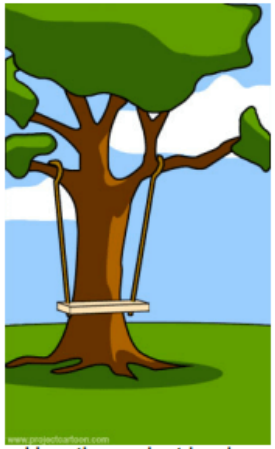
CSci 3081W: Program Design and Development

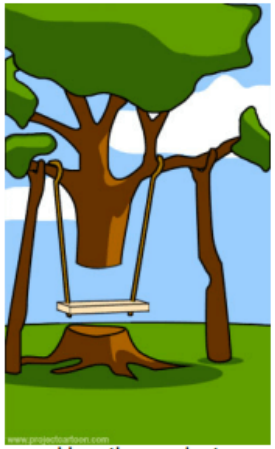
Lecture 13 – Development Processes

Adapted from Kevin Wendt and Praveen Mittal's lecture

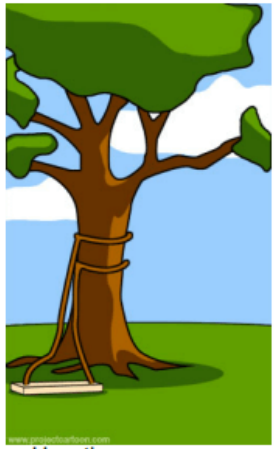
How the customer explained it



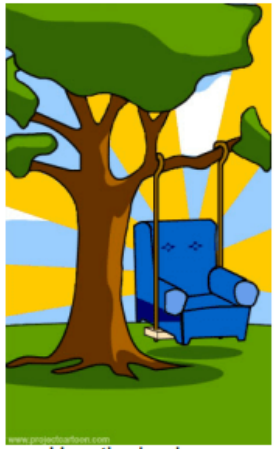
How the project leader understood it



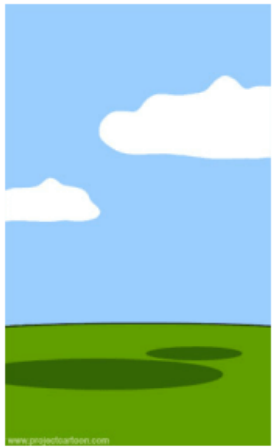
How the analyst designed it



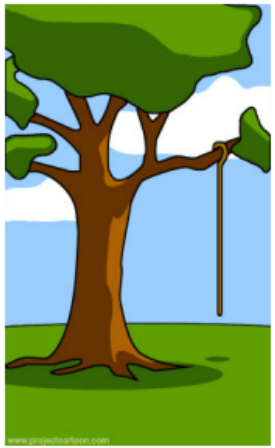
How the programmer wrote it



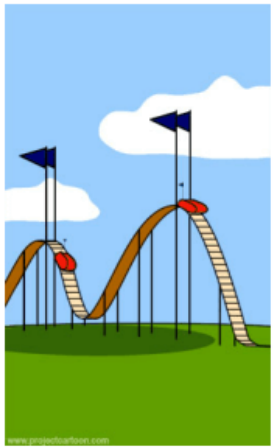
How the business consultant described it



