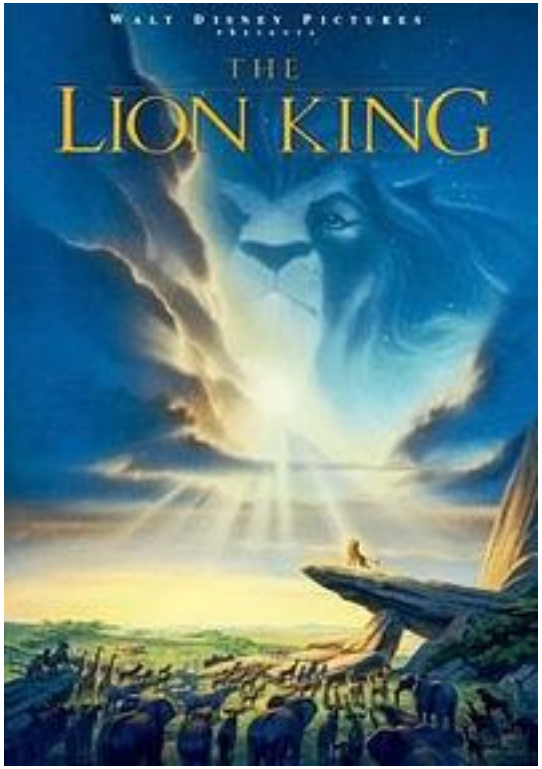
- Ever play the operator game?
  - ***Documentation***
  - ***Tooling***
  - ***Hand-offs*** *(process exchanges or role turn-over)*

- Remember, you cannot read other people's minds

CIRCLE OF LIFE

1. Teams come, operate, evolve or disband
2. People come, grow, and eventually move on
3. Projects come, grow, enter stasis or evolve

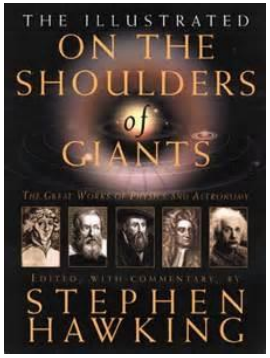
Your project has to accommodate these facts of project life



PROJECT INFLUENCES

- Scale
  - Affects the ability to know “everything”
  - Complexity becomes a critical factor, if it wasn’t already

- Legacy
  - Rarely is everything from scratch
  - Being able to extend others’ work is essential



PROFESSIONALISM

Personal Ethics

- Confidentiality
  - Respecting confidences of employers or clients regardless if there is a formal agreement
- Competence
  - Accurately reflect what you can do and accept only work that is within your competence
- Intellectual Property
  - Protecting the IP of employers and clients
- Misuse
  - Do not use skills or resources inappropriately

Effects

- Developers and administrators may have access to highly confidential information
- Systems that do not work can destroy a company
- IPR violations can be result in fines or cease and desist orders
- System abuse can paralyze a company

IEEE Advancing Technology for Humanity

http://www.ieee.org/about/corporate/governance/p7-8.html

acm Association for Computing Machinery

https://www.acm.org/about/code-of-ethics

PROCESS

- **noun** pro·cess
  a series of actions that produce something or that lead to a particular result
  http://www.merriam-webster.com/dictionary/process

Typical “Good” Qualities

Effective	Actually Used
Efficient	Reusable
Relevant	Managed
Valid	Measurable
Usable	

Beware: it is easy to become over-zealous or lost in process

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

- Purpose
  - Lead to good software
  - Reduce risk
  - Enable visibility and measurement
  - Enable teaming

- Key attributes
  - Outcomes/results of processes are key deliverables or products
  - Roles are clear
  - Pre and post conditions are understood and held true

