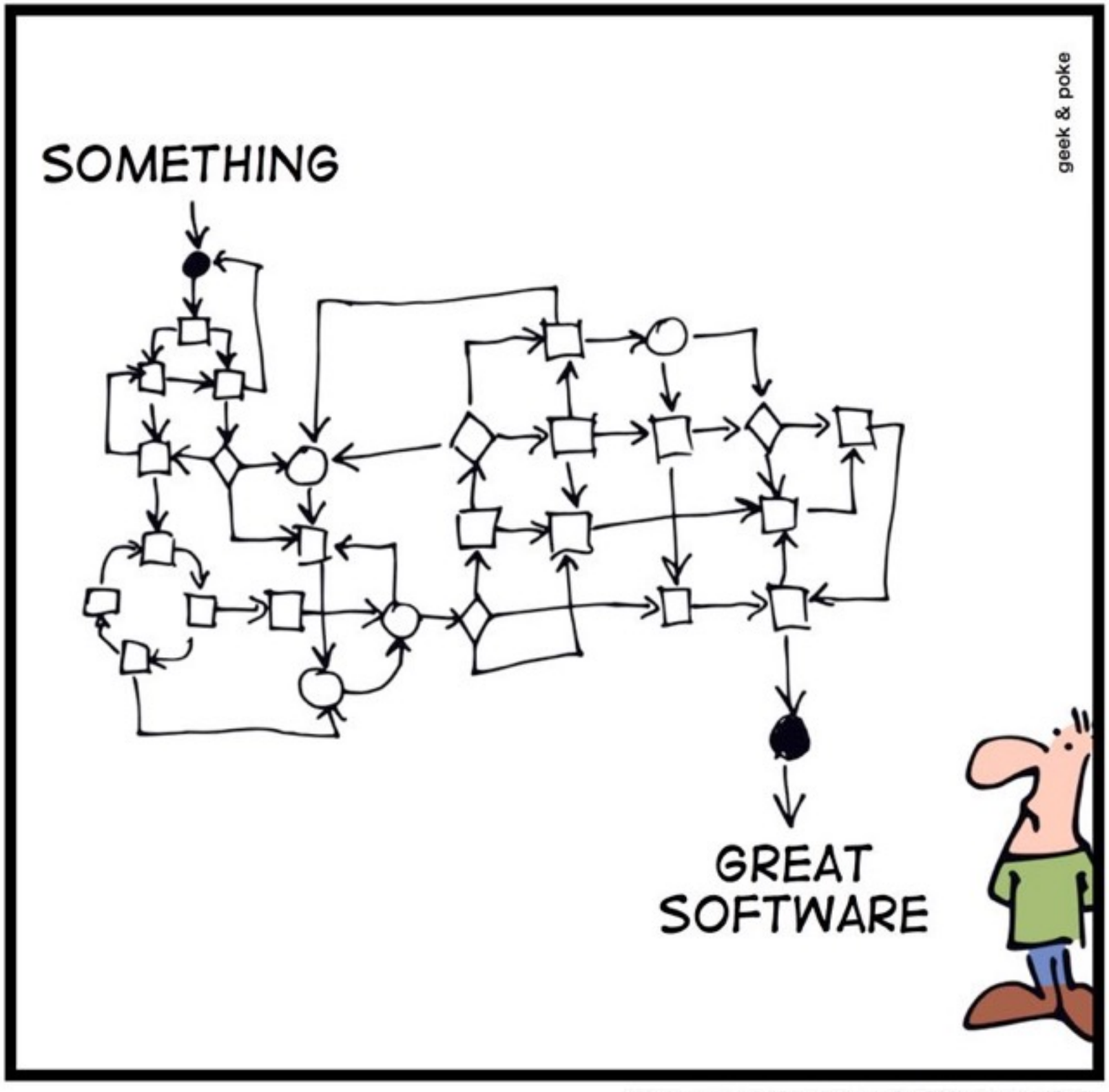
The Software Life Cycle

Software Engineering
Andreas Zeller • Saarland University

SIMPLY EXPLAINED

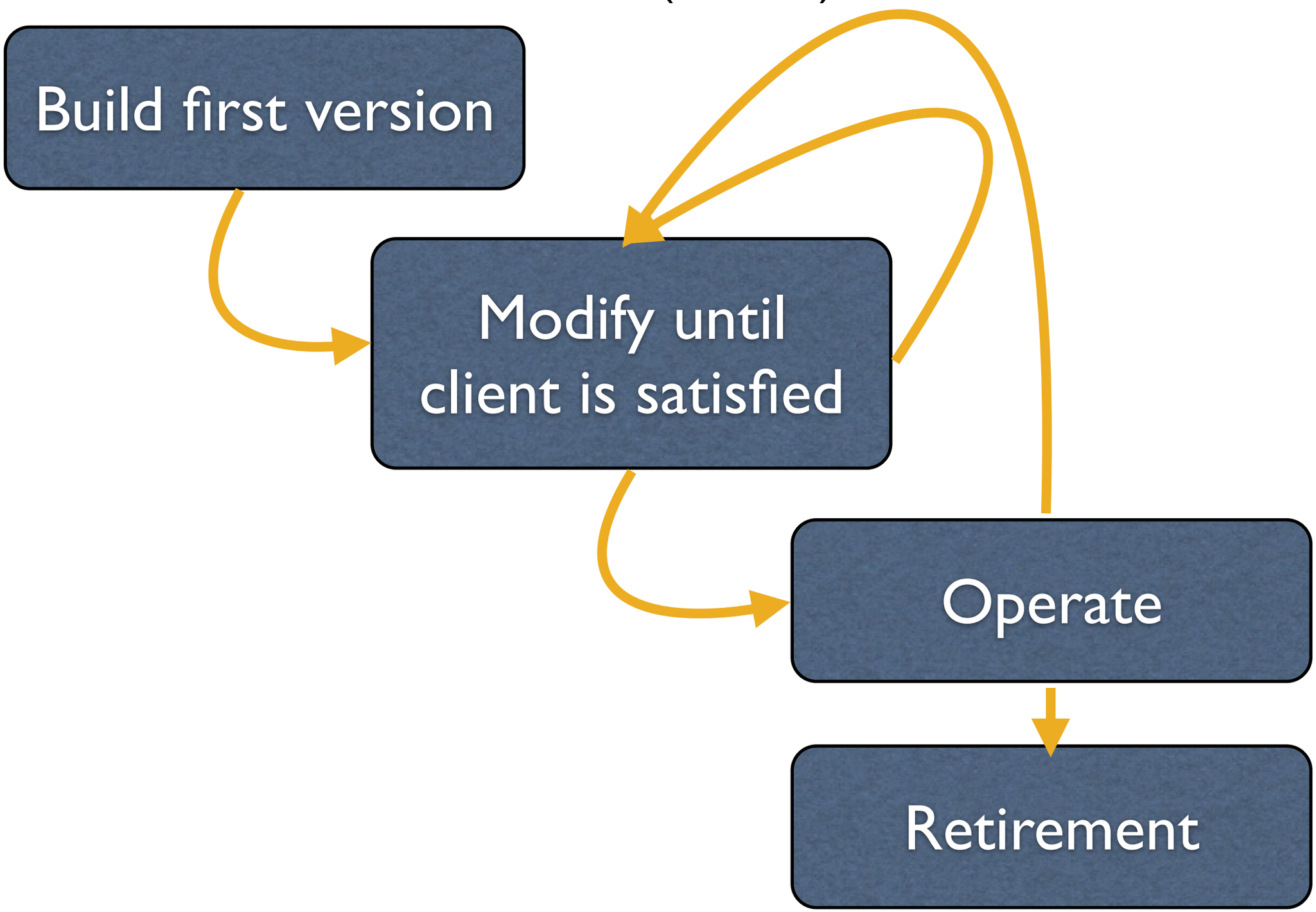


DEVELOPMENT PROCESS

A Software Crisis

Code and Fix

(1950–)



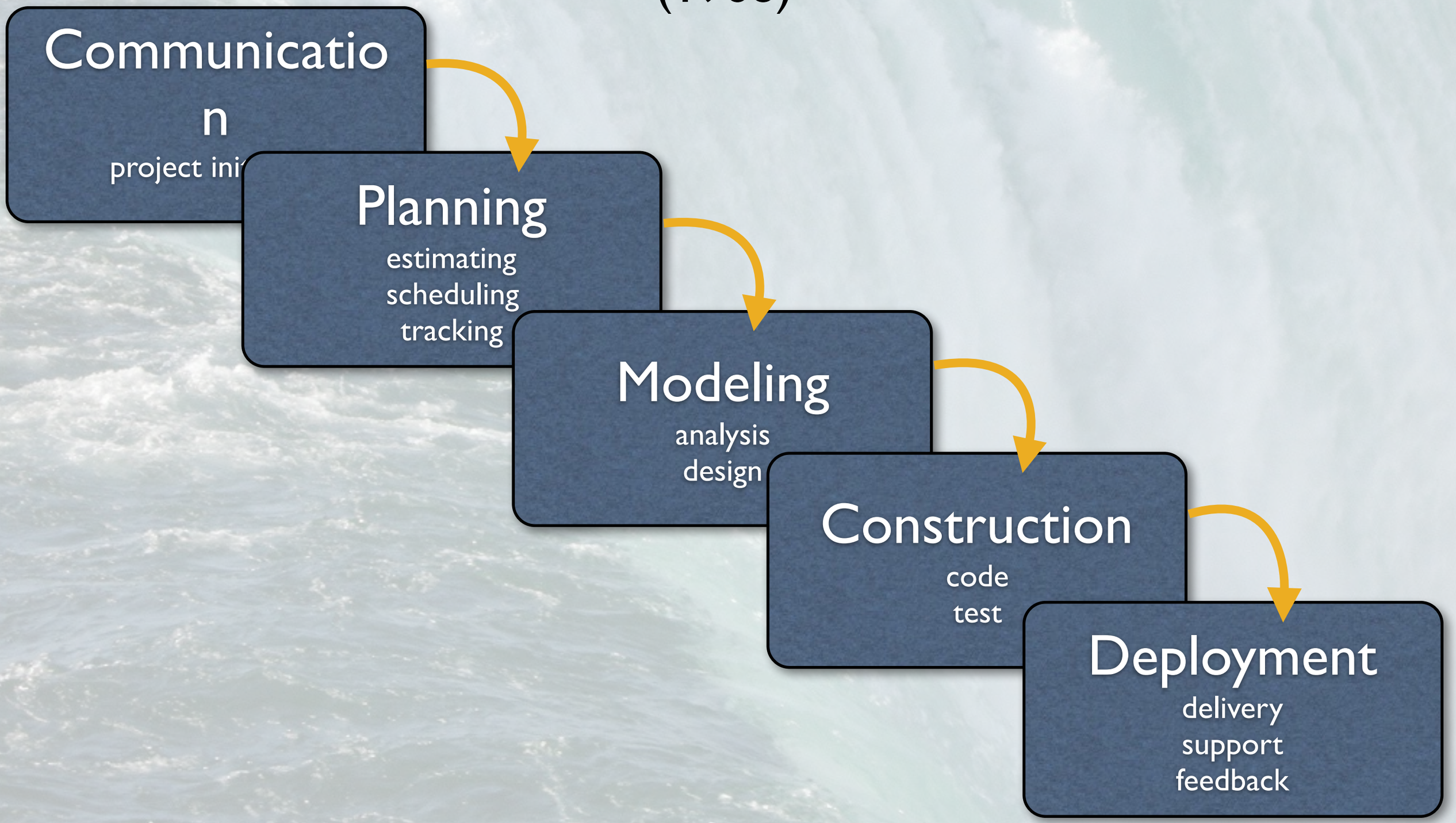
Code and Fix: Issues

- No process steps – no specs, docs, tests…
- No separation of concerns – no teamwork
- No way to deal with complexity

