EECE 310

Software Process Models

When things go wrong

- Denver airport
- HealthCare.gov
- UBC Connect!

Why do projects fail?

- Unrealistic project **goals**
- Inaccurate **estimates** of needed resources
- Badly defined system **requirements**
- Unmanaged **risks**
- Poor **communication**
- Poor project **management**
- Stakeholder politics / pressure
- Improper testing
- ….

Software process

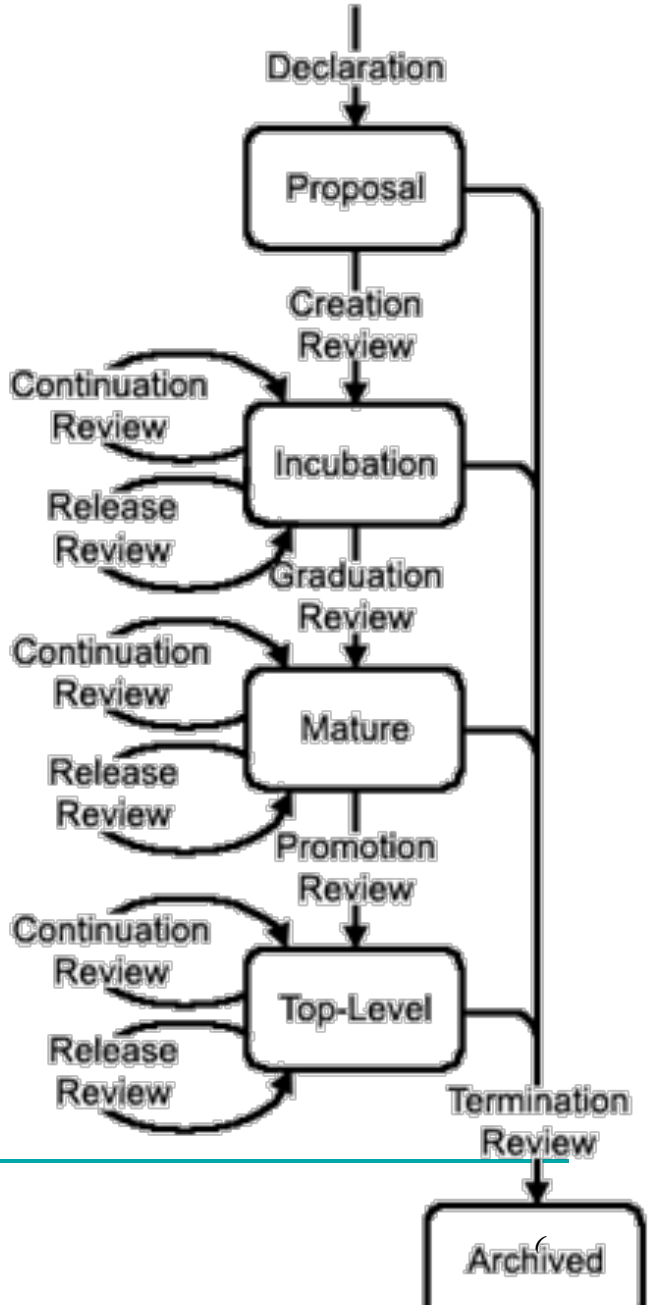
- A software process is a structured set of *repeatable* activities to develop a software system.
- Defines **who** is doing what, **when** and **how** to reach a goal.

Software process

- Process descriptions also include:
  - **Products**, which are the outcomes of a process activity;
  - **Stakeholders**: people who care (about the outcome)
    - Managers
    - Developers
    - End Users/Customers
    - Testers
    - Architects
    - ...