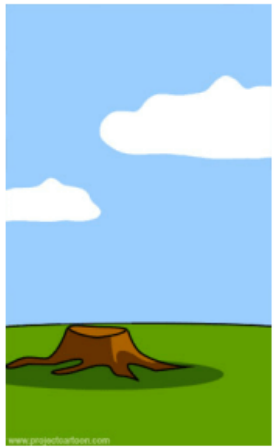
How the project was documented



What operations installed



How the customer was billed



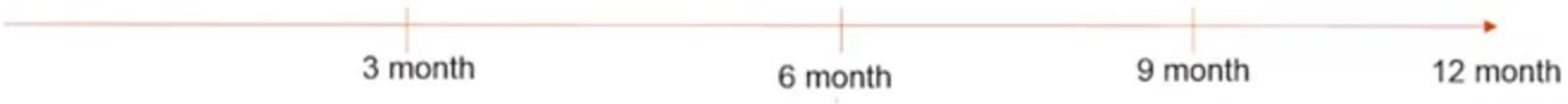
How it was supported

What the customer really needed

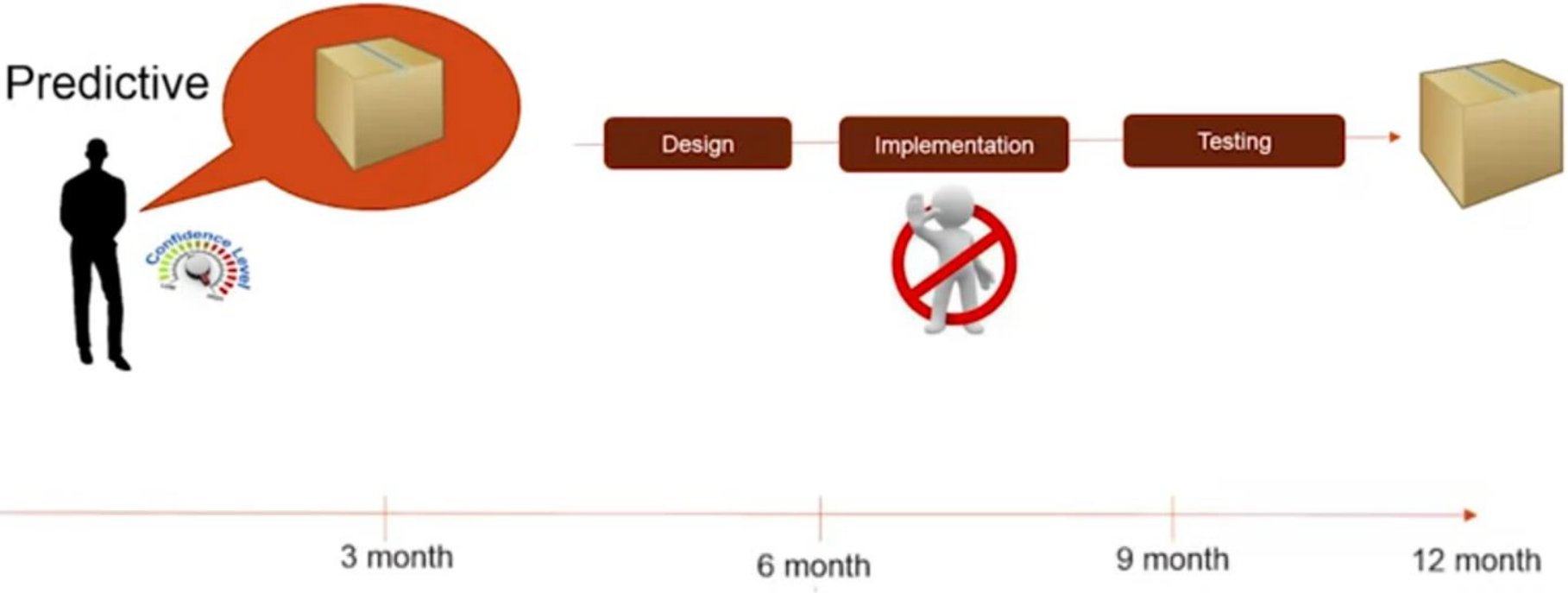
Software Development Models

- We have all the pieces of the system: req, spec, design, implementation, testing
- Piece them together with a model
  - Many models
  - Why use so many? Why not just one?
- Model Classification
  - Predictive vs adaptive
  - Incremental vs iterative vs Both vs None
- Some models
  - Waterfall
  - Sashimi (similar to waterfall)
  - V-model