Code and Fix

Waterfall Model

(1968)



Communication

Communication

project initiation

6.6 *Map Series Tool*

Use Case Description	
Summary	User generates one or more maps from a series of maps for a given boundary feature (compartment, landscape etc).
Actors	EIMS User
Pre-Conditions	User requires one or more maps sheets from a series, for a boundary feature.
Post-Conditions	Map or series of maps is generated and printed
Priority	Required

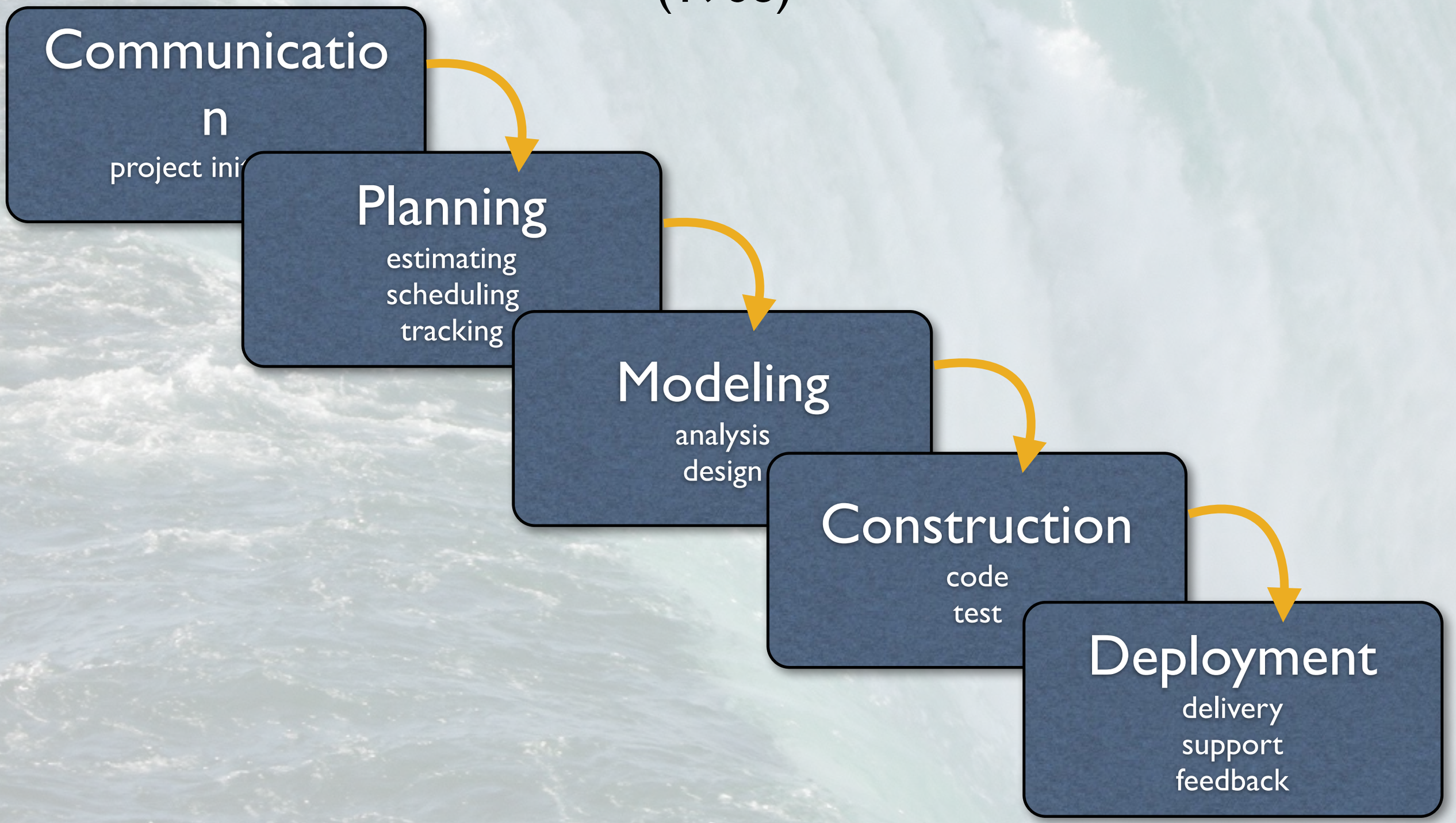
Scenario
1) User starts the tool.
System displays a list of map series that the user can select from. Default map series will be 'Landscape 1:7920'. Can be set at any scale.
2) User selects map series on form.
System then determines if any boundary features are selected. *A. Features Selected:* i. *If features are selected, it asks the user to if they want to generate a map series for the selected feature. Only one feature can used at a time.* *B. No Features Selected:* i. *If no features are selected, or user opts to select the feature manually, the system prompts the user to select the district and compartment of interest from pull downs. It then zooms to that location, generates the map sheet boundaries, draws them with the map sheet names.*
3) User can select individual sheets on screen, or select to print just an index map, or the entire series.
System starts generating and printing maps based on the selected sheets.
4) User collects maps from printer

Notes

Deployment
Tool in ArcMap and in ArcGIS Server

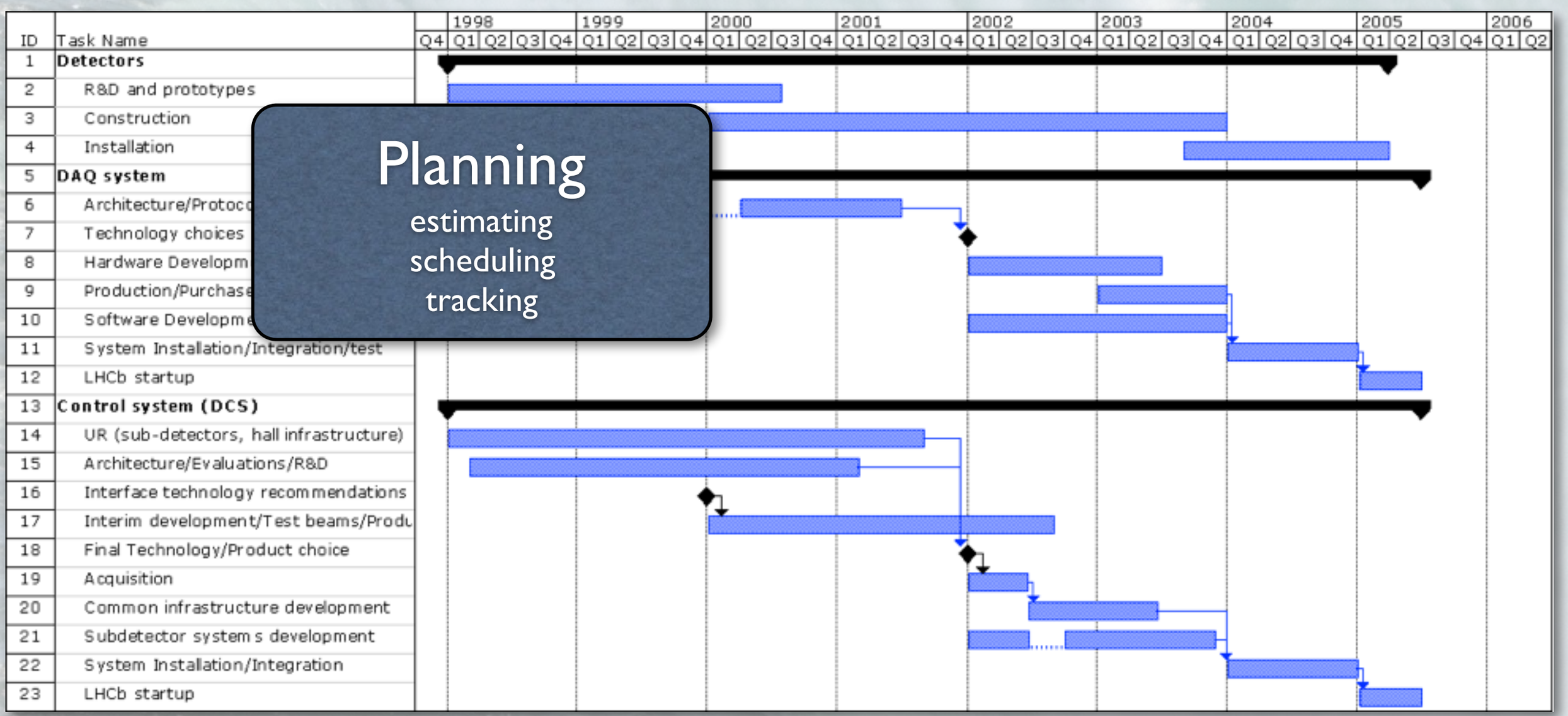
Waterfall Model

(1968)



Planning

ID	Task Name
1	**Detectors**
2	R&D and prototypes
3	Construction
4	Installation
5	**DAQ system**
6	Architecture/Protocc
7	Technology choices
8	Hardware Developm
9	Production/Purchase
10	Software Developme
11	System Installation/Integration/test
12	LHCb startup
13	**Control system (DCS)**
14	UR (sub-detectors, hall infrastructure)
15	Architecture/Evaluations/R&D
16	Interface technology recommendations
17	Interim development/Test beams/Produ
18	Final Technology/Product choice
19	Acquisition
20	Common infrastructure development
21	Subdetector systems development
22	System Installation/Integration
23	LHCb startup