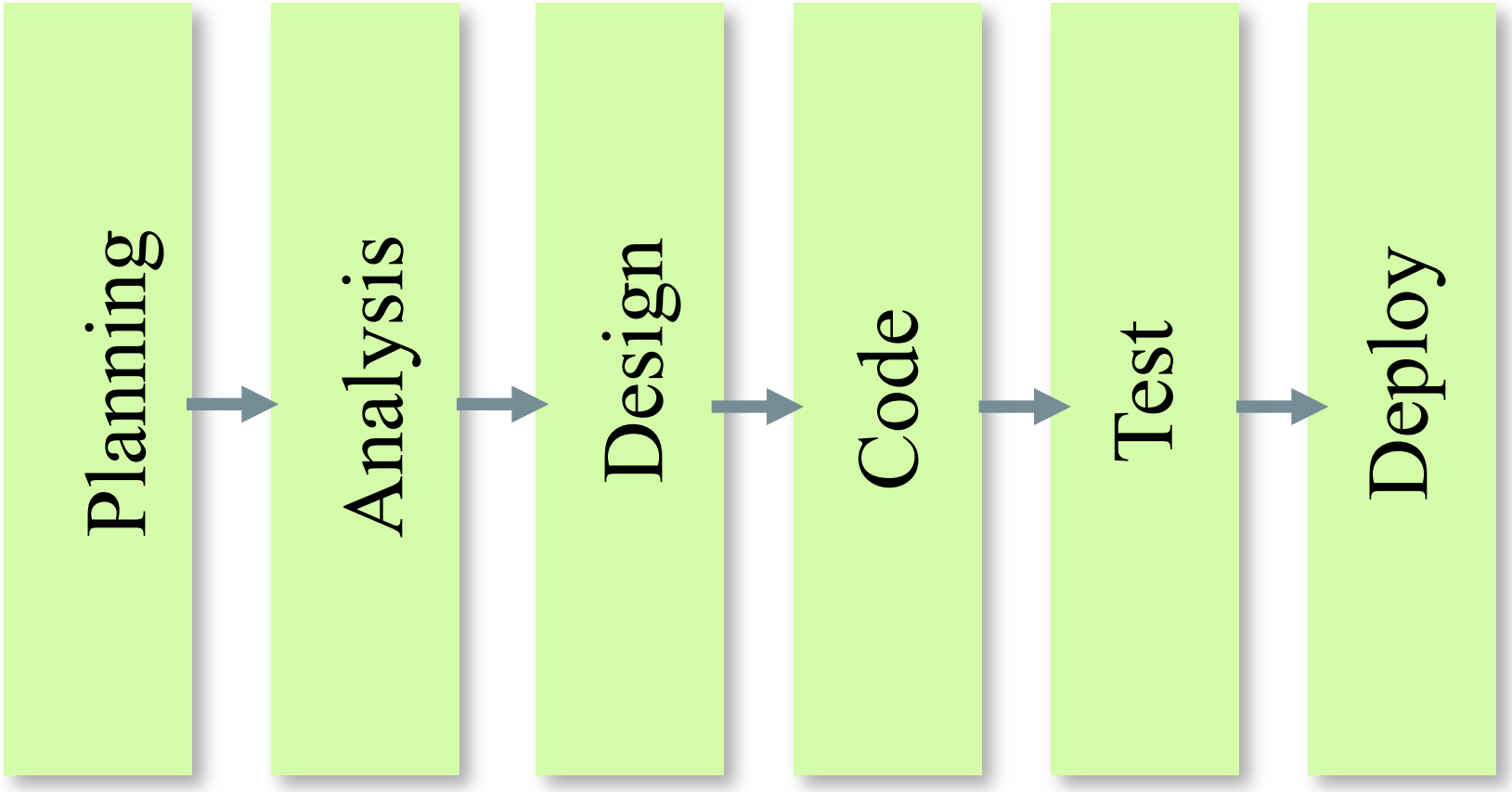
The Eclipse dev. process

- Eclipse consists of “Projects”
- Each project does its own internal process
- Eclipse community has “simultaneous releases” e.g. Ganymede, Helios, etc.