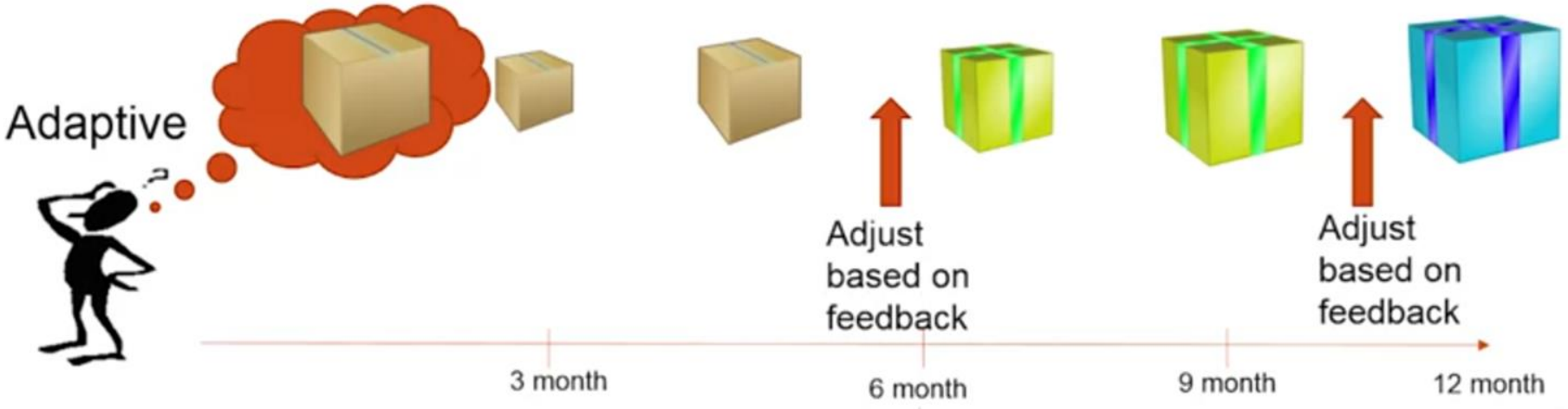
Predictive vs. Adaptive



Predictive vs. Adaptive



Predictive vs. Adaptive



Incremental vs. Iterative

- Incremental

Incremental vs. Iterative

- Iterative

Incremental vs. Iterative

- Incremental
  - Pretty clear idea
  - Build on top of existing system
  - Can be predictive or adaptive
- Iterative
  - Not a super clear idea
  - Sometimes there’s actual replacement

Incremental vs. Iterative vs. Both vs. None

Incremental

Iterative

Incremental vs. Iterative vs. Both vs. None

Both:

Agile is both (we'll learn this later)

Labs 7, 8, 9, and 10 were also both. Incrementally built a system each week. Iteratively had new versions of a system each week.

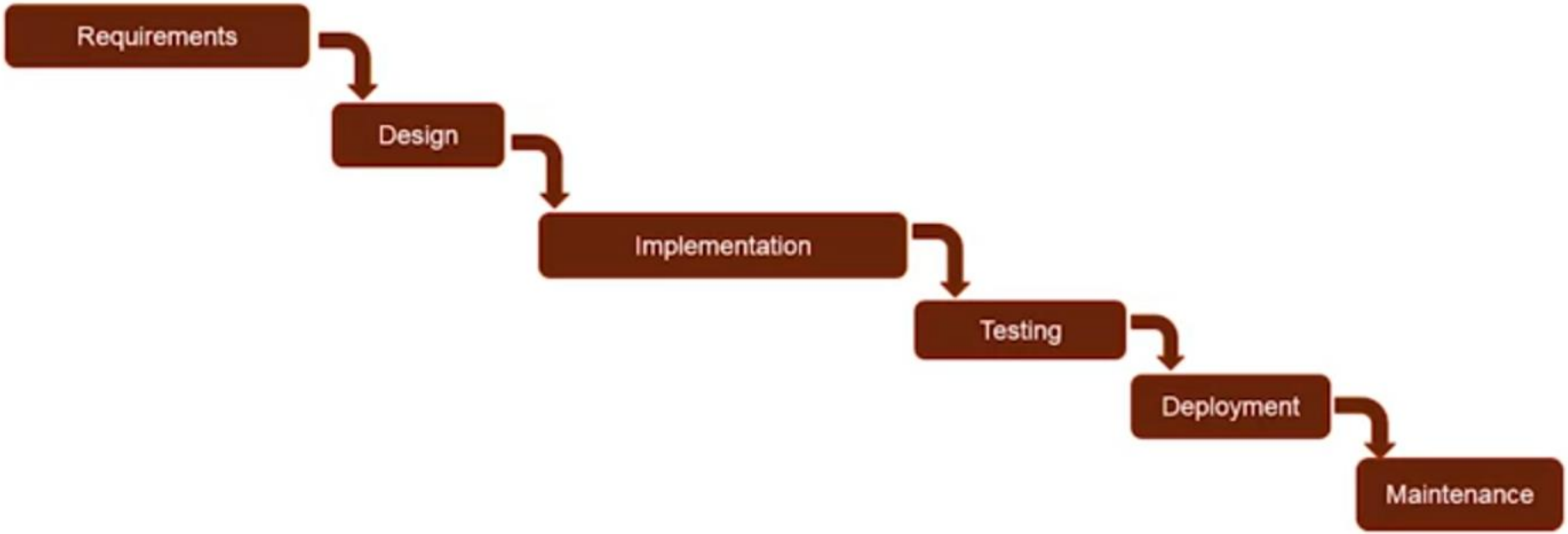
None:

A completely predictive model would be neither. (Personally, I think one could argue that a predictive model could be incremental with one iteration.)