KEY ELEMENTS IN ANY SDLC

1. Feasibility

2. Specification
3. Architecture and Design
4. Development
5. Validation

6. Evolution/Maintenance

The devil is in the details of how the steps are organized and executed

The Promise

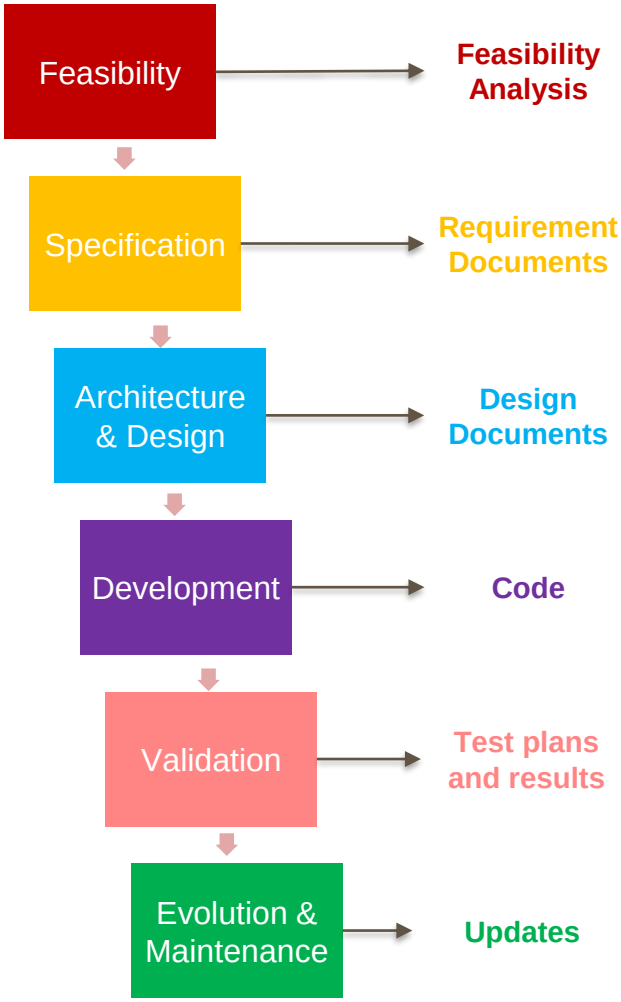
The Reality

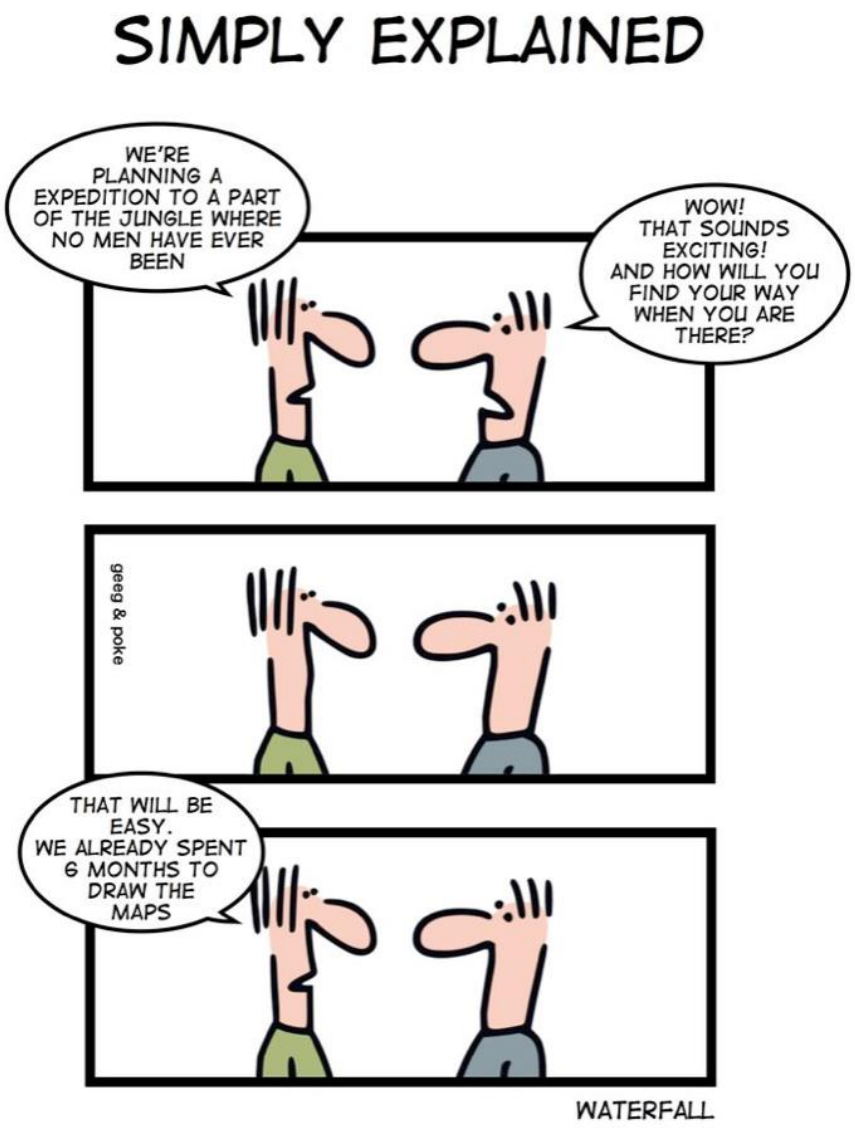
Nothing is ever as simple as it seems

PROCESS MODELS

WATERFALL MODEL (CIRCA 1968)

- Sequential process phases
  - One step completes before next one starts
- Rational process
  - Enables careful planning
    - *This is how construction is done.*
  - Good for
    - *some piece of the system cannot be easily changed (e.g. hardware)*
    - *where explicit and exhaustive testing is required before launch*
- Challenges
  - Heavyweight process
    - *Meaning the process is followed systematically and completely (slow)*
    - *Specification is a negotiation process*
    - *Specifications precede the system*
  - **World rarely is known upfront and even more rarely stays fixed**
    - *Hard to adapt to upstream changes once the step completes*




SIMPLY EXPLAINED
WE'RE PLANNING A EXPEDITION TO A PART OF THE JUNGLE WHERE NO MEN HAVE EVER BEEN
WOW! THAT SOUNDS EXCITING! AND HOW WILL YOU FIND YOUR WAY WHEN YOU ARE THERE?
geeg & poke
THAT WILL BE EASY. WE ALREADY SPENT 6 MONTHS TO DRAW THE MAPS
WATERFALL