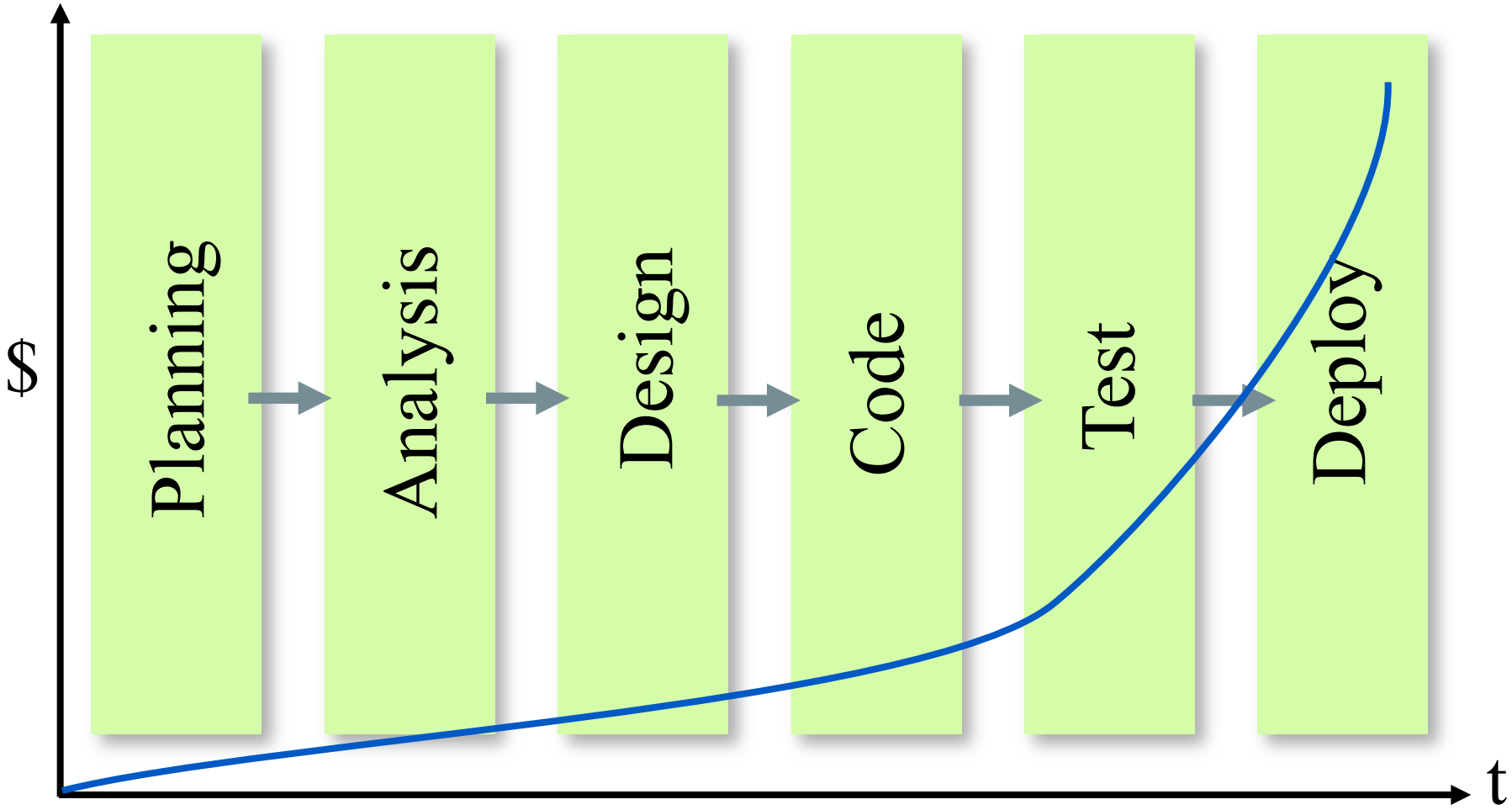


http://www.eclipse.org/projects/dev_process/development_process_2011.php#6_4_Releases

Stages in software development?

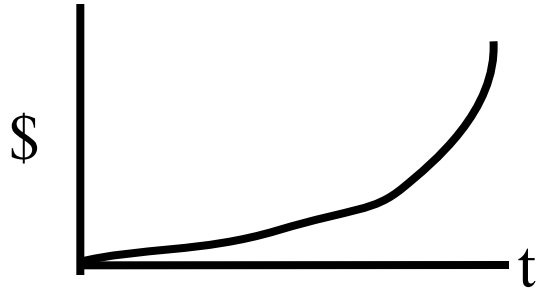


The software change cost curve



Key Insight: Negative feedback loop

We fear this:



So we insist on this:

Which leads to this!

Developers in the process

- Process creates specialized roles
  - "Developer" is too coarse-grained
    - Project manager
    - Business Analyst (Requirements)
    - Software Architect (Design)
    - Software Developer (Implementation, Maintenance)
    - Software Tester (Verification, Debugging)
    - Software Intern (Coffee, code review)

Software processes

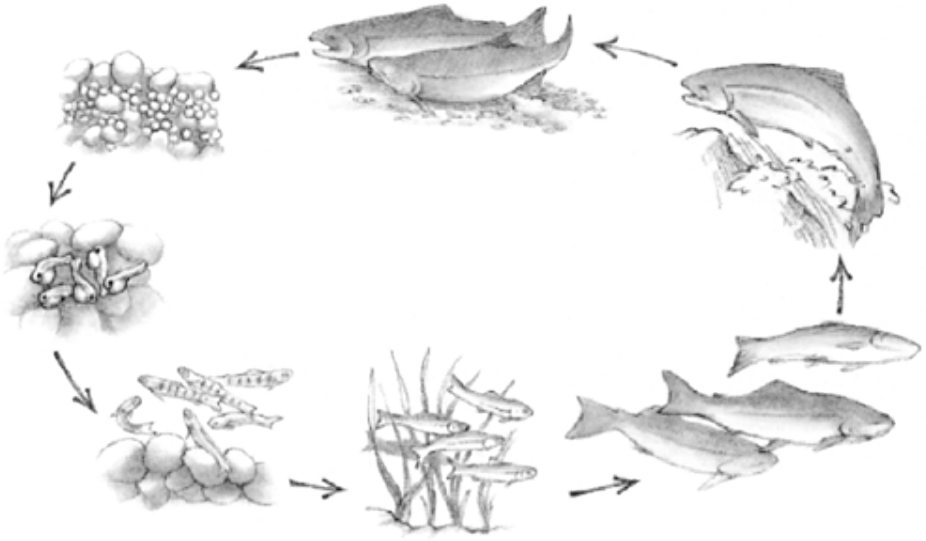
- Many different software processes, all include:
  - requirements elicitation,
  - system specification,
  - design,
  - implementation,
  - integration,
  - testing,
  - deployment, …
- Goals of each activity
  - Mark out clear set of steps to perform
  - Produce tangible item(s)
  - Allow for review of work
  - Specify actions to perform in the next activity