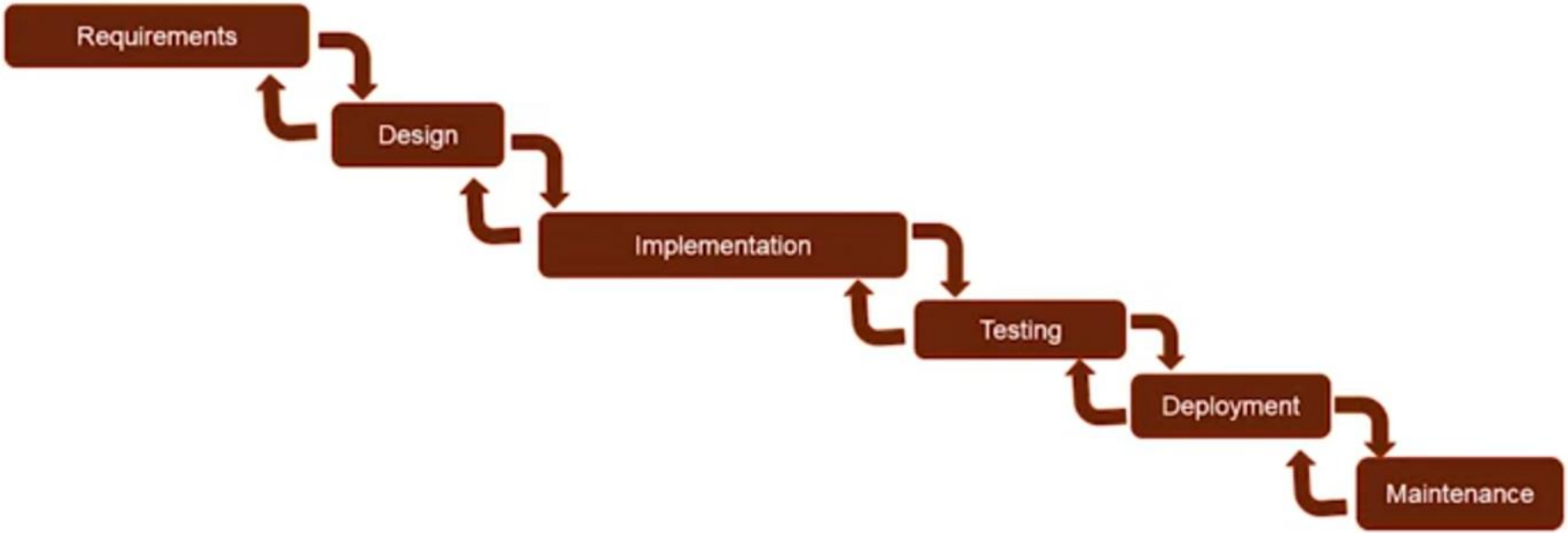
What we will learn about models

- Model name
- Mechanics
- Characteristics
- Predictive vs. adaptive
- Pros vs. cons
- When to use?