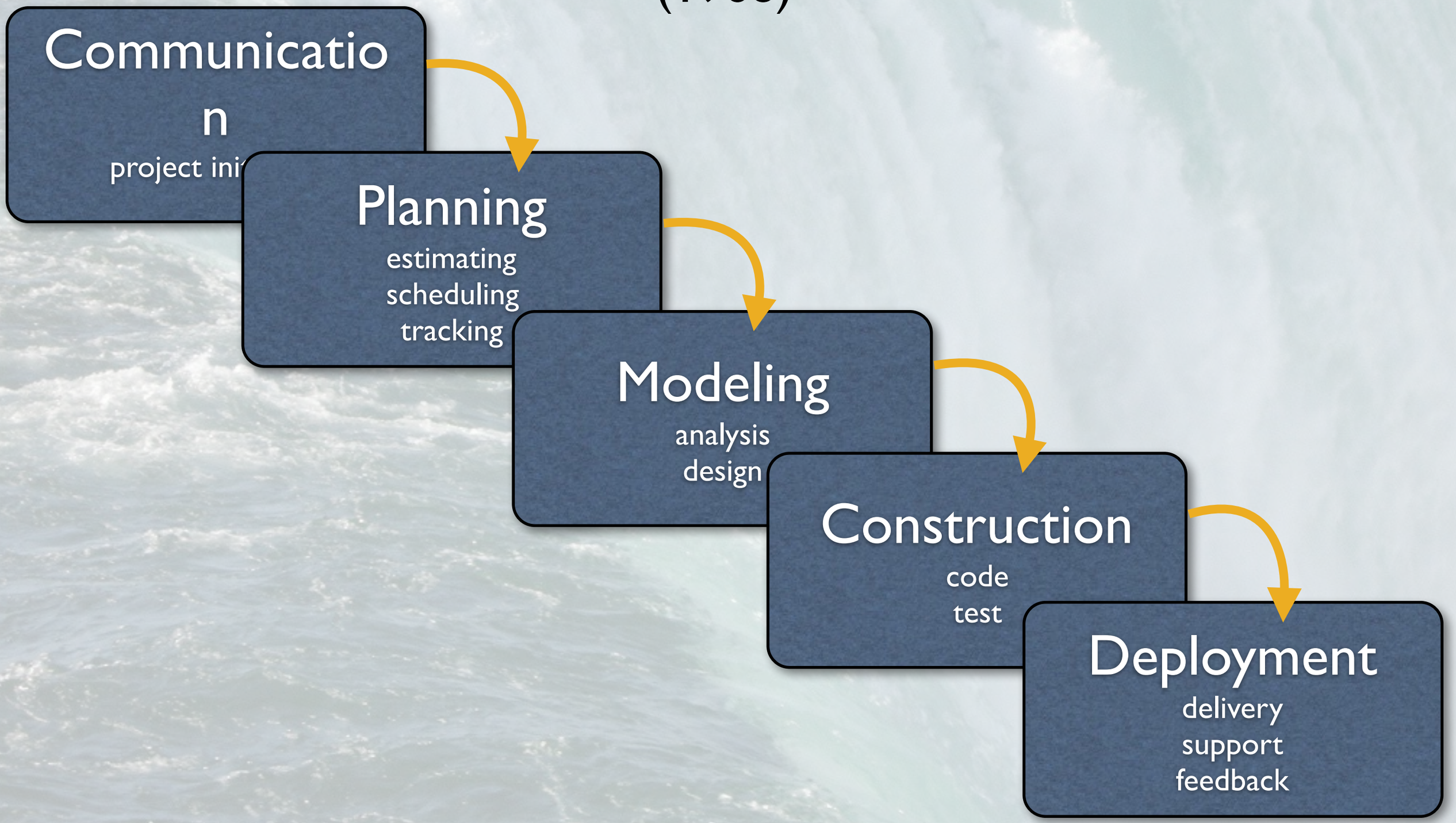


Planning

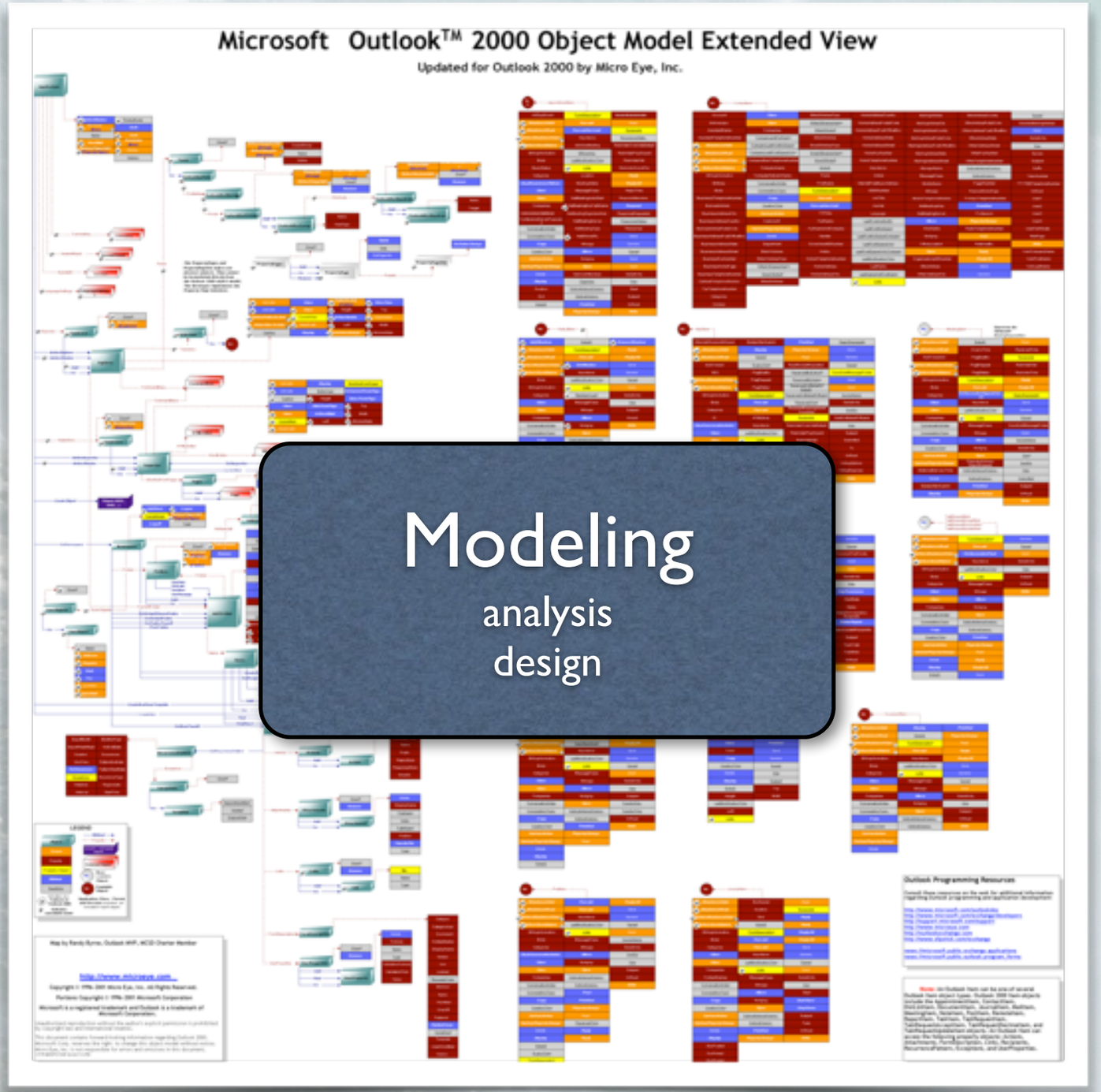
estimating
scheduling
tracking

Waterfall Model

(1968)

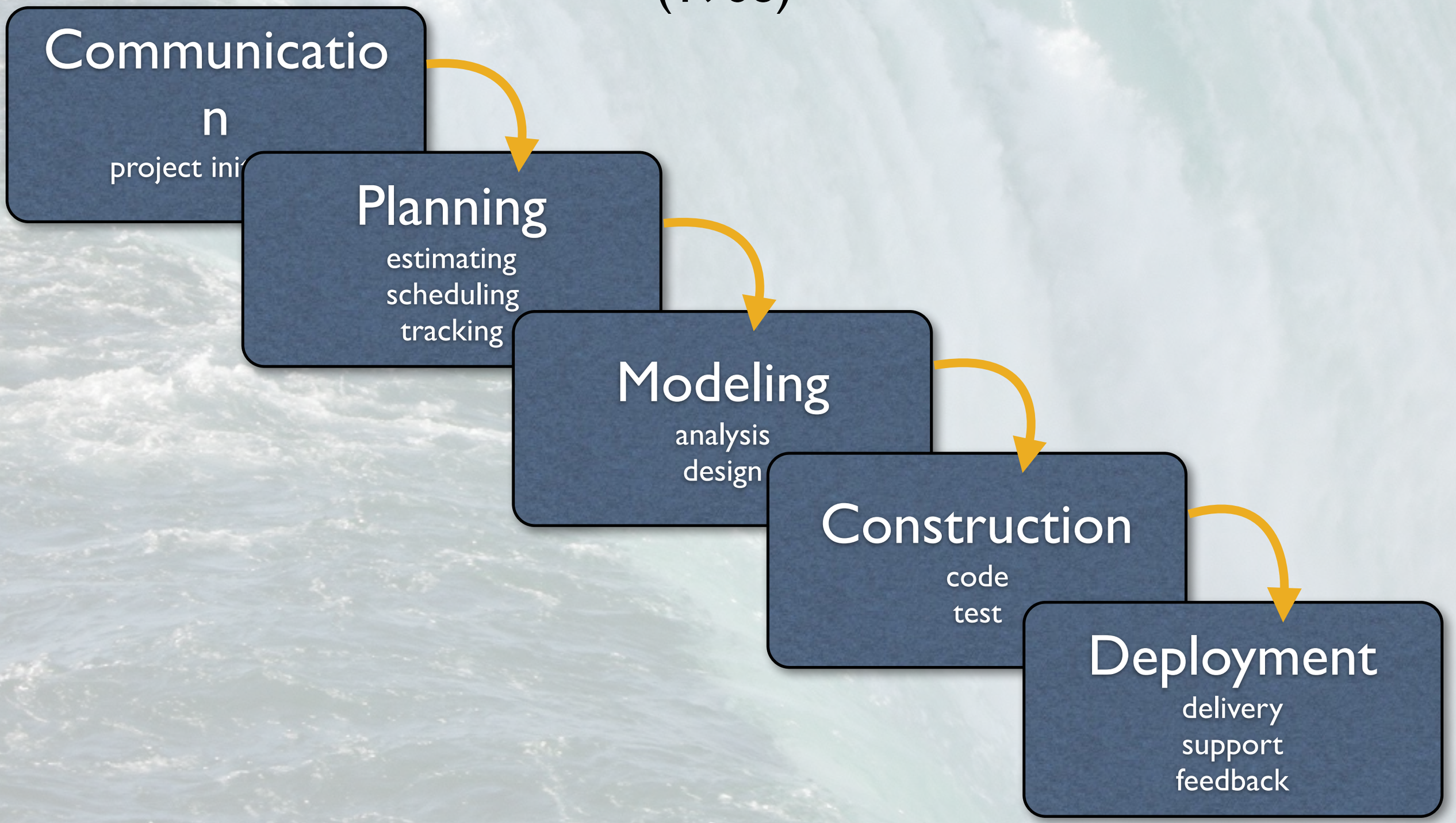


Waterfall Model (1968)