WATERFALL MODEL

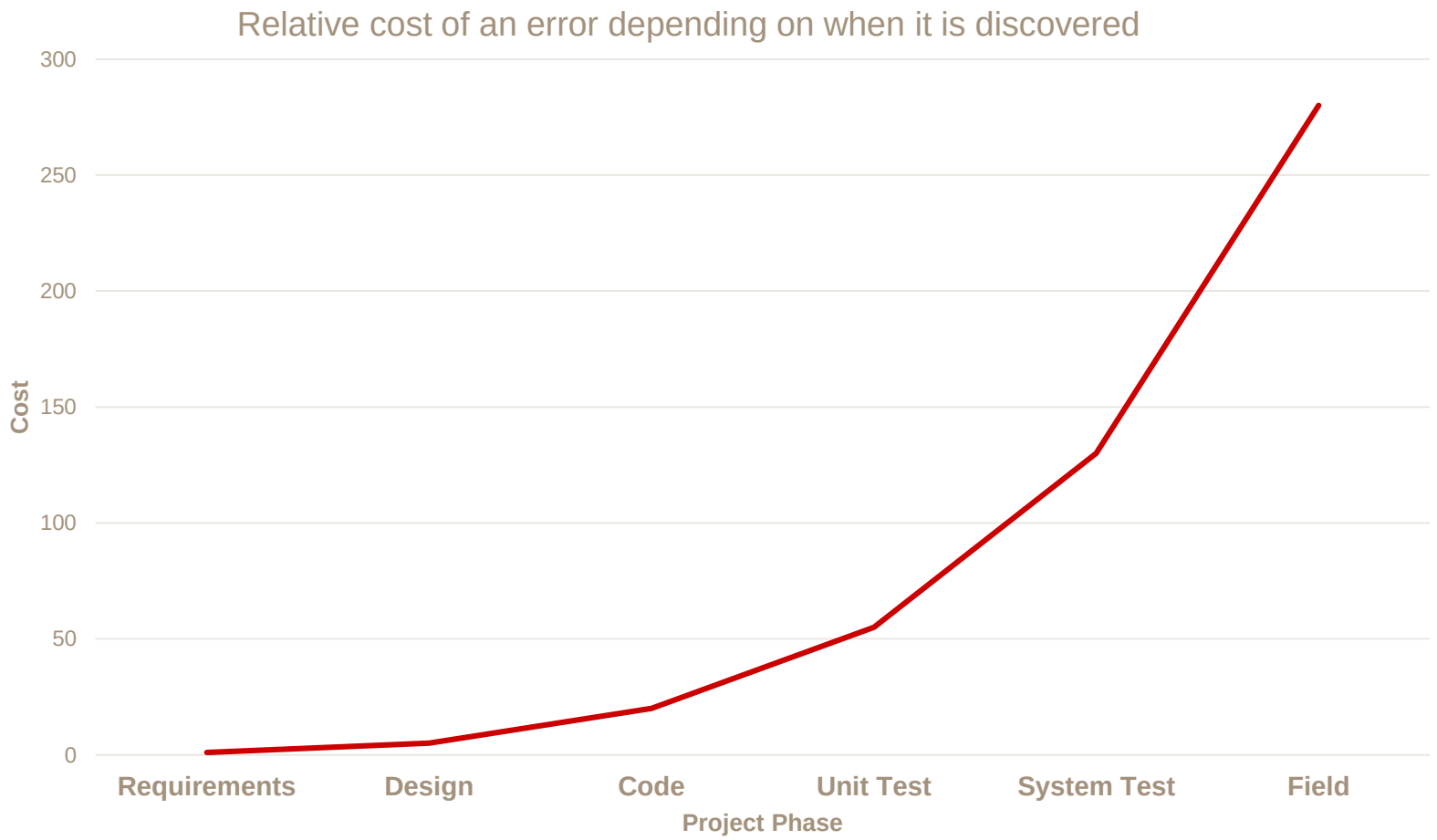
- Real projects rarely follow a sequential flow
- Hard to state all requirements explicitly
- No maintenance or evolution involved
- Customer must have patience
- Any blunder can be disastrous

BOEHM'S FIRST LAW

Errors are most frequent during *requirements* and *design* activities

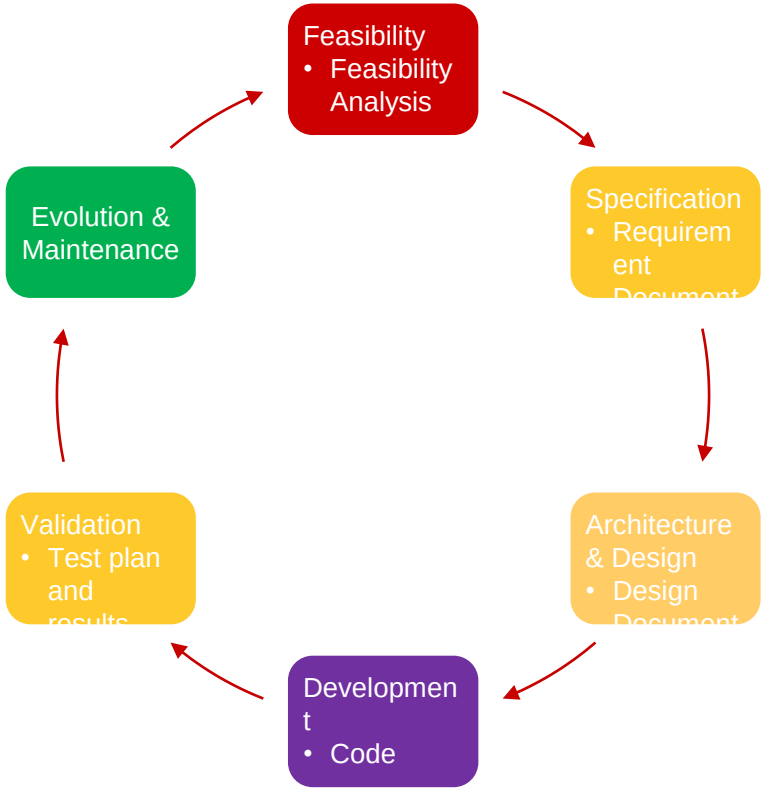
and are more expensive the later they are removed.