Waterfall


Requirements
Design
Implementation
Testing
Deployment
Maintenance

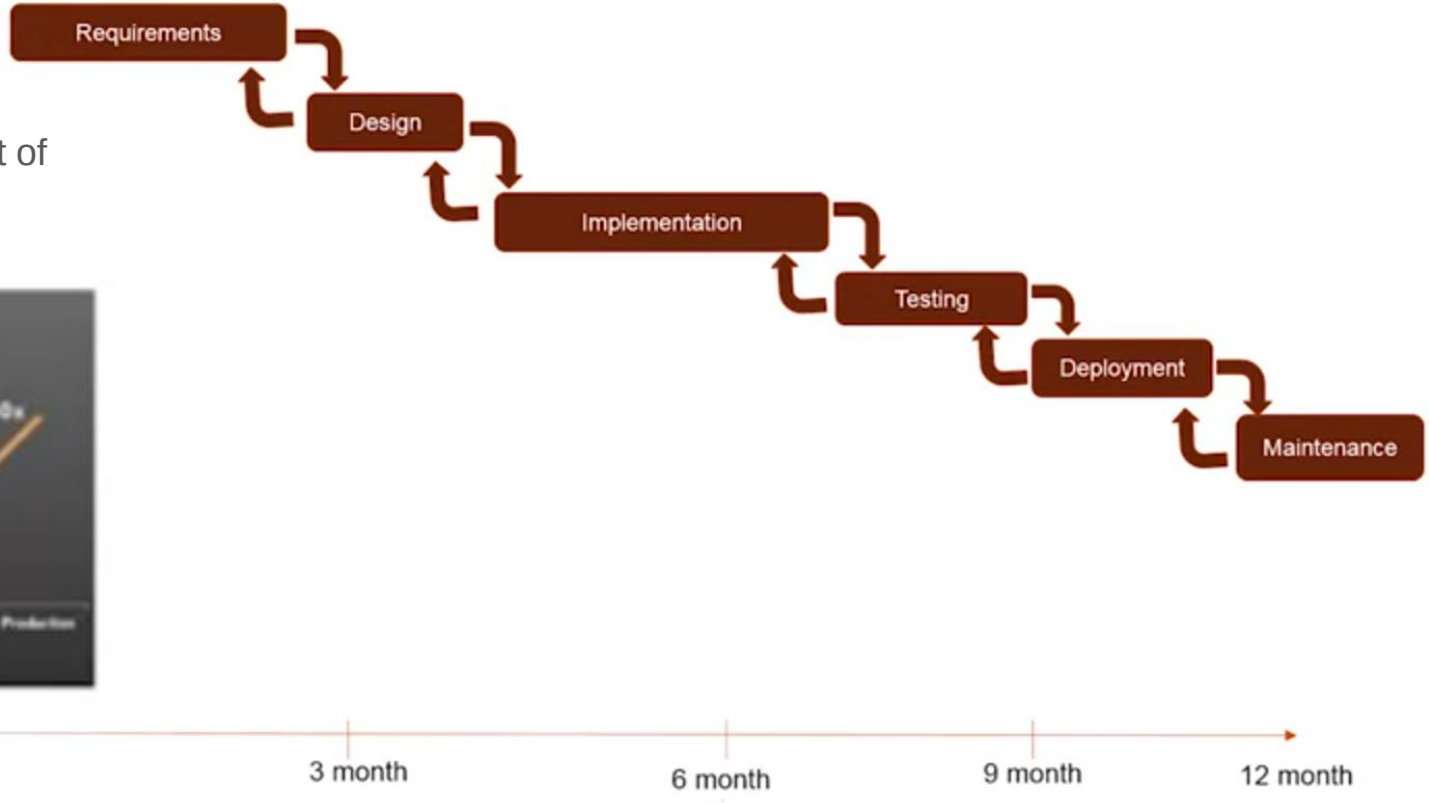
Waterfall


Requirements
Design
Implementation
Testing
Deployment
Maintenance

Waterfall

Cost of change

As time elapses, cost of change increases



Waterfall

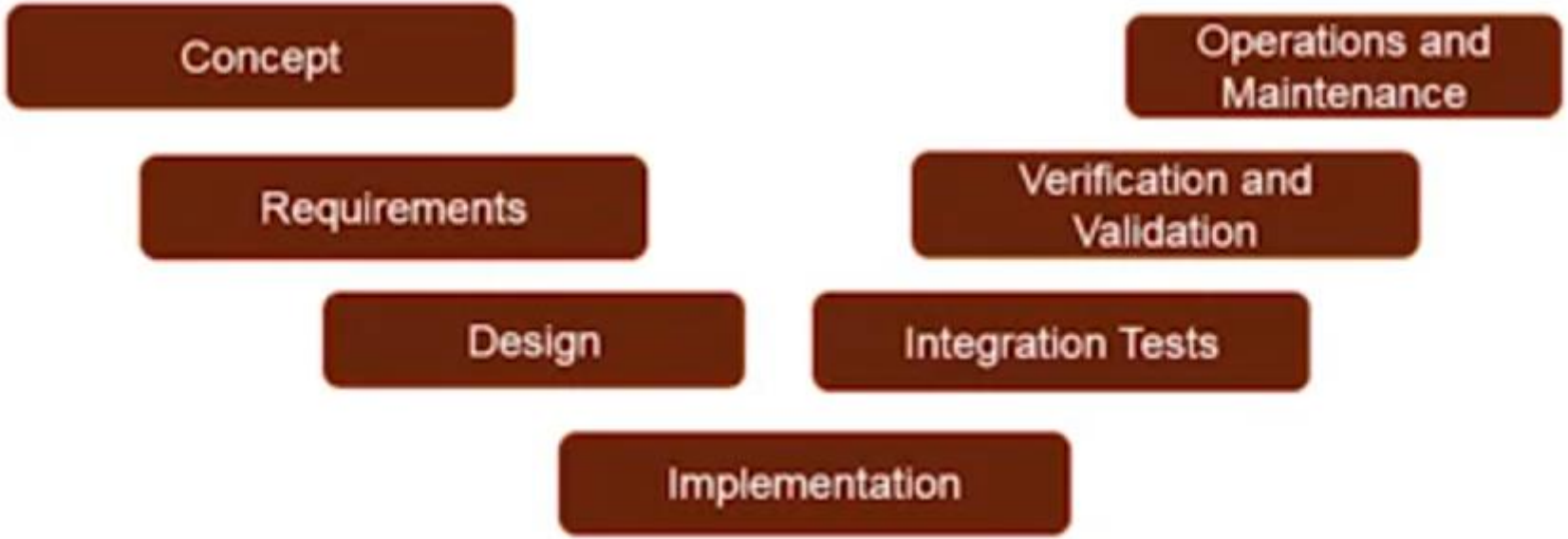
- Assumptions
  - 1. We fully understand the requirements, we were given the requirements with no mistakes in them, the requirements won’t change
  - 2. Team has experience building similar software.
  - 3. The whole process from requirements all the way until the release will be perfect.