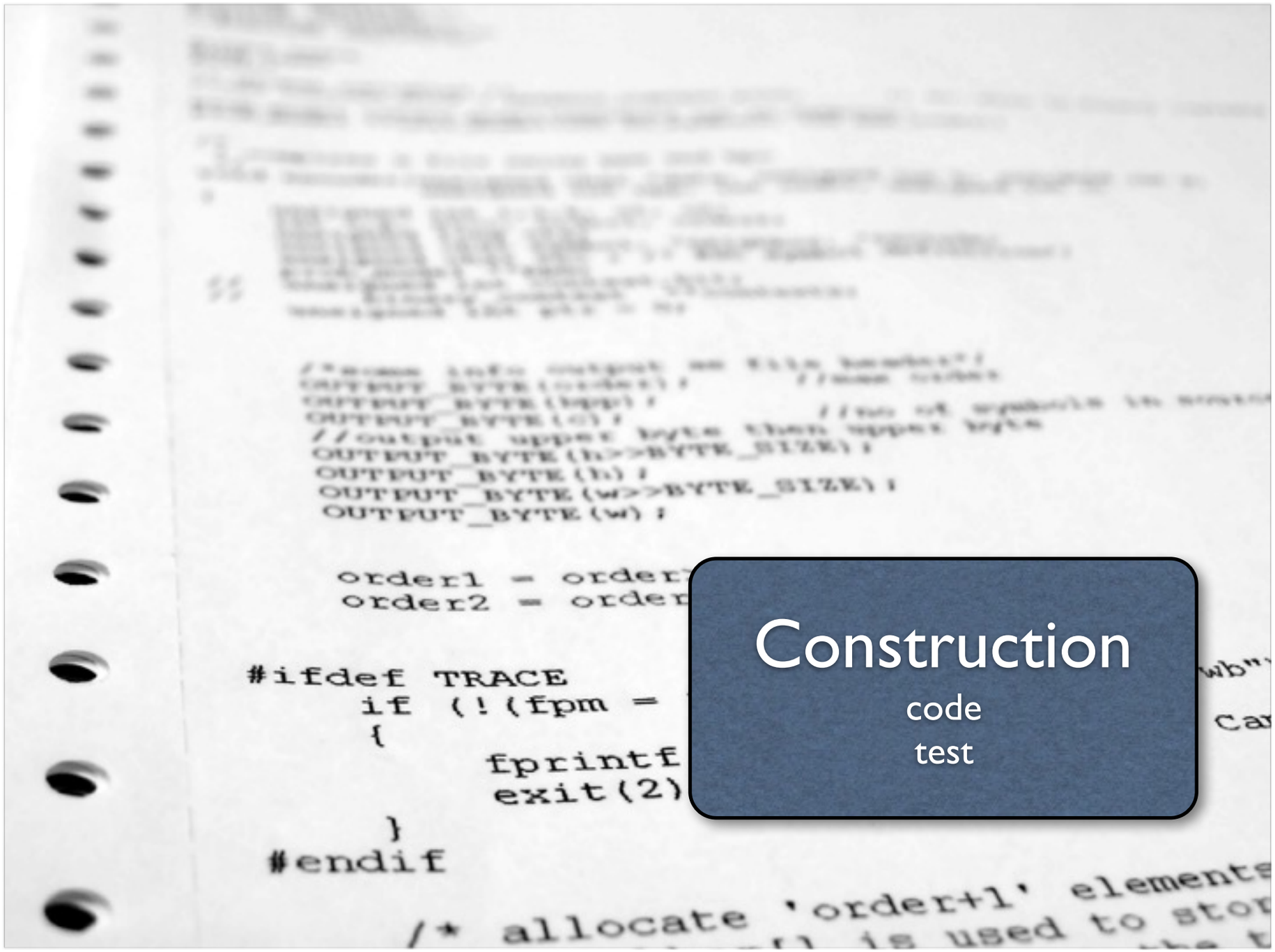


Waterfall Model

(1968)

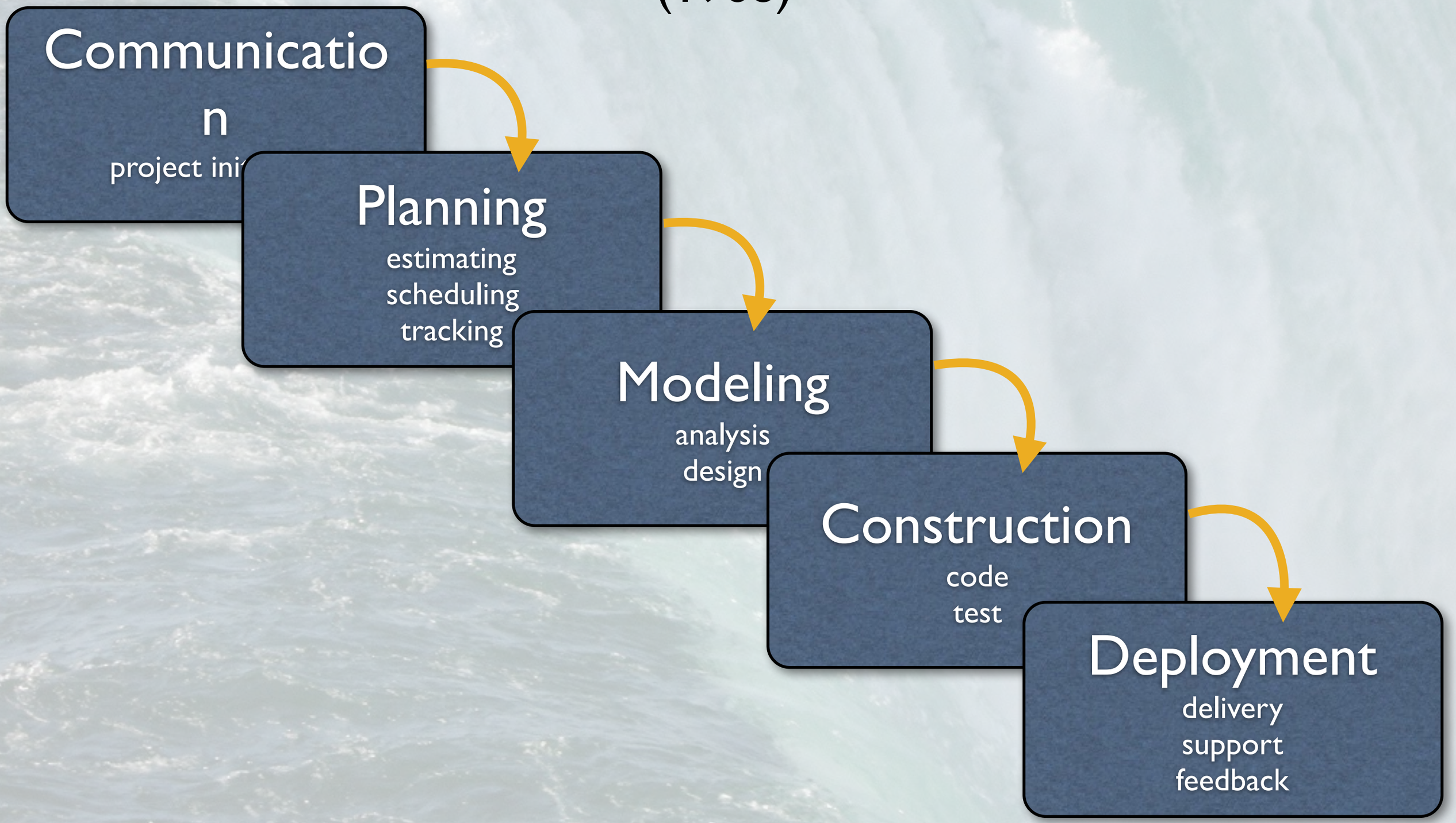


Waterfall Model



Waterfall Model

(1968)



