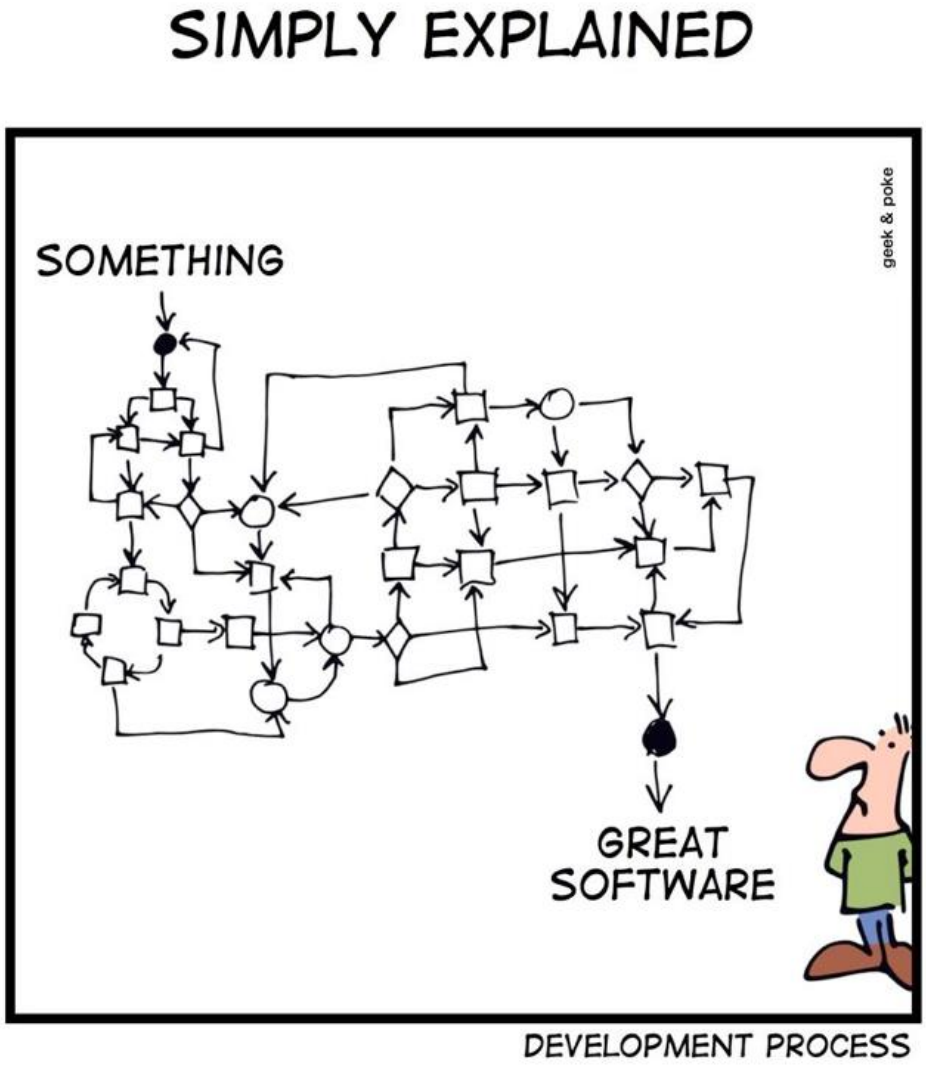
SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

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


SIMPLY EXPLAINED
SOMETHING
geek & poke
GREAT SOFTWARE
DEVELOPMENT 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