Benefits of a software process

- provides an organizational tool: activities cannot be forgotten
- provides a large-scale shared framework in which to work
- facilitates necessary communication
- forces us to break down the problem
- provides a management tool

Software life cycle

- Series of phases in the life of a software system.



- Each phase can go through several iterations

Phases vs. activities

- Activities performed concurrently, but with differing intensity

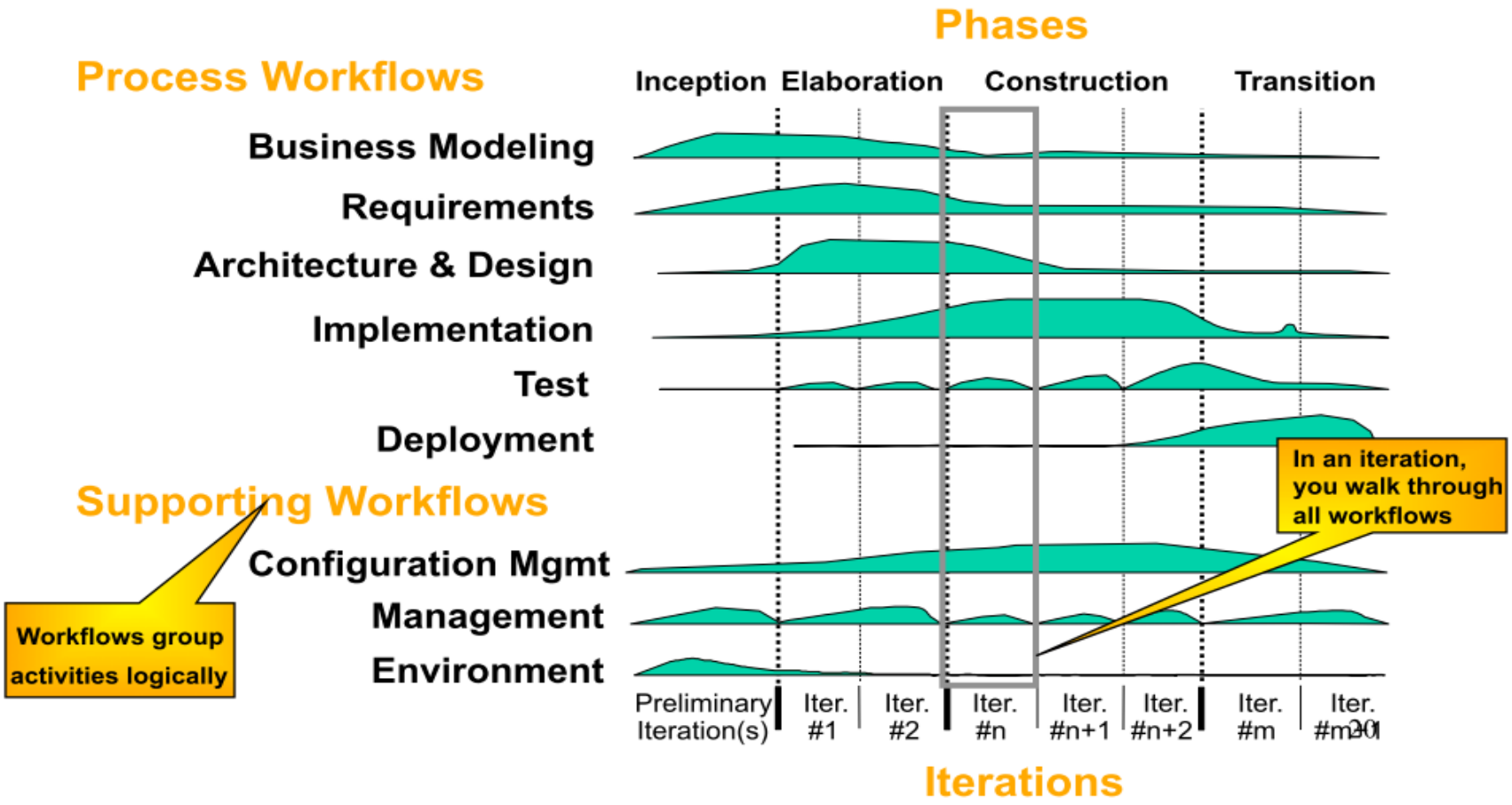


image from Maibaum and www.upedu.org

Why should you care?

