AUCSC 320
Models of Software Engineering

How do you order the stages?

- Requirements, Design (High and Low Level), Development, Verification, Deployment, Maintenance
- Everybody agrees Wrap Up is at the end

Linear Models (Predictive Models)

- Each stage finished, reviewed, and finalized before moving on (pure linear)
  - Requirements driven (works well for a classroom)
  - Nicknamed BDUF = Big Design Up Front
  - Progress easy to see
  - Less emphasis on a prototype
  - Maintenance completely separate from creation

Linear Models (Predictive Models)

- Each stage finished, reviewed, and finalized before moving on

  - Questions to keep in mind:
    - What if user's needs change while product is being developed?
    - Is it ok for Testing to be completely separate from Development?
    - Should one spend so much time on activities that aren't directly code (final product)?

Linear Models (Predictive Models)

- Especially good when
  - Project is small
  - Requirements are simple, and you know them
  - Requirements won’t change
  - Deadline is soon (time is short)

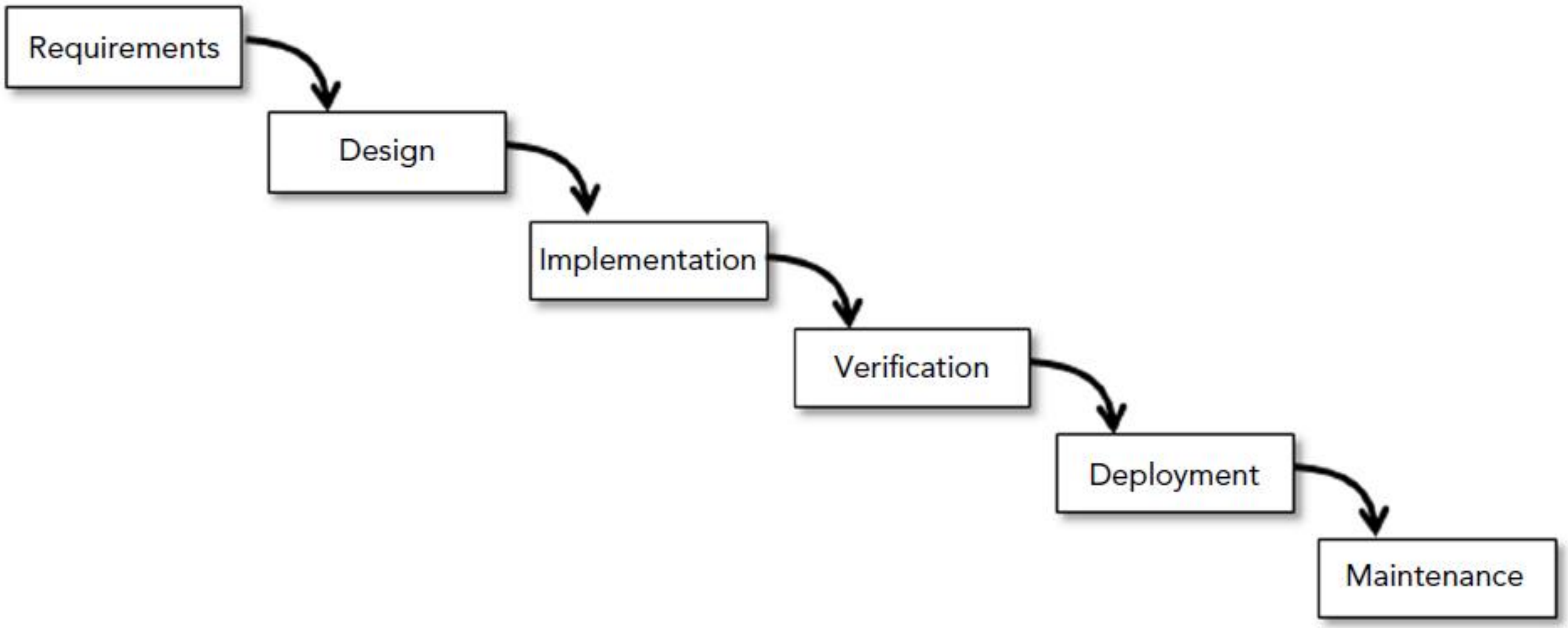
Adaptive Models (Iterative Models)