WHAT THIS MEANS IN PRACTICE



ITERATIVE MODELS

- System is created by successive versions.
  - Go through each process step, then iterate
    - *Similar to how you are taught to write a paper*
  - Includes feedback between steps
- Lowers the cost of implementing requirement changes
- Allows some client/user feedback to be considered
- Smaller sized steps means delivery of something comes sooner
  - Value is created earlier
- It may not be clear where in the program the project is
- Changes can lead to messy designs and implementations



AGILE MANIFESTO

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

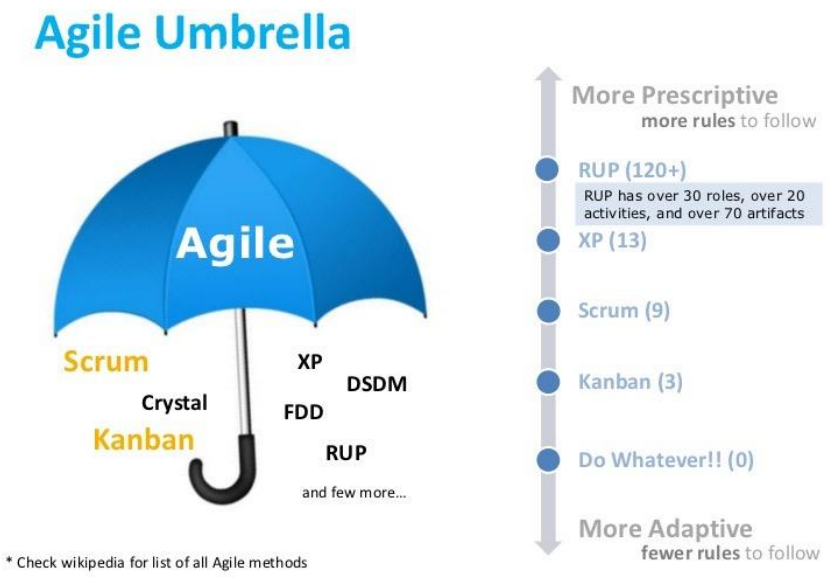
Responding to change over following a plan

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

http://agilemanifesto.org/

- This is a response to over-zealous and rigid process mongering
- Emphasizes getting to the right result versus creating a lot of useless documents, over-planning, or blindly following process
- However, this is NOT a repudiation of documentation or plans.

AGILE IS A SET OF SDLC APPROACHES



Glossary:

- RUP – Rational Unified Process
  - *https://en.wikipedia.org/wiki/Rational_Unified_Process*
- XP – Extreme Programming
  - *https://en.wikipedia.org/wiki/Extreme_programming*
- DSDM - Dynamic systems development method
  - *https://en.wikipedia.org/wiki/Dynamic_systems_development_method*
- FDD - Feature-driven development
  - *https://en.wikipedia.org/wiki/Feature-driven_development*

Borrowed from Haresh Karkar
http://www.slideshare.net/hareshkarkar/overview-of-agile-methodology

AGILE

- Emphasis
  - producing small increments of software in a reasonably short time frame
  - Entire process is run during a sprint
  - Sprint results are deployed