- To work as a software developer
  - Know the terminology, don't be lost
  - Understand the purpose
  - Understand your role/task
  - Less frustration
- To lead a project as a software engineer
  - Assess the risks
  - Choose *the right* process model
  - Have a successful project!

Software process *models*

- A software process model is an **abstract representation** of a software process.
- Many different types
- No silver bullet, each good for different situations
- Like different cars
  - Truck to carry goods
  - Bus to transport groups of people
  - Volvo for safety
  - Ferrari for speed (and showing off)

Software process models

- Waterfall (sequential)
  - separate and distinct phases of specification and development
- Spiral
  - spiral cycles
  - risk management at each stage
- Agile (iterative)
  - incremental/iterative approach
  - short cycles with tangible outcomes
- …

Software process models

- In practice, most large systems are developed using a process that incorporates elements from all of these models.

- Different models for different
  - types of software
  - types of companies
  - types of management

- There are no right or wrong software processes.

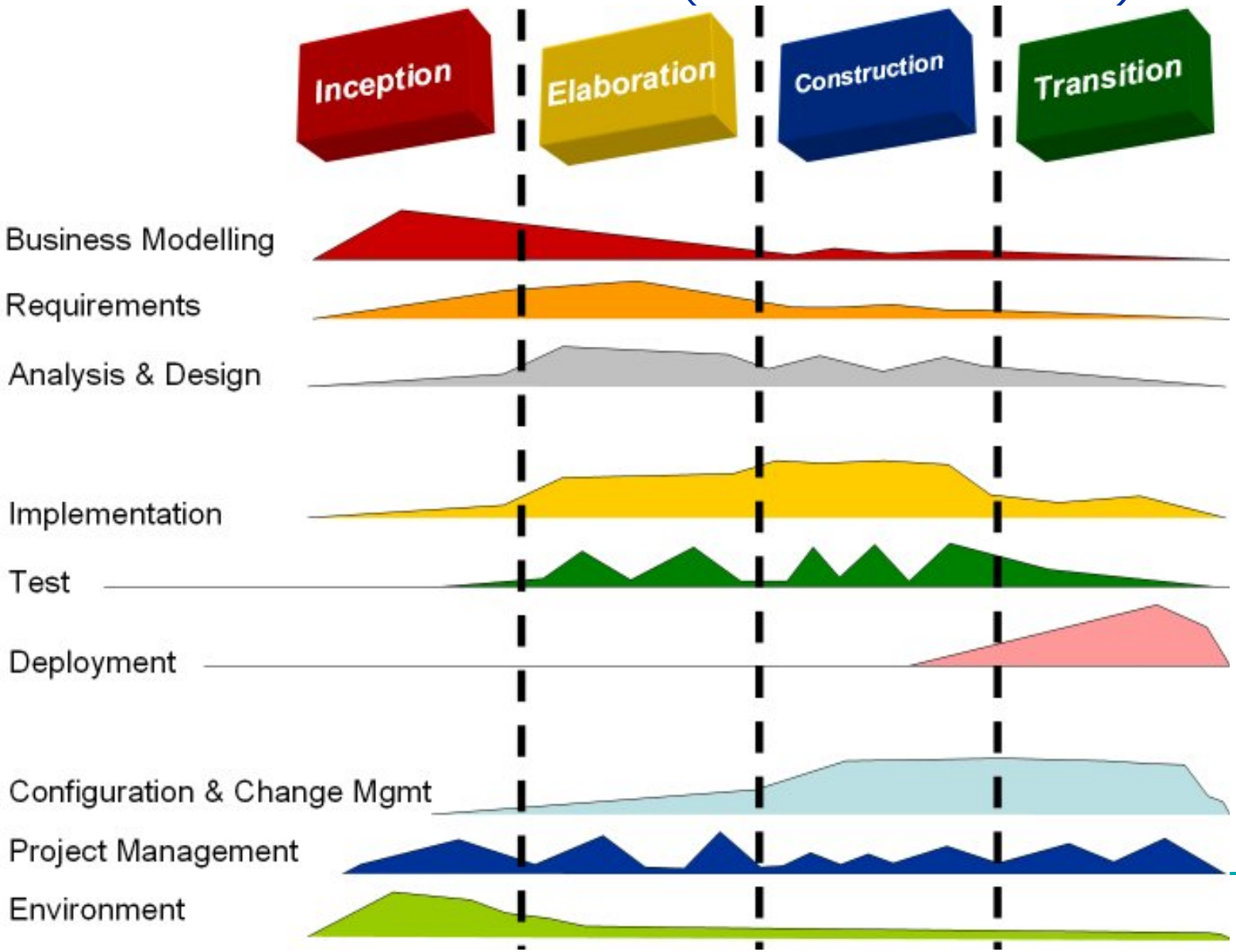
Examples:

- **RUP** (Rational Unified Process)
  - is a framework for large organizations and teams

  while

- **Scrum**
  - is intended for a **product team** with stringent guidelines.