- When to use
  - Predictable, repeatable work

Waterfall

- Predictive/Adaptive Scale
  - Completely predictive
- Pros
  - Simple, easy for everyone to understand
  - Predictable
  - Efficient
- Cons
  - Not flexible for change
  - First release happens at the end of the process

V-Model

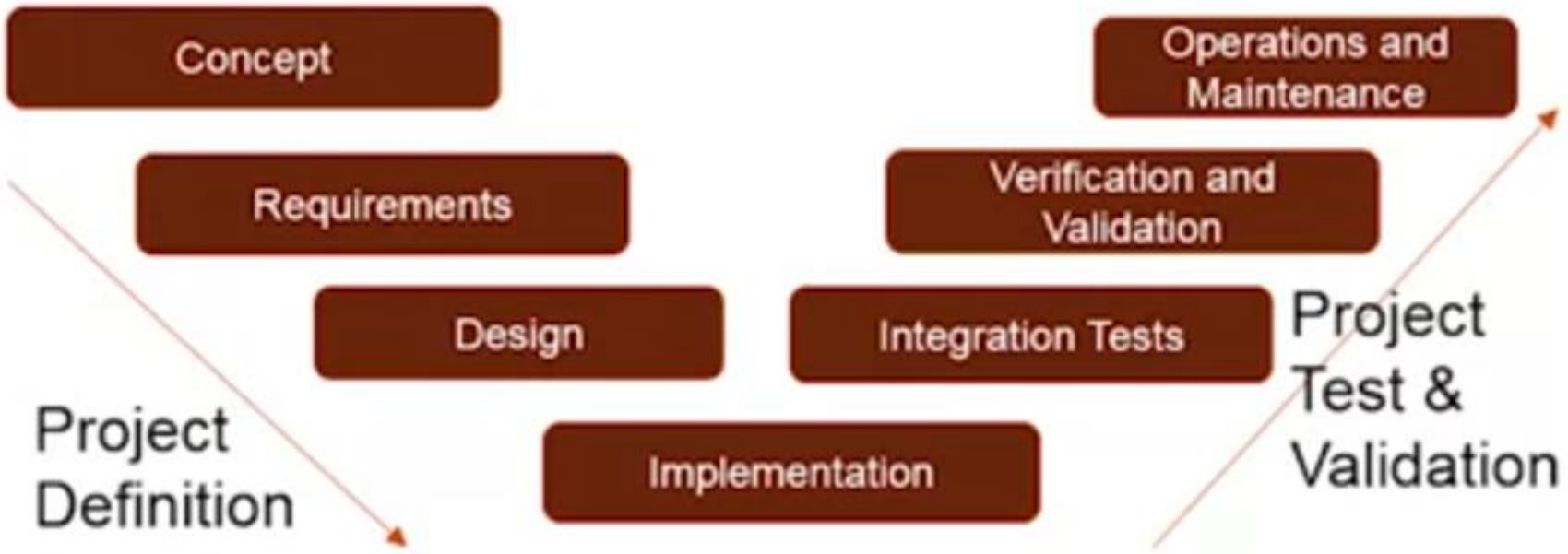

Concept
Operations and Maintenance
Requirements
Verification and Validation
Design
Integration Tests
Implementation