- Revisit stages
- Re-evaluate and change direction, if necessary

- E.g. change the product goals

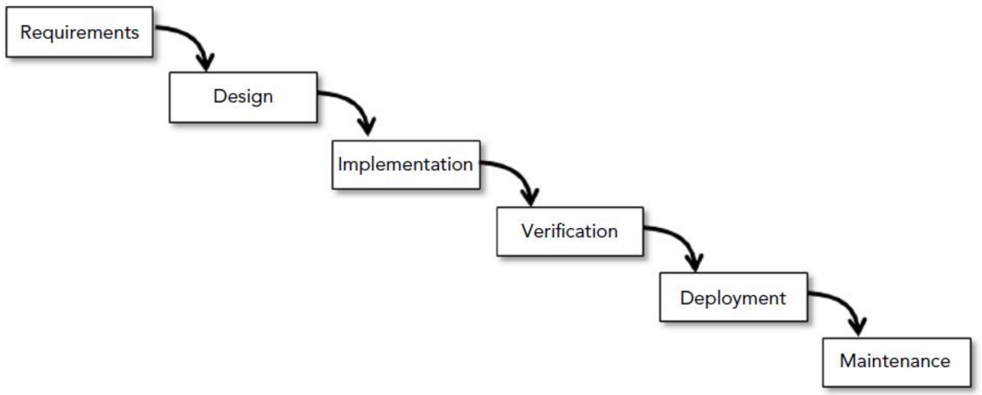
Waterfall Model

Picture from Stephens



- First model
- Strict order of steps

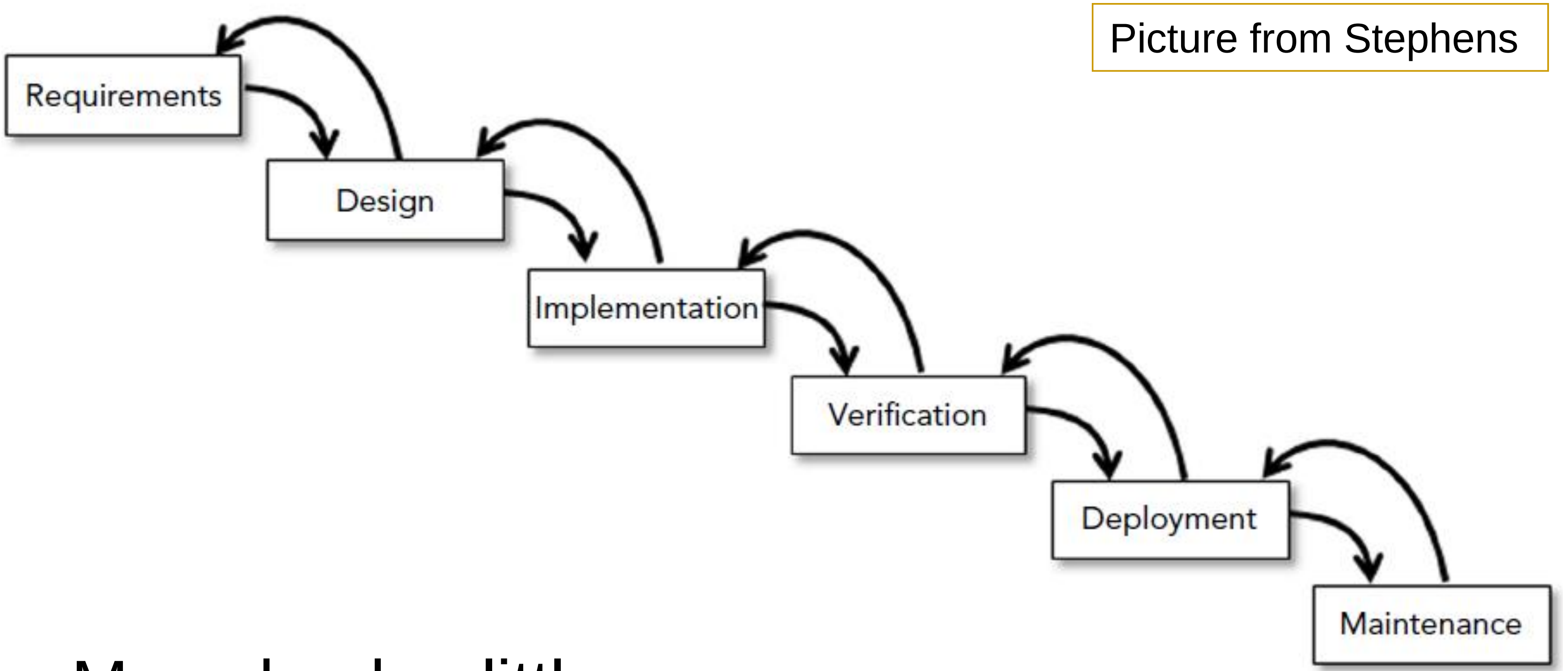
Waterfall Model



- When:
  - Requirements all known in advance
  - Requirements won’t change
  - Team has experience
  - Enough time to do everything sequentially
  - Testing is easy