- Antithesis of Waterfall
  - *Plans develop incrementally and evolve*
- Client collaboration versus client negotiation
- Specification follows from working system, not the reverse
  - *Immediate feedback from deployment*
- Responding to change rather than following a plan
  - *Enhancements, new features, and bug fix are all prioritized as candidates for focus during next sprint*
  - *Emphasis on keeping scope small*
- Although the impact of changes will grow over time

> "[…] is like driving at night in the fog. You can only see as far as your headlights, but you can make the whole trip that way."
>
> — E.L. Doctorow, Writers At Work: The Paris Review Interviews

SCRUM

Emphasis on small, semi-independent teams ideally delivering discrete pieces of a system

Team ideally has total responsibility for the components it produces
Leads to devOps models

1. Team
   - *Small, cross-functional, self-organizing units*

2. Scope
   - *Small deliverable scope delivered in consensus priority order*
   - *Priorities can be adjusted (typically at sprint start)*

3. Timeline
   - *Small iterations (2-3 weeks is typical) emphasizing delivery at the end*

HOW SCRUM TYPICALLY OPERATES

A sprint is one iteration through the process

The backlog contains all the work needing doing

- Includes features and other tasks

User stories describe the function from the consumer's perspective

- ✦ The User may be another software component/system.

- You estimate how much work/time each User Story will take
- The client provides her view of the priorities in the backlog.
- The tech team reassesses priorities to allow for dependencies or difficulties
- The backlog is now a roadmap for the sprint or day within the sprint

Daily stand-ups to discuss progress and plans for what is next

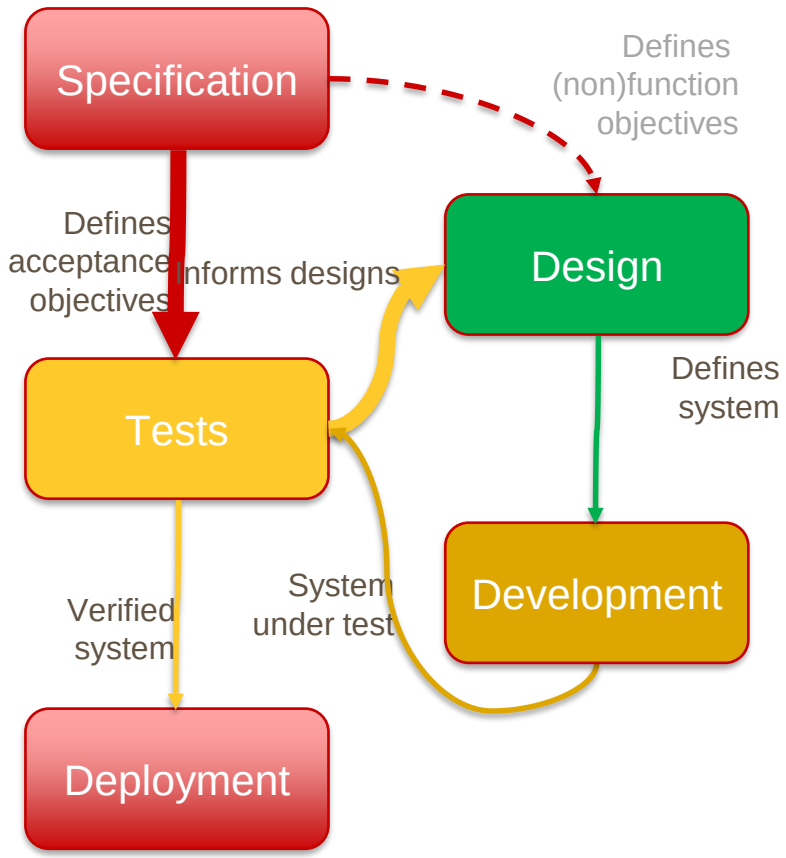
- A scrum-master choreographs the sprint and keeps the team focused and distractions at bay.

SCRUM

In rugby, a scrum is the way you restart the game after a minor infraction.