Phases and activities (from R.U.P.)

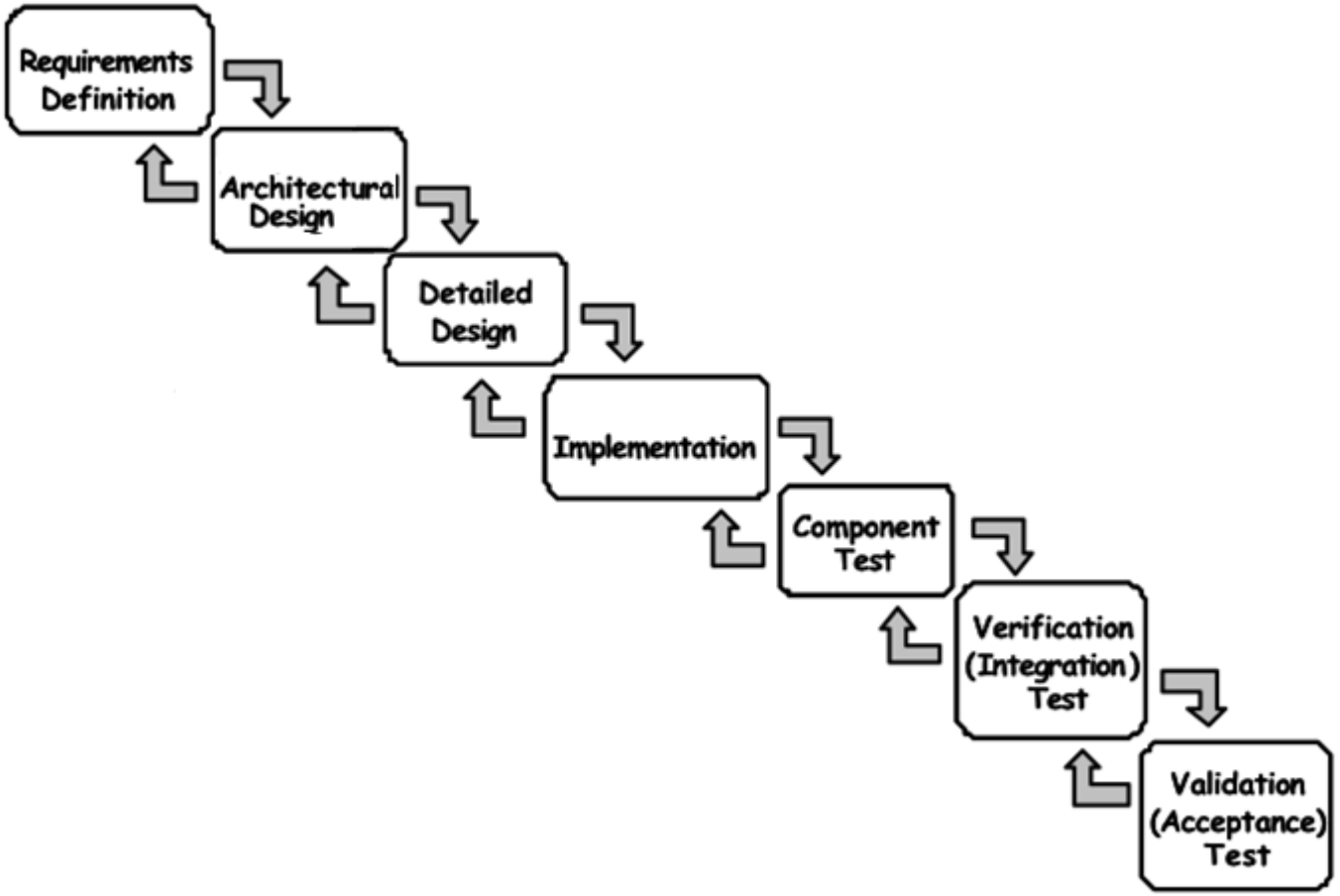


Sequential process model

- Waterfall

image by Earthwatcher on flickr.com

Waterfall model



(a.k.a. BDUF: Big Design Up Front)

Royce, 1970: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

Waterfall: Advantages

- Good for well-understood but complex projects
  - Tackles all planning up front
  - No midstream changes = efficient process
- Provides support for an inexperienced team
  - Orderly, sequential, easy-to-follow model
  - Relatively slow progress
  - Reviews at each stage