Waterfall With Feedback Model

Picture from Stephens



- Move back a little
  - One step, maybe two, if necessary
- Moving back is a step backwards – try not to

Sashimi

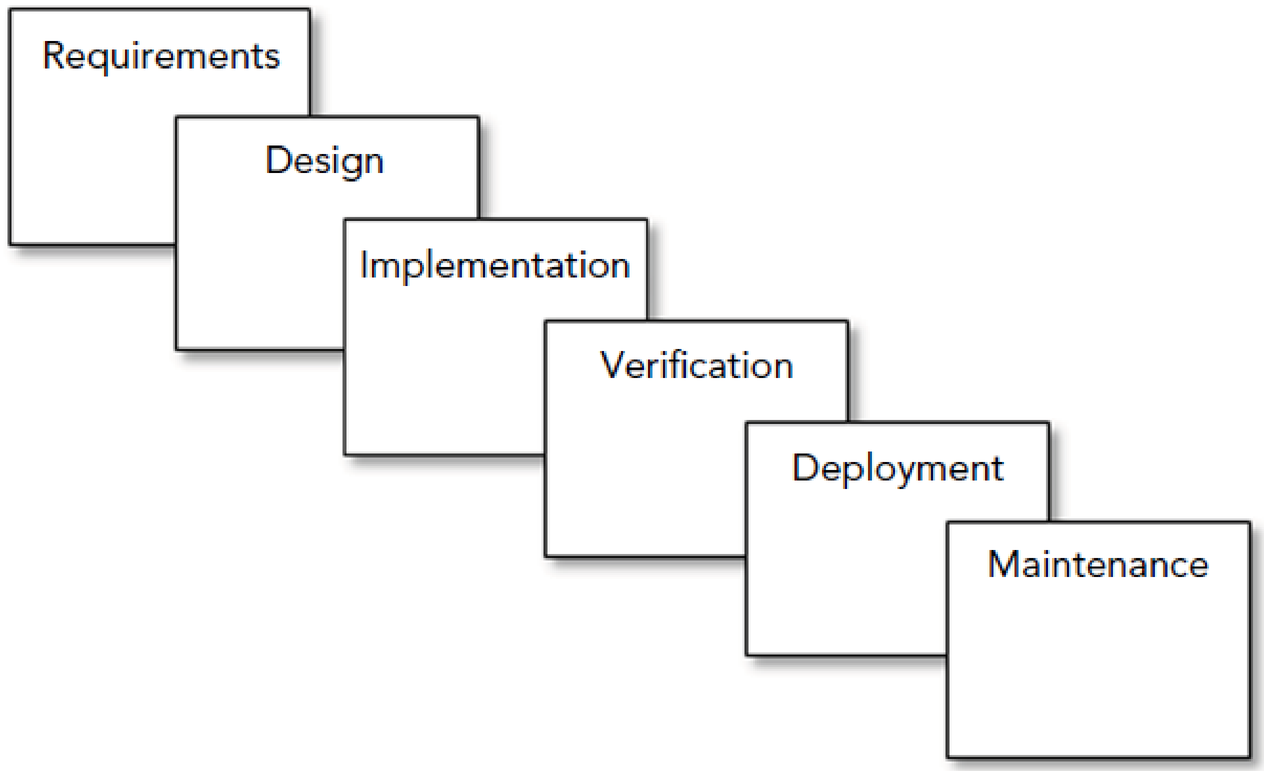
- Also called Sashimi Waterfall
- Also called Overlapping Waterfall
- Also called Waterfall with Overlapping Phases

- Named after a Japanese dish

https://www.manusmenu.com/salmon-sashimi-with-ponzu

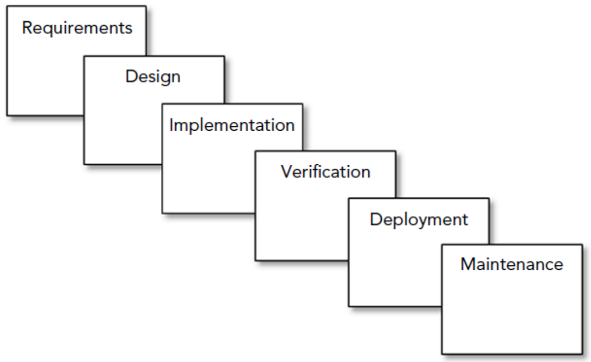
Sashimi

Picture from Stephens



- Coders don’t need to wait for designers to be finished
- Does Deployment really overlap Verification?