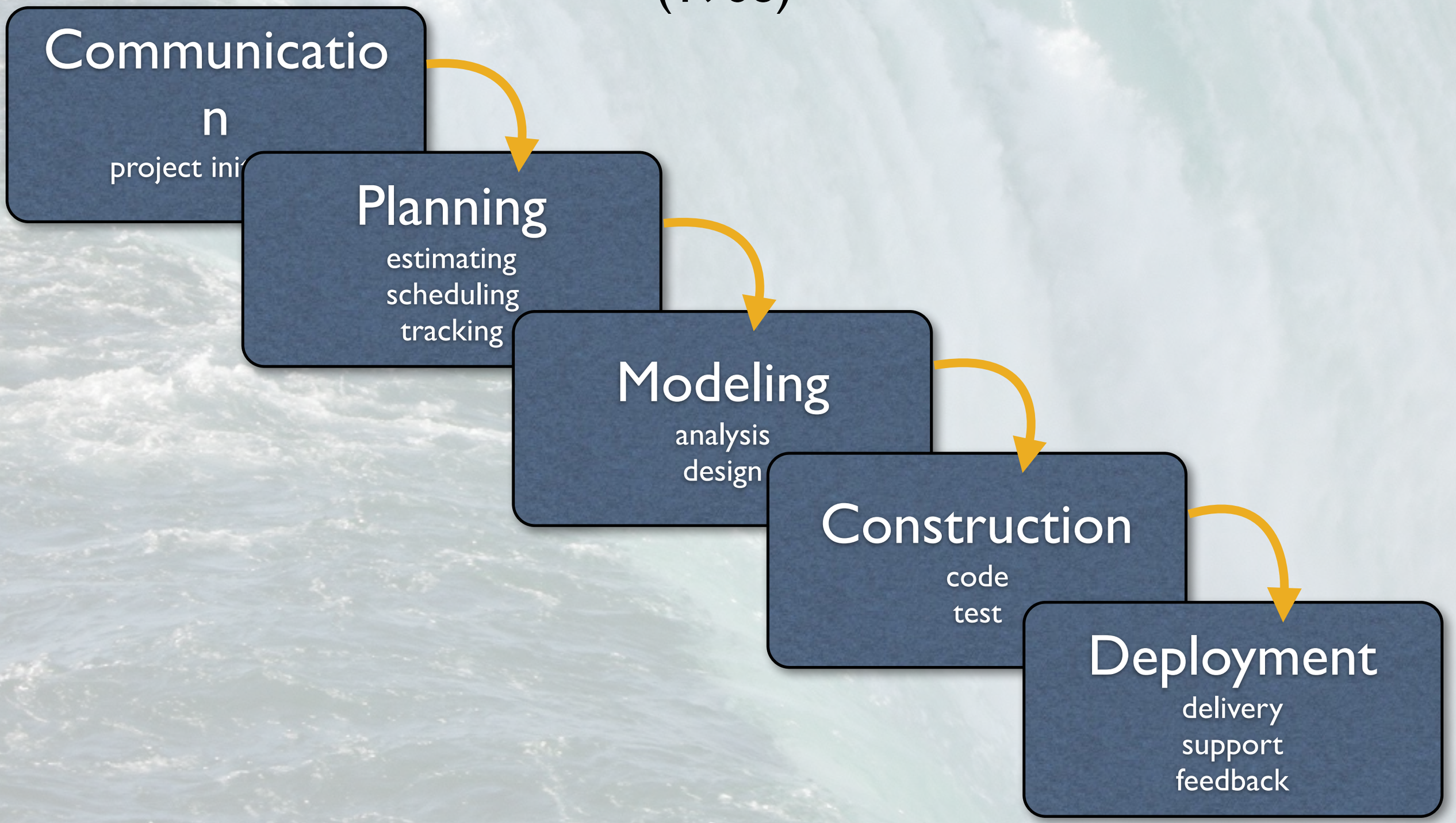
Deployment

Deployment

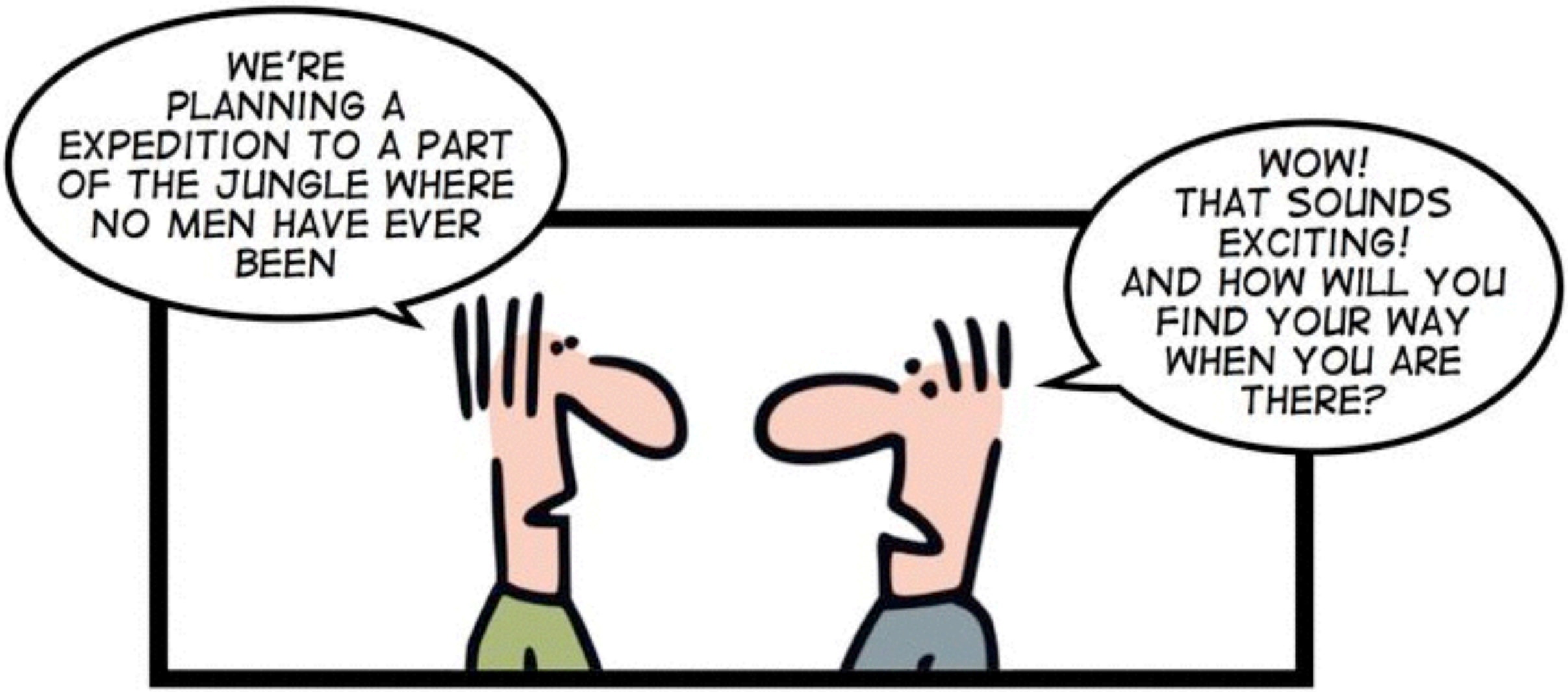
delivery
support
feedback

Waterfall Model

(1968)



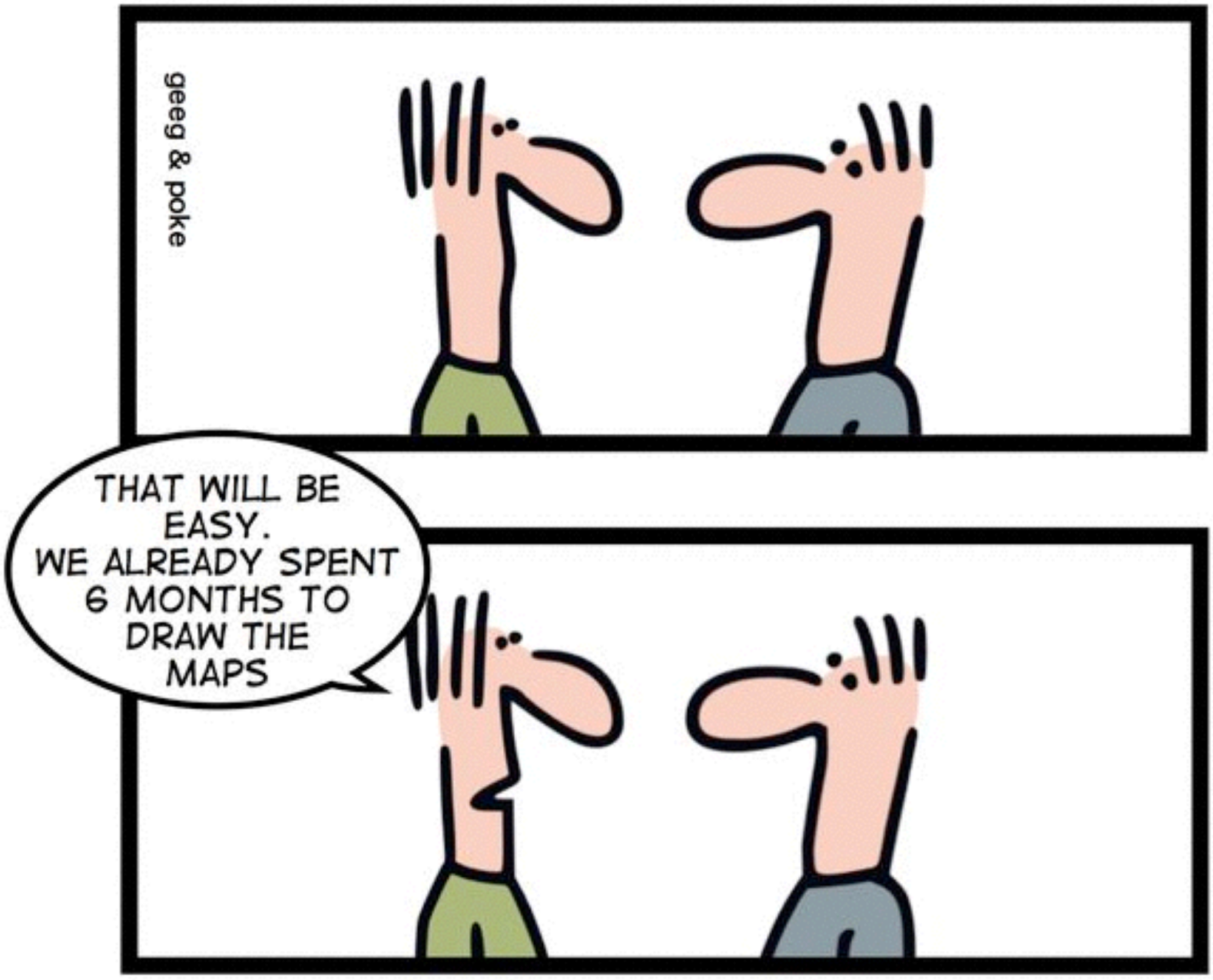
SIMPLY EXPLAINED



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


geeg & poke
THAT WILL BE EASY.
WE ALREADY SPENT 6 MONTHS TO DRAW THE MAPS
WATERFALL

Waterfall Model

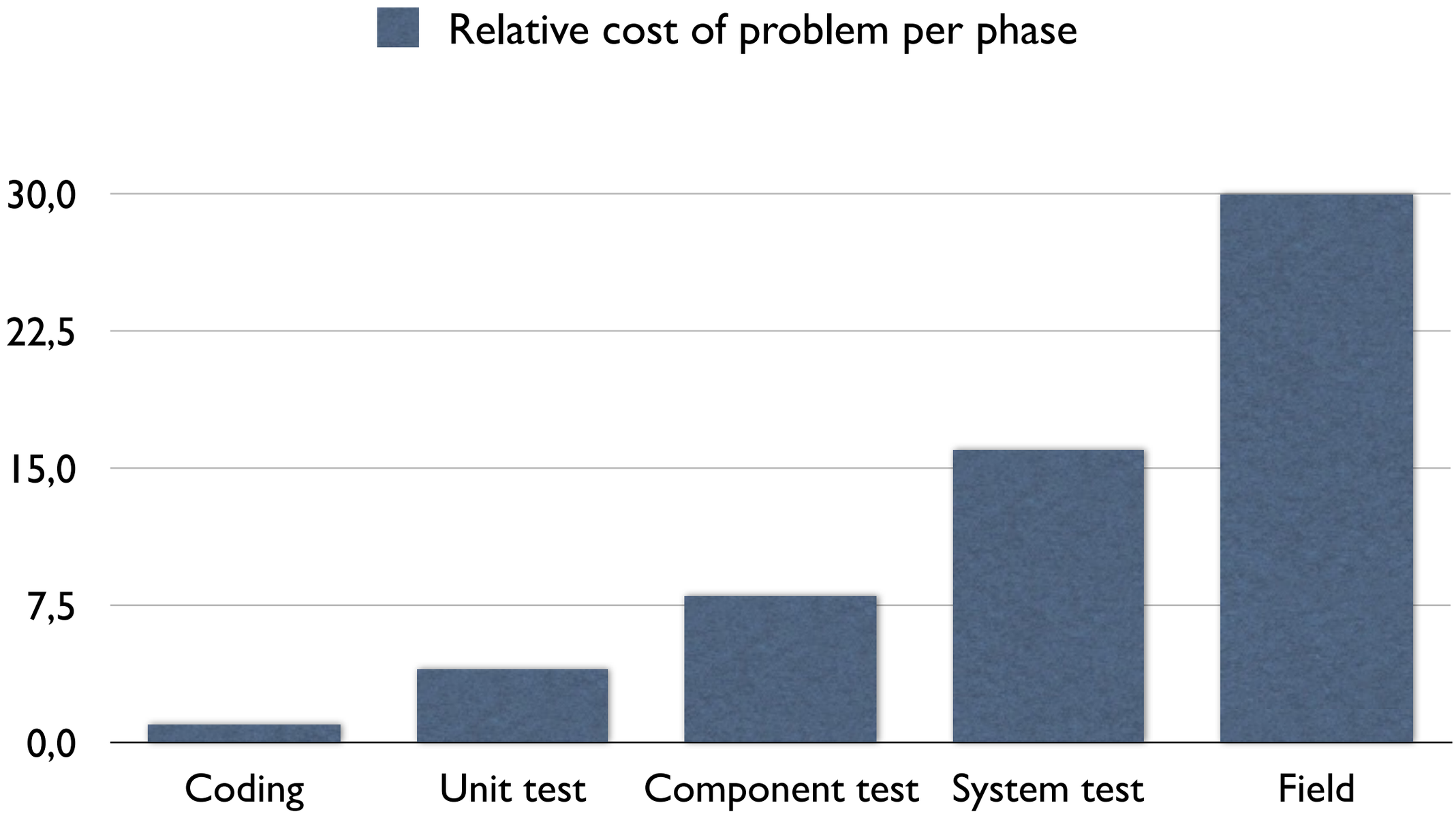
(1968)