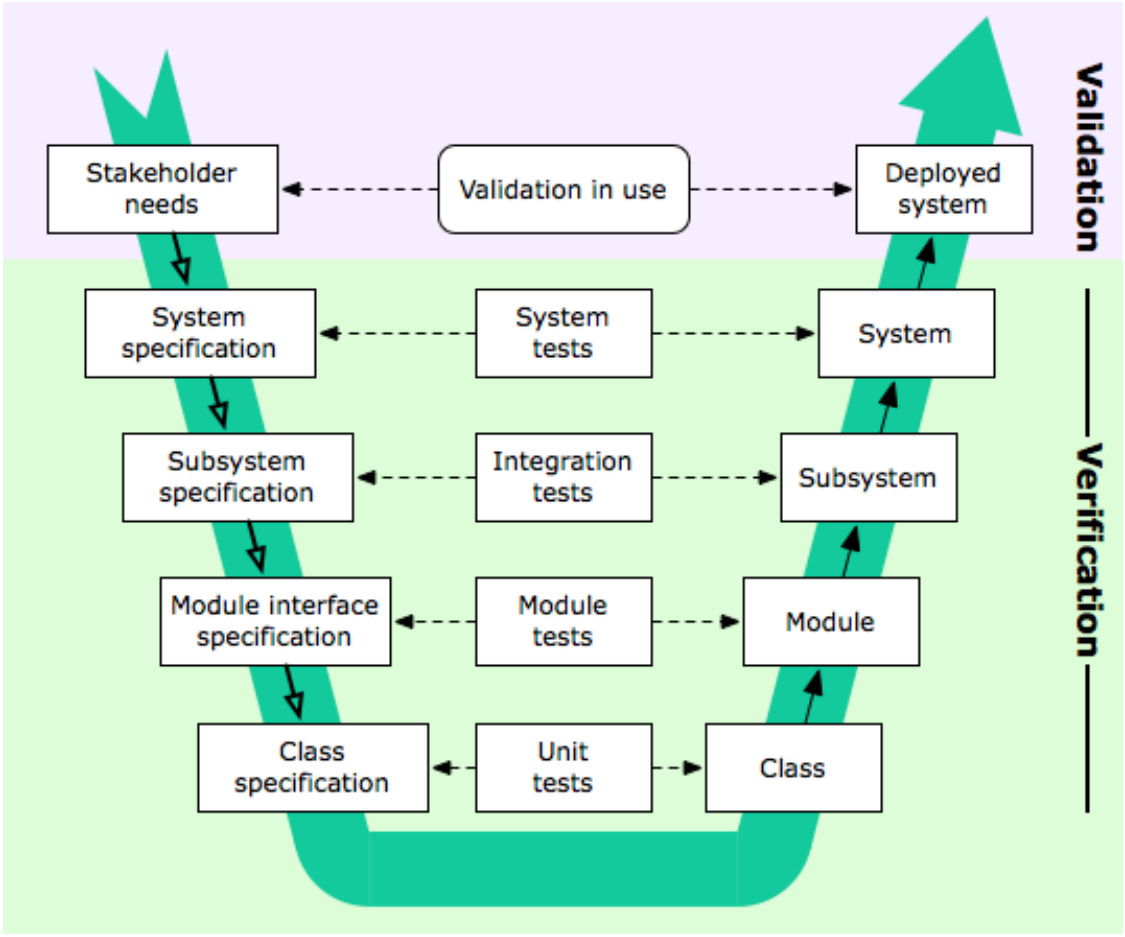
Waterfall: Drawbacks

- Getting requirements right up front (not always possible)
- Relatively heavy-weight (reviews are massive affairs)
- No integration until the end (no incremental process)
- Resistant to change
- Final result not necessarily client-driven

V-Model

- An extension of Waterfall
- Process steps are bent upwards after the coding phase (like a V)
- Emphasis on testing: tests should be developed at each phase