Sashimi



- Prototype: Requirements, Design & Code all at same time
- Easier to change a previous stage based on going ahead
- Stuff done “ahead” should not be “expensive”

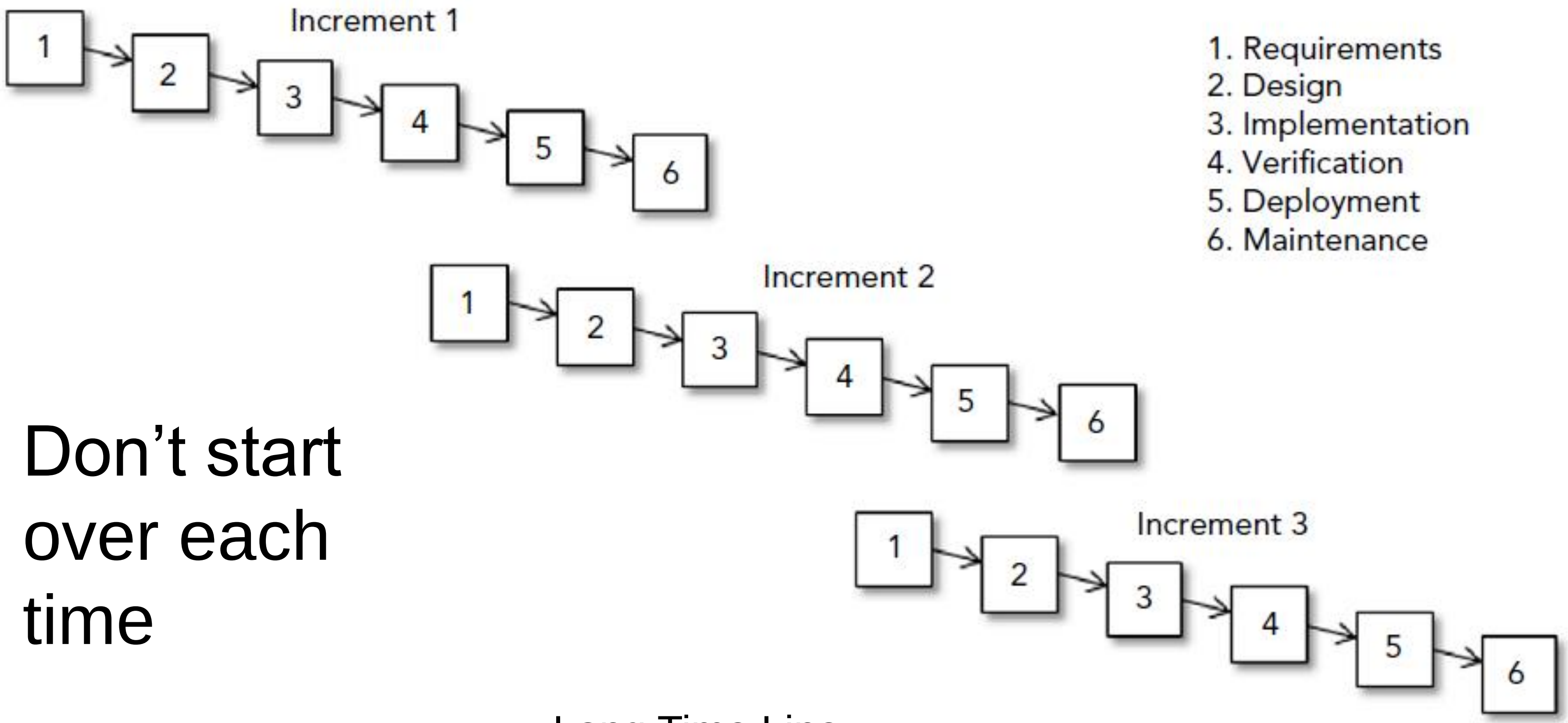
Incremental Waterfall

- Also called Incremented Waterfall
- Also called Multi-Waterfall, or Multiple Waterfall