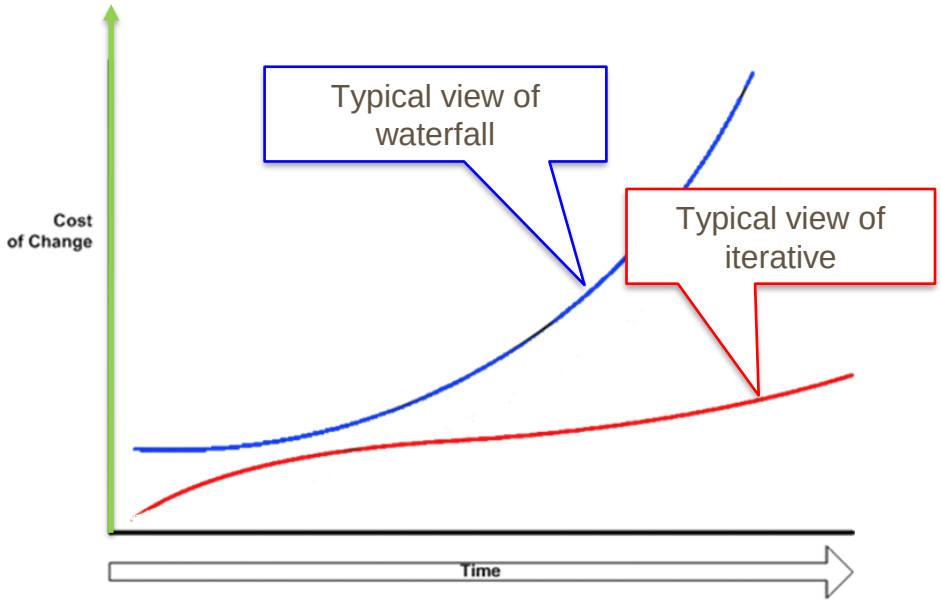
TEST-FIRST DESIGN

- Puts test specification as the critical design activity
  - Understands that deployment comes when the system passes testing
- Clearly defines what success means
  - No more guesswork as to what "complete" means
- The act of defining tests requires one to understand how the solution works



KEY CONCERNS DRIVING IN SELECTING A PROCESS

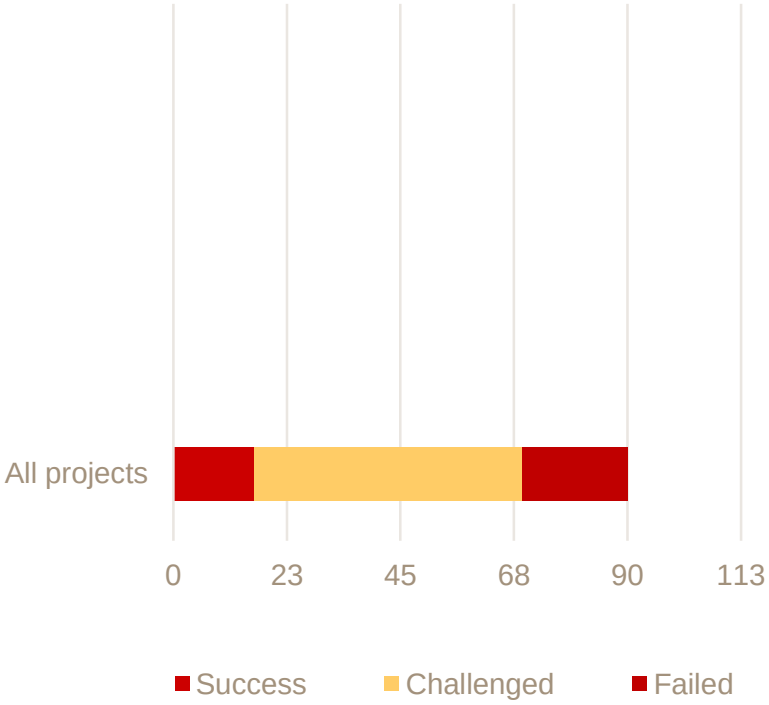
- What is the net tolerance for finding errors in deployment?
- How fast is the market moving?
- Are the teams experienced with a particular process?
- Is the contract fixed and firm?
- When do the clients expect to see something?