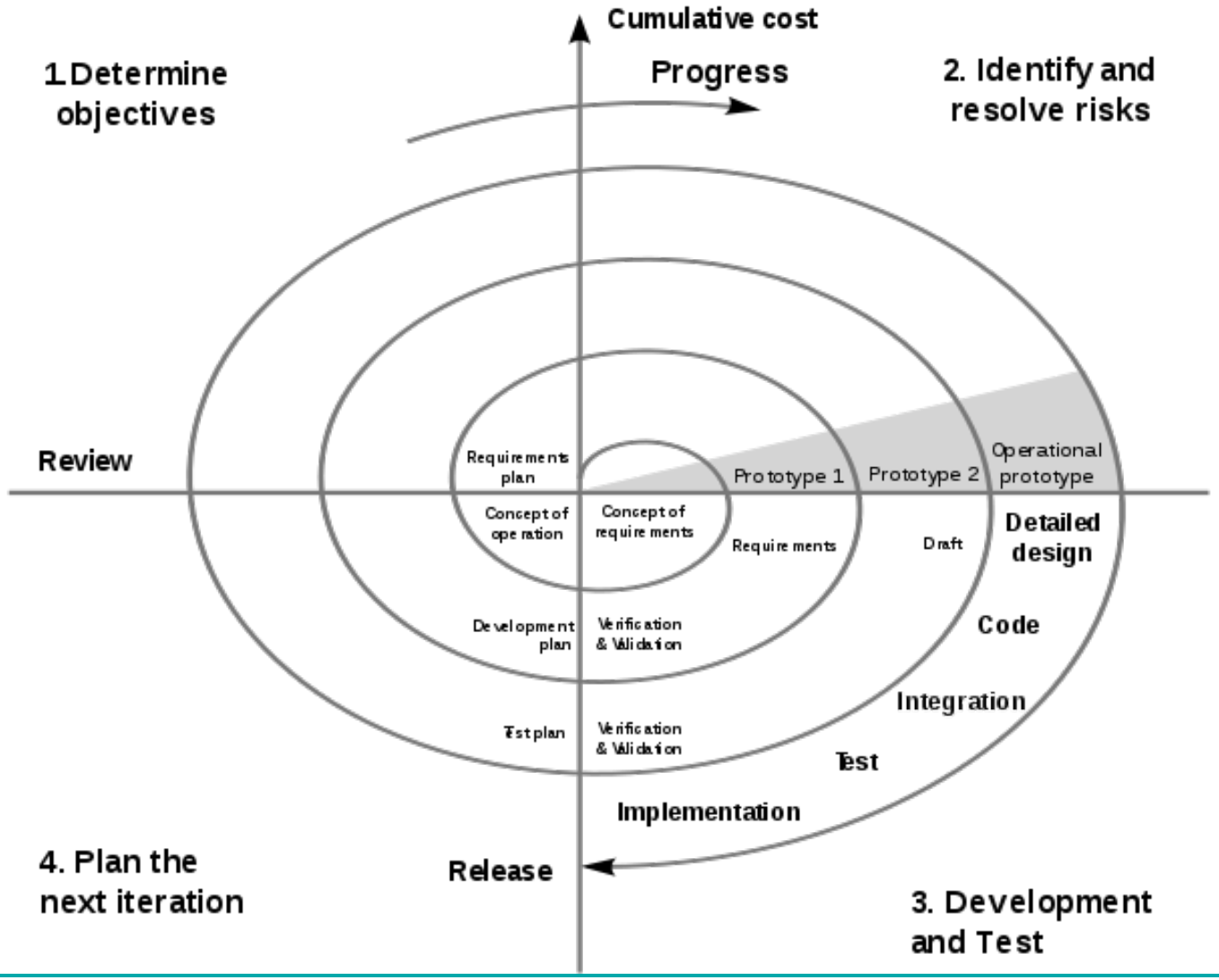
V-Model


Stakeholder needs
Validation in use
Deployed system
Validation
System specification
System tests
System
Subsystem specification
Integration tests
Subsystem
Module interface specification
Module tests
Module
Class specification
Unit tests
Class
Verification
Key
A
B
B elaborates the means by which A can be satisfied
S <-- T --> C
If C passes T, then C meets S sufficiently well
A
B
B must meet its specification before A can

Spiral model

- An iterative “waterfall” with iterative releases
- Iterations fairly lengthy (e.g., 2 years)
- Intended for larger projects (> $1m)
- Emphasis on **risk** analysis
- E.g., develop underlying architecture, then add end-user features.

Spiral Model



Boehm B, "A Spiral Model of Software Development and Enhancement", ACM SIGSOFT Software Engineering Notes", "ACM", 11(4):14-24, August 1986

Coping with change

- Change is inevitable in large software projects
  - Business changes
  - New technology or platforms or APIs
  - Changing requirements
  - New management
- Change leads to rework or new functionality
- Design the process so that changes can be accommodated at relatively low cost

Agile Models / Principles

- The goal of agility: develop software in the **face of changing environment** and **constrained resources**
- **Incremental** and **iterative**
  - Development/delivery broken down into increments (parts of required functionality)
  - Requirements are prioritized and highest priority requirements are included in early increments.
- **self-organizing** cross-functional **teams**
- More a set of principles than a fixed model; many variations of agile processes

Waterfall: battleship, protected against everything that might happen…

Agile: speedboat, can change direction very quickly, is fast

In-class exercise

1. What software process model would best fit the Airport Luggage System?
2. How about UBC Connect?
3. Provide reasons for each.

In-class exercise

1. What software process model would best fit the Airport Luggage System?
    1. Waterfall or Spiral
2. How about UBC Connect?
    1. Agile
3. Provide reasons for each.