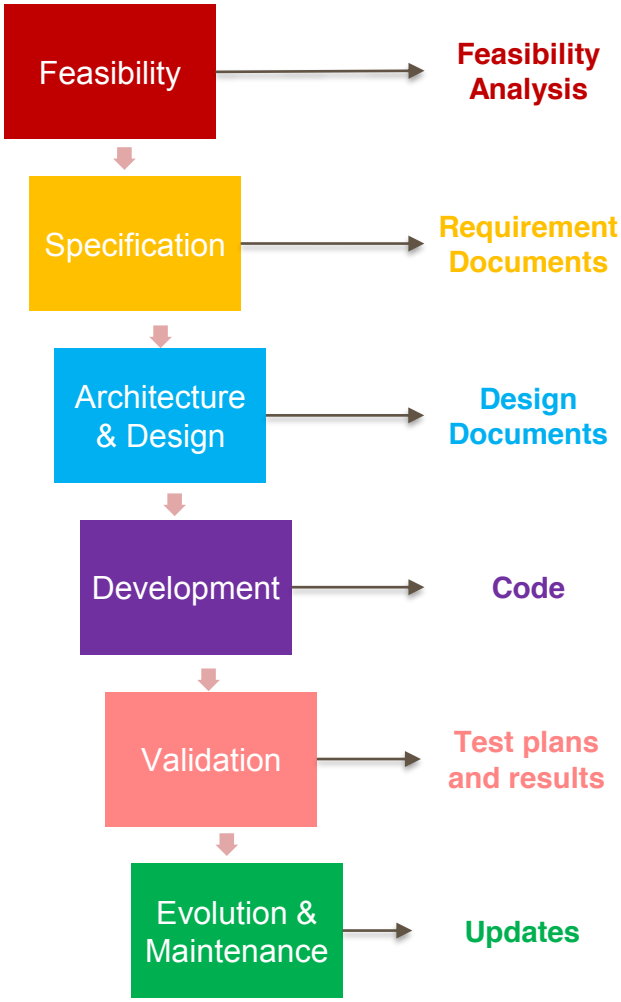
6. Evolution/Maintenance

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

PROCESS MODELS

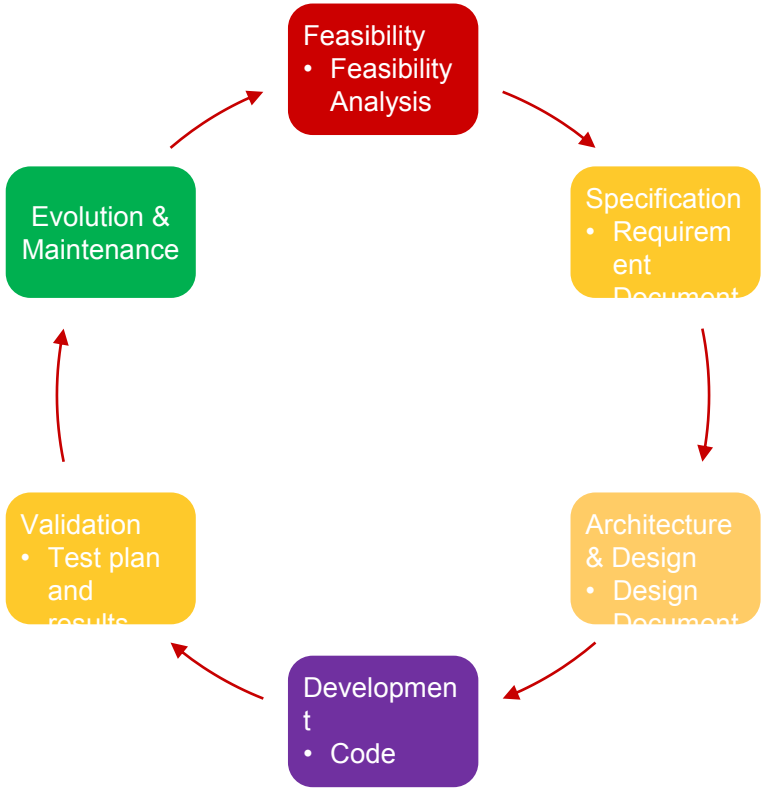
WATERFALL MODEL (CIRCA 1968)

- 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*



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

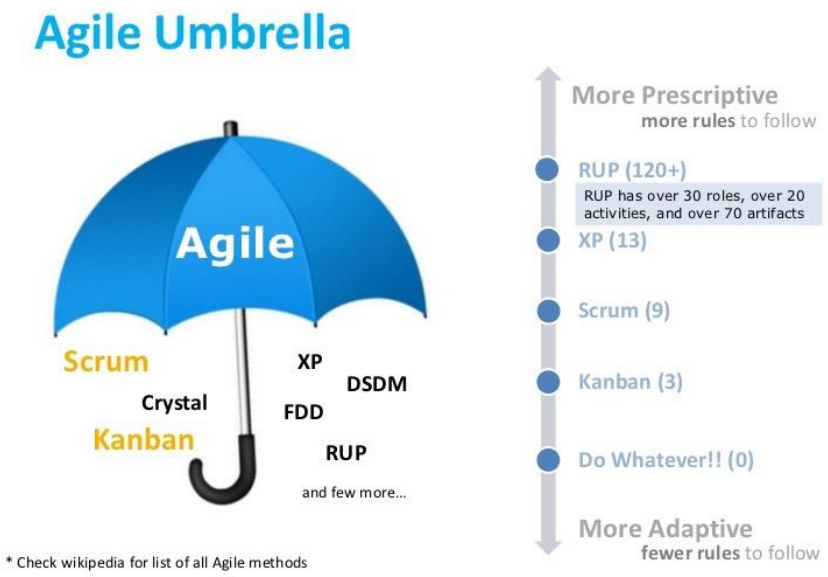
Responding to change over following a plan

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*