- 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

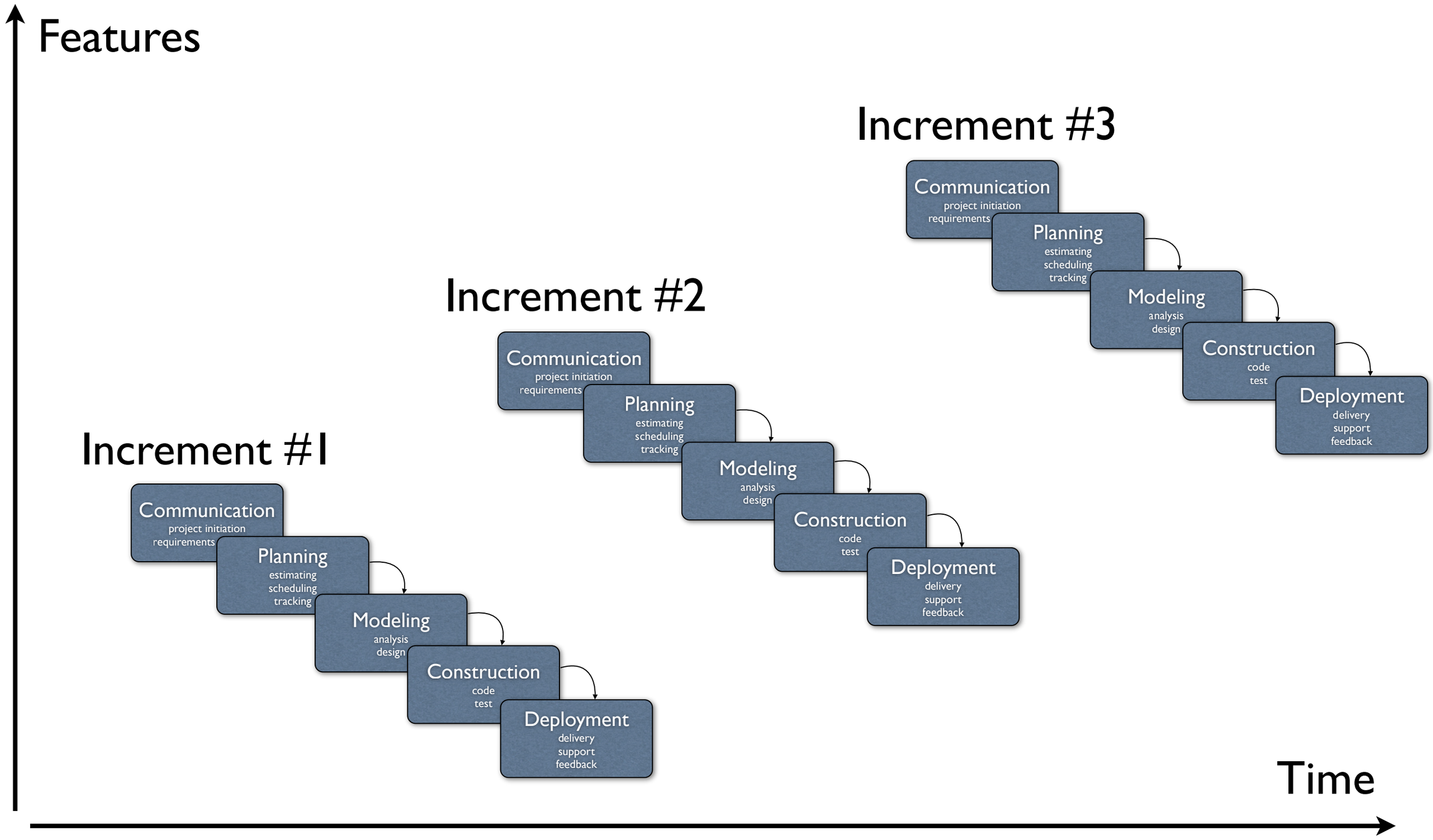
Boehm's first law

Errors are most frequent
during *requirements* and *design* activities
and are the more expensive
the later they are removed.

Problem Cost



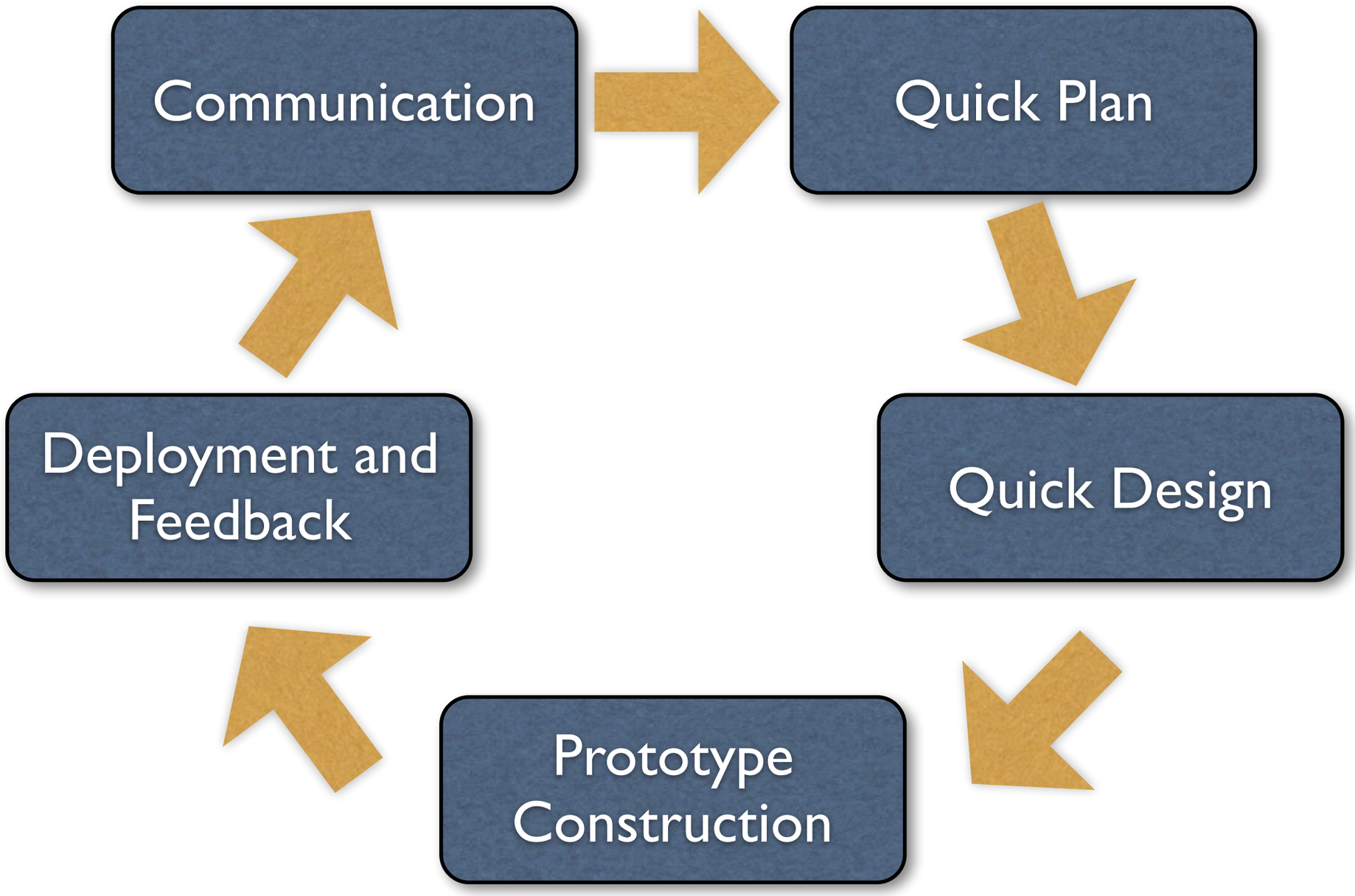
Incremental Model



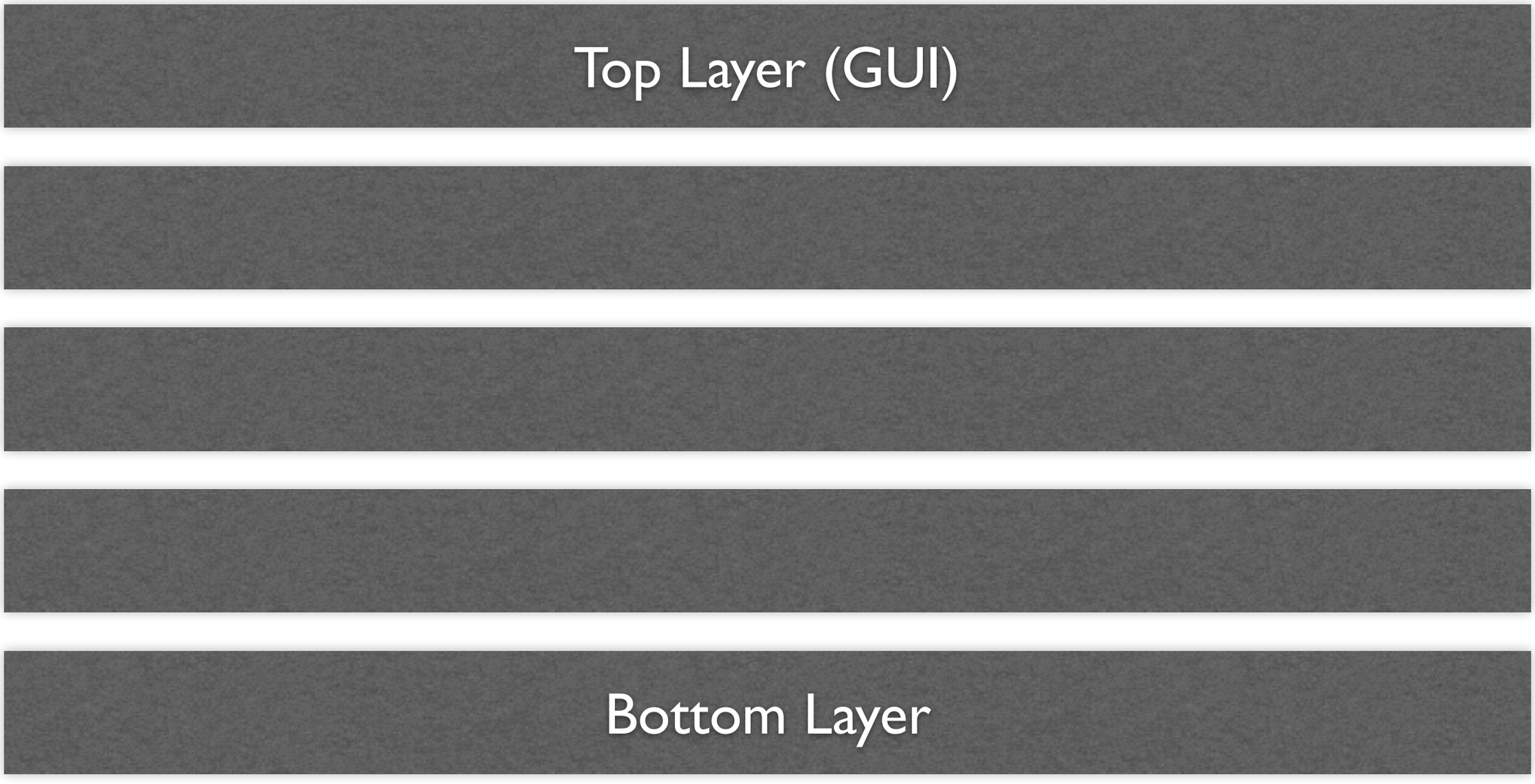
Incremental Model

- Each linear sequence produces a particular "increment" to the software
- First increment typically core product; more features added by later increments
- Allows flexible allocation of resources