- Each increment (i.e. each waterfall) ends with a usable product
- Each increment has more features than previous
- Increments over a long time line

Incremental Waterfall

Picture from Stephens

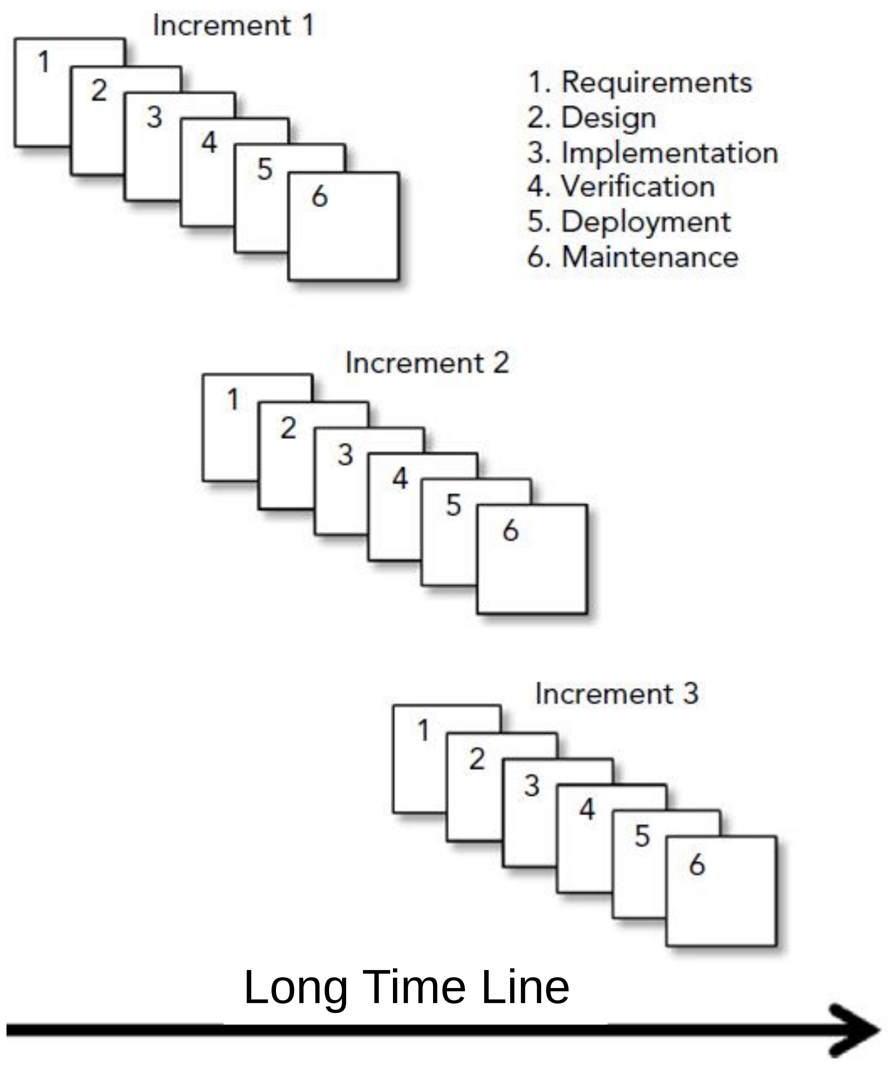


- Don't start over each time

Long Time Line

Incremental Waterfall

Picture from Stephens



Incremental Waterfall

- Big changes between increments are NOT possible
  - User has a product after each, and you don’t want to completely change the feel
- When is it best just to start over?
  - “I’ve worked with programs that had been used and modified for decades, and it was nearly impossible to make any significant changes without breaking something” (Stephens)

V-Model

- Waterfall bent in middle and folded up
- Design and Testing driven

Linear Models Summary

- “If the design is correct and everything stays on track, the project is like a luxury train coasting majestically into Grand Central Station.
- However, if something goes wrong, the project is more like a train engulfed in flames and speeding toward a dynamited bridge.”
- Stephens, pg. 283

Linear Models Summary

- Biggest Problems:
  - Unexpected Change
  - Fuzzy Requirements

Adaptive Models (Iterative Models) Summary

- Start: move through stages fast to create a smallest useful program (subset)
  - This should always give a baseline

- Increment: keep adding to this (design, code, test)

  - If wrong direction, just get rid of current increment