Validation and Verification

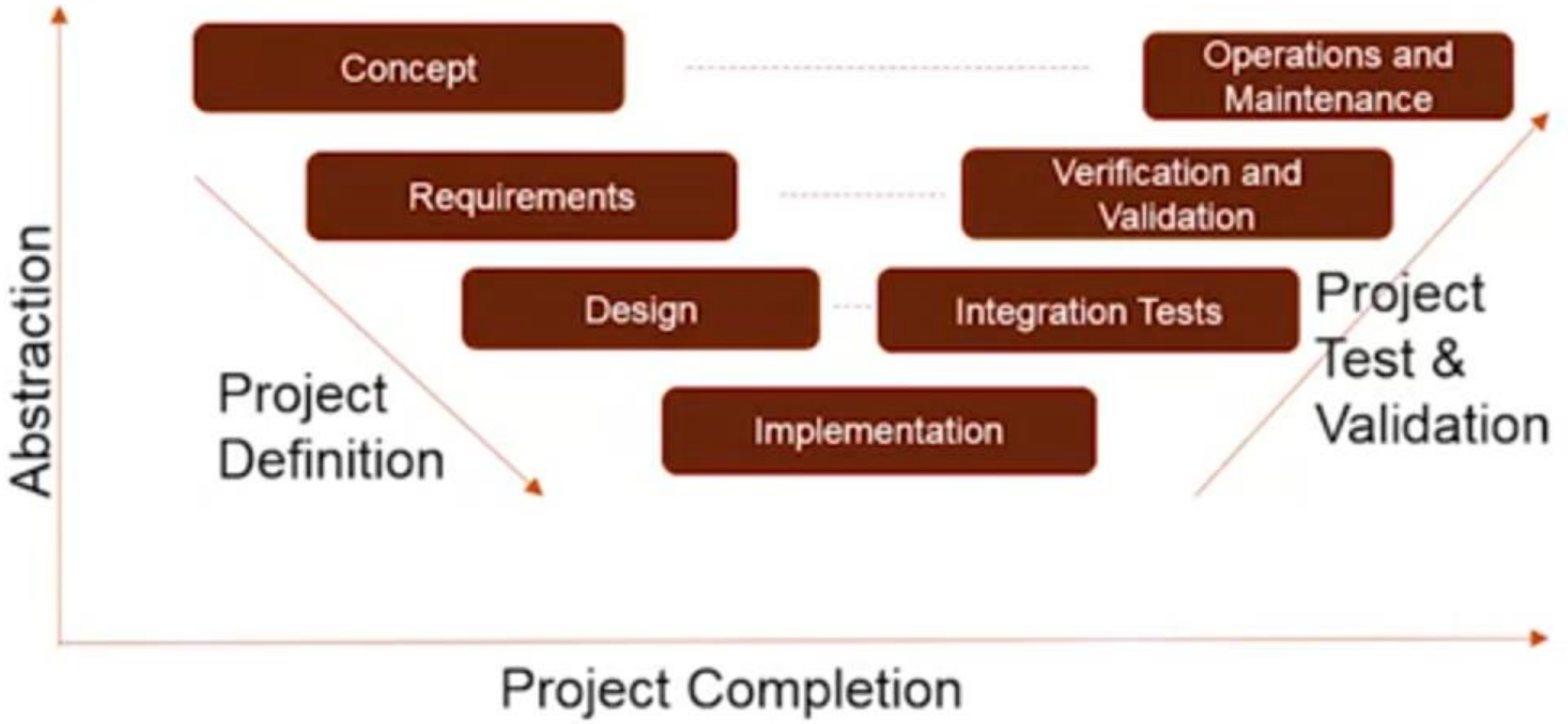
V&V are two procedures which check requirements and specifications

Validation checks requirements - am I building the right thing?
Verification checks specifications - am I building the thing correctly?

V-Model


Concept
Requirements
Design
Implementation
Integration Tests
Verification and Validation
Operations and Maintenance
Project Definition
Project Test & Validation

V-Model


Concept
Operations and Maintenance
Requirements
Verification and Validation
Design
Integration Tests
Implementation
Project Definition
Project Test & Validation
Abstraction
Project Completion