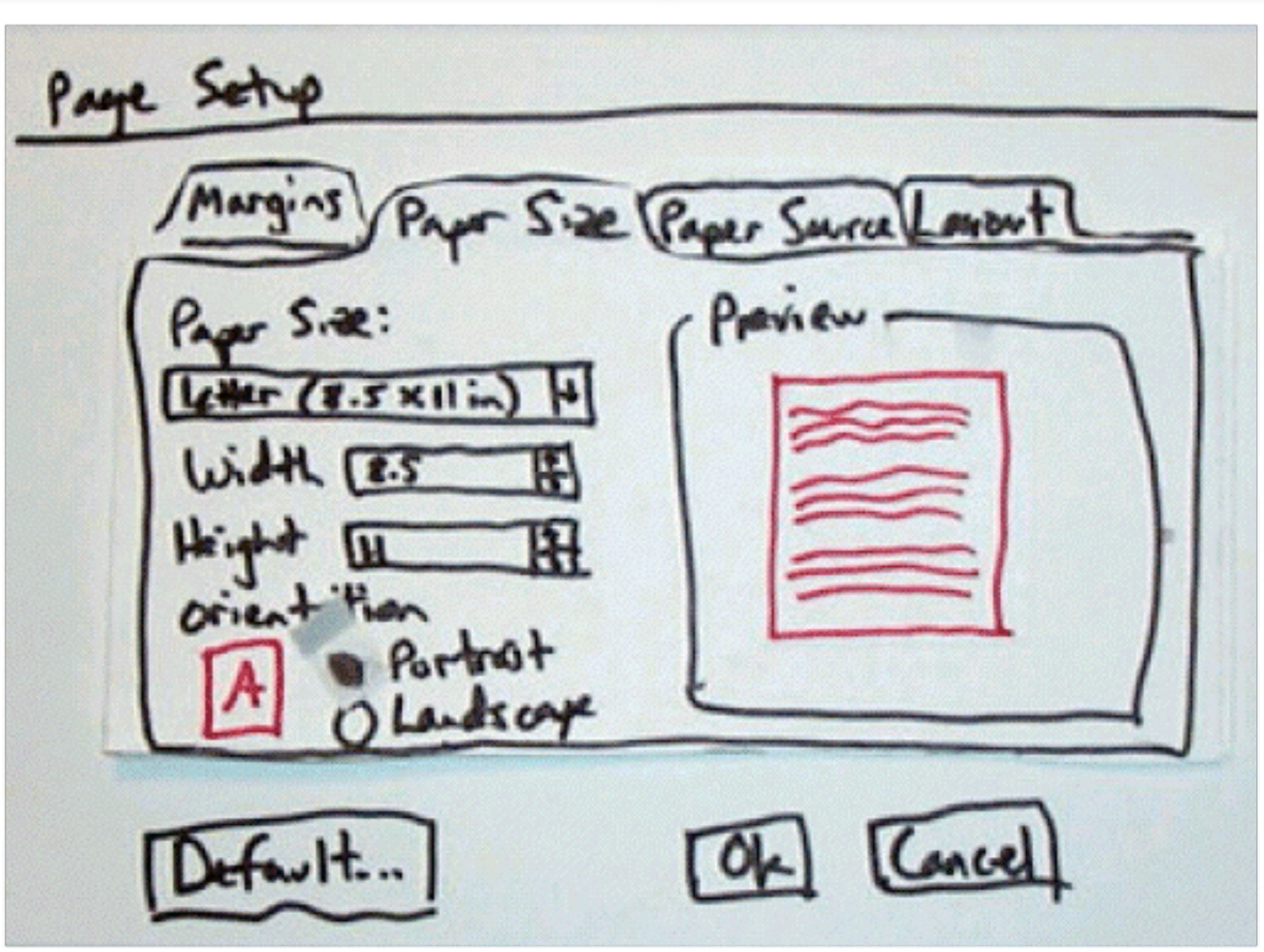
Prototyping



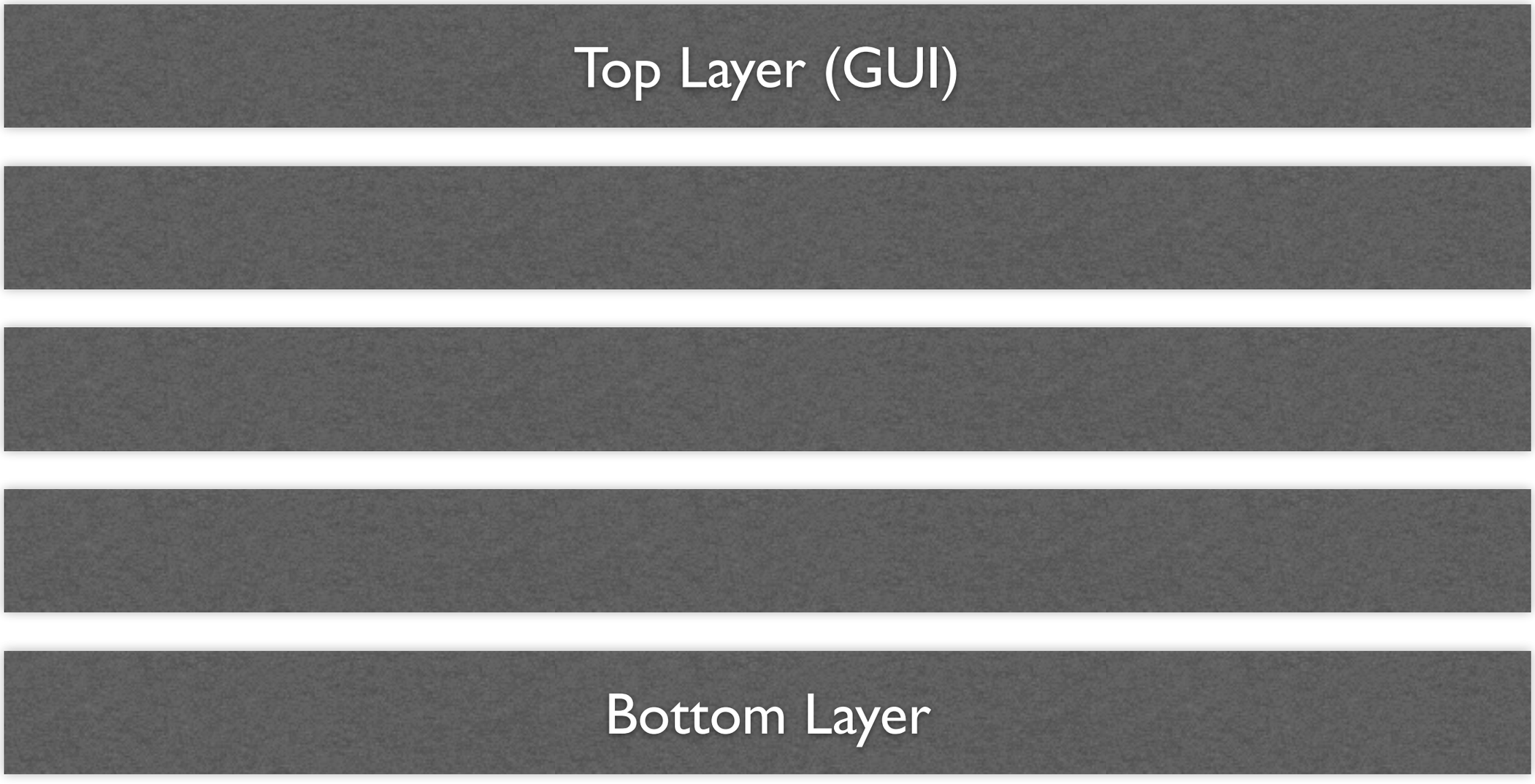
Prototypes



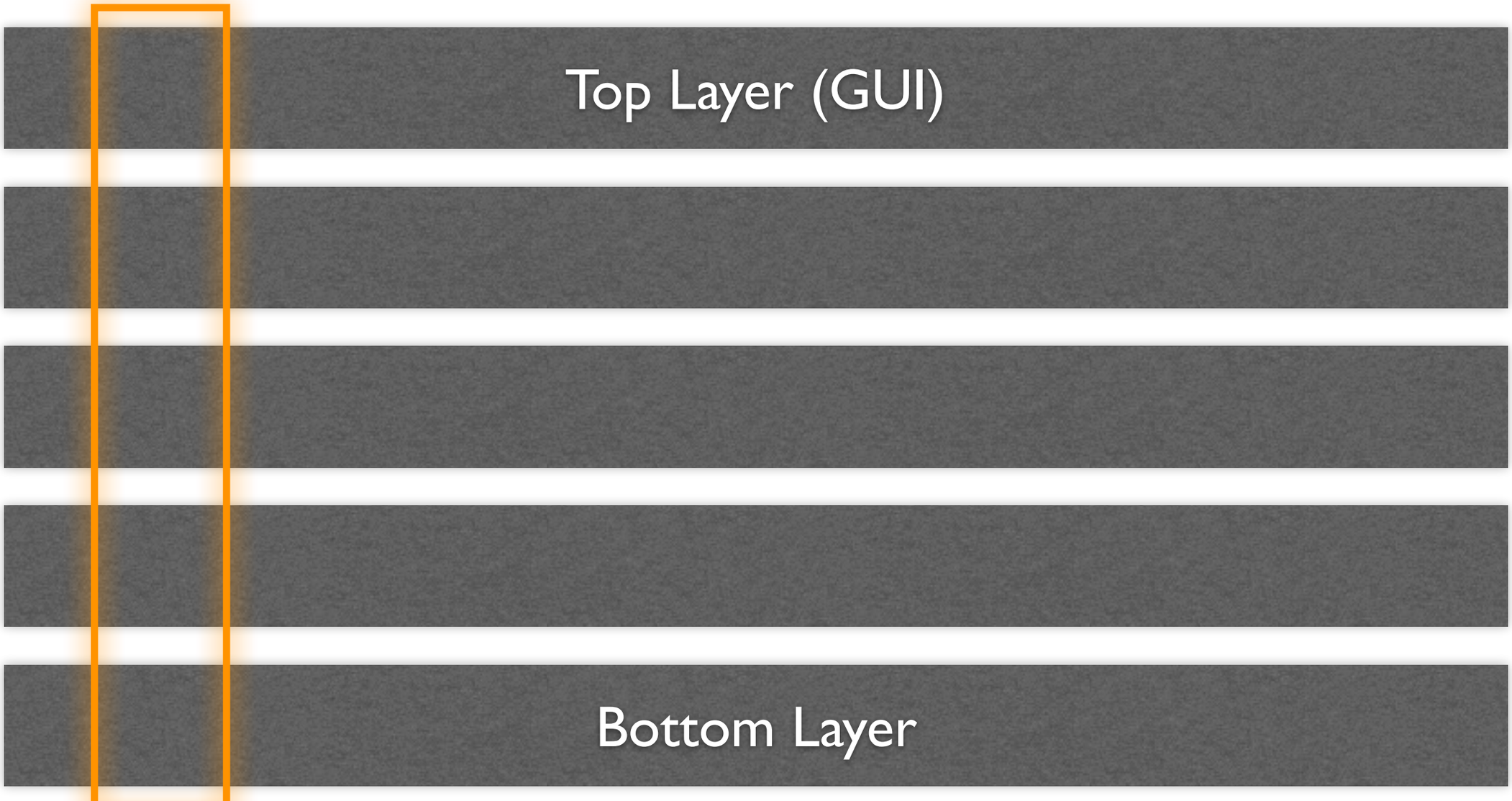
Horizontal Prototype



Prototypes



Vertical Prototype

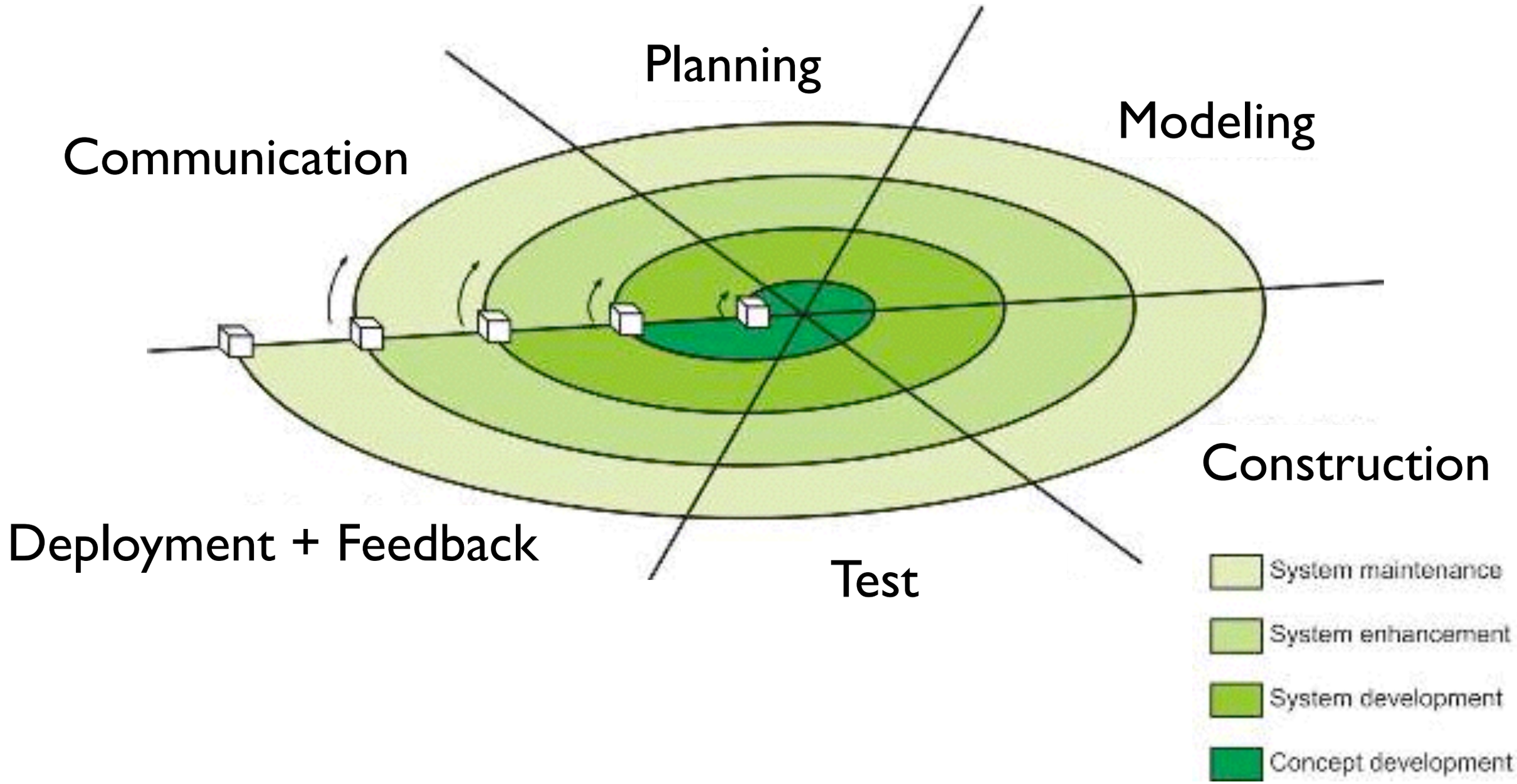


Prototypes

- A *horizontal prototype* tests a particular *layer* (typically the GUI) of the system
- A *vertical prototype* tests a particular *functionality* across all layers