Adapted from
http://www.agilemodeling.com/essays/costOfChange.htm

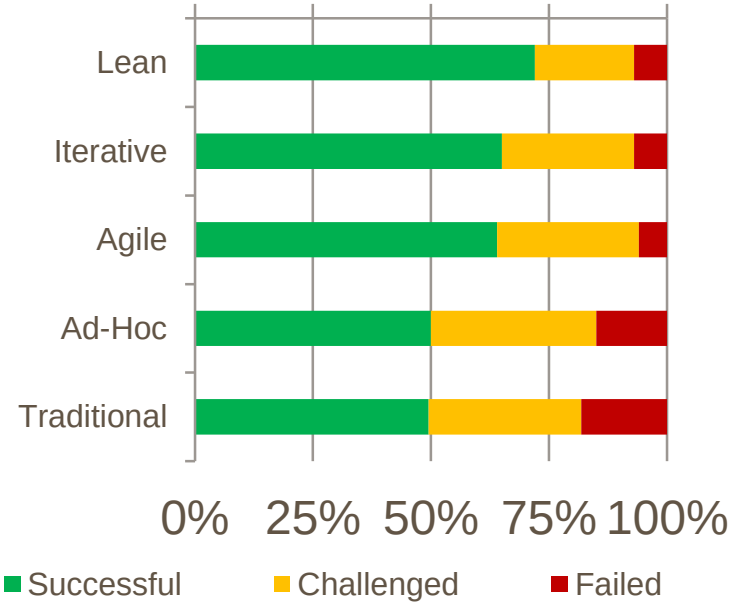
EVEN WITH ADVANCES IN PROCESS, PROJECT SUCCESS REMAINS RISKY

Pessimist View



Standish Group (UK), Chaos Study, 1995

Optimist View



Dr. Dobb's Journal 2013 IT Project Success Survey posted at www.ambysoft.com/surveys/

WALKING THE SDLC STEPS

FEASIBILITY

- Determines if a project should be attempted
  - Usually done once at the beginning by senior (trusted) team members

Feasibility study is a proposal

- Does not require prototyping, but often includes it

The decision maker is the audience

- This person may not be sufficiently technical
- In large organizations, this can be a walk-up multiple hierarchies

- Budget processes and staffing usually follow from a positive response

- Two outcomes
  1. Yea
  2. Nay

WHAT GOES INTO A FEASIBILITY STUDY

1. Recommendation
2. Technology
3. Economic
4. Legal
5. Operational
6. Schedule

UNCERTAINTY MAKES THIS VERY HARD

Challenges

- Clients are unsure what they need at a useful level of detail
- Benefits are hard to quantify
- Impacts and recognizing unintended consequences is even harder to quantify
- Approach is often based on very rough guesses
- Organizational structures may need change
- Assumptions may be faulty

Mitigations

- Experience can guide process
  - *But the most experienced people may not be the most technically current*
- Solicit support and build interest for the project
  - Beware of irrational enthusiasm
    - *Leads to unreasonable expectations*
    - *Senior executives rarely forget your promises*

THE BOSS' VIEW (ADAPTED FROM BILL ARMS)

The Main Line

- Senior member(s) of the client's organization decide whether to begin a major software project.
  - Client: who is this project for?
  - Scope: is it well defined? Where are there dependencies and on whom?
  - Benefits: are the benefits real and quantifiable? Do I trust these numbers?
  - Technical: Is the project possible?
    - *Is there at least one technical way to carry out the project?*
  - Resources: what are the estimates of staff, time, equipment, etc.?
  - What are the options if the project is not done?

Additional Considerations

- Do I trust this team?
- Have we tried this before?
- Market maker? Fast follower?
- Is this really worth investing in?
- Are there IPR issues?
- License dependencies?
- Can this organization pull this off?
  - Management capabilities
  - Development capabilities
  - Operational capabilities
  - Sales capabilities