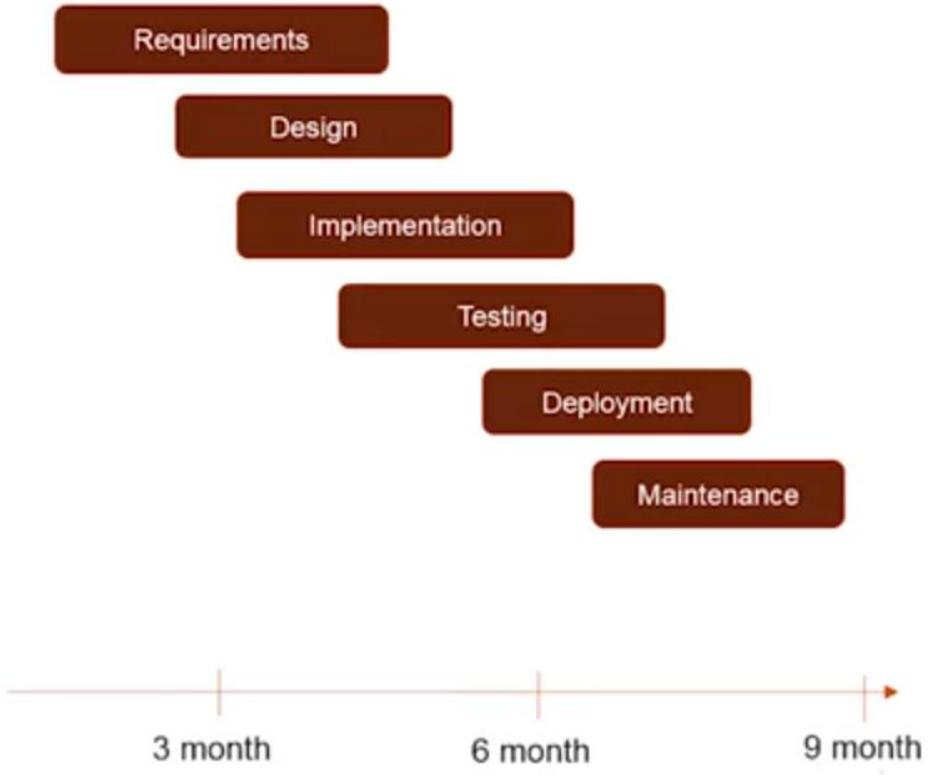
V-Model

- Mechanism – emphasis on validation earlier in the process
- Predictive/Adaptive Scale
  - Predictive – no feedback/changes
- When to use?
  - When there’s some ambiguity in the requirements so having early validation could be useful.

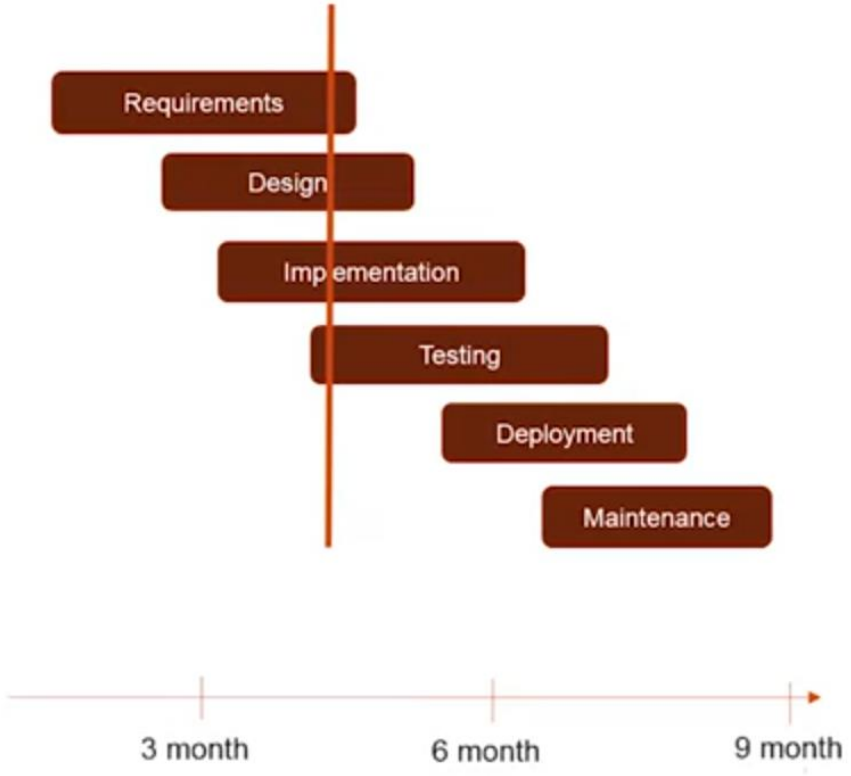
V-Model

- Pros
  - Earlier detection of potential issues
- Cons
  - More upfront work

Sashimi


Requirements
Design
Implementation
Testing
Deployment
Maintenance
3 month
6 month
9 month

Sashimi


Requirements
Design
Implementation
Testing
Deployment
Maintenance
3 month
6 month
9 month

Sashimi

- Mechanism – a phase can start before the previous phase ends
- Predictive/Adaptive Scale
  - Almost completely predictive but maybe a little bit adaptive
- When to use?
  - To shorten the time scale
  - Resource utilization

Sashimi

- Pros
  - Shorten development time
  - People can start working without waiting for the previous phase to finish
  - Can learn more easily
- Cons
  - Reworks

Software Development Memes

Project Management Textbook (optional)

Software Project Management: A Unified Framework

Hardcover is $5.78
Used Hardcover is $0.88

Project Management Textbook (optional)

Requirements analysis and evolution activities consume 40% of life cycle costs

Software design activities have an impact on more than 50% of the resources

Coding and unit testing activities consume about 50% of software development effort and schedule

Test activities can consume as much as 50% of a project’s resources

Configuration control and change management are critical activities that can consume as much as 25% of resources on a large-scale project

Documentation activities can consume more than 30% of project engineering resources

Project management, business administration, and progress assessment can consume as much as 30% of project budgets

Project Management Memes

Project Management Memes