- Resist pressure to turn a prototype into a final result!

Spiral Model

(1988)

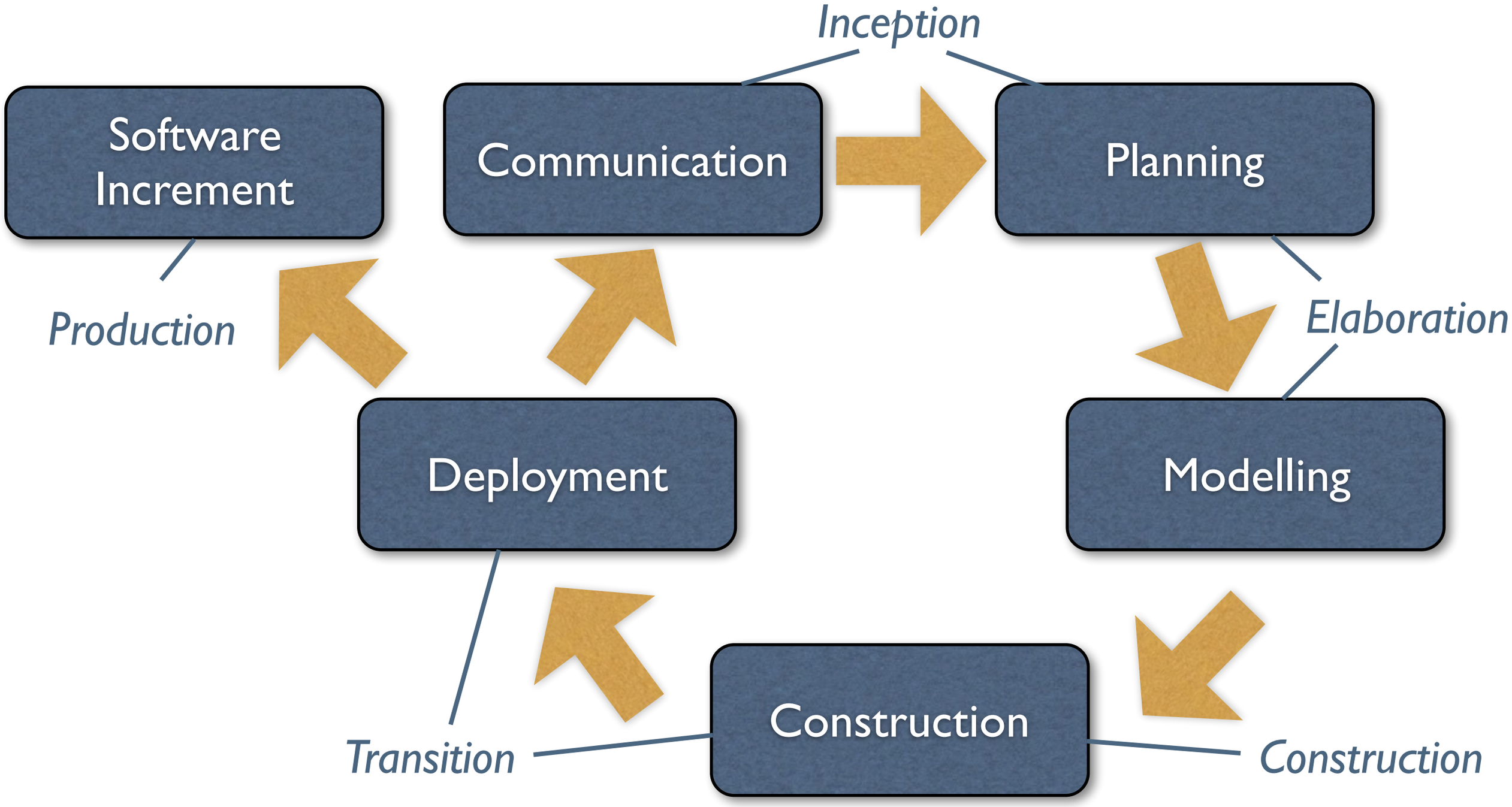


Spiral Model

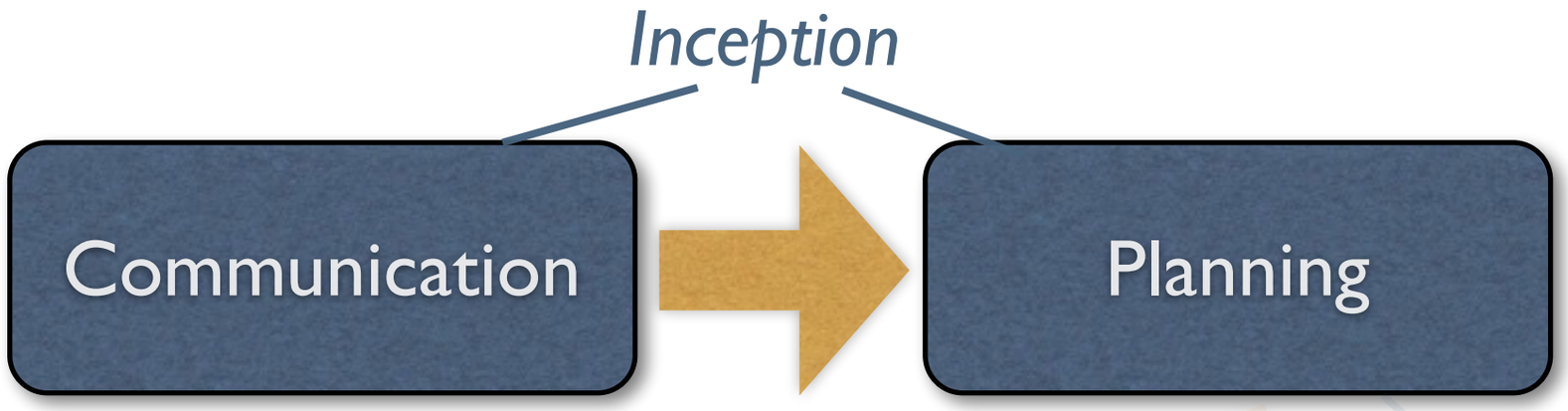
- System is developed in series of evolutionary releases
- Milestones for each iteration of the spiral
- Process does not end with delivery
- Reflects iterative nature of development

Unified Process

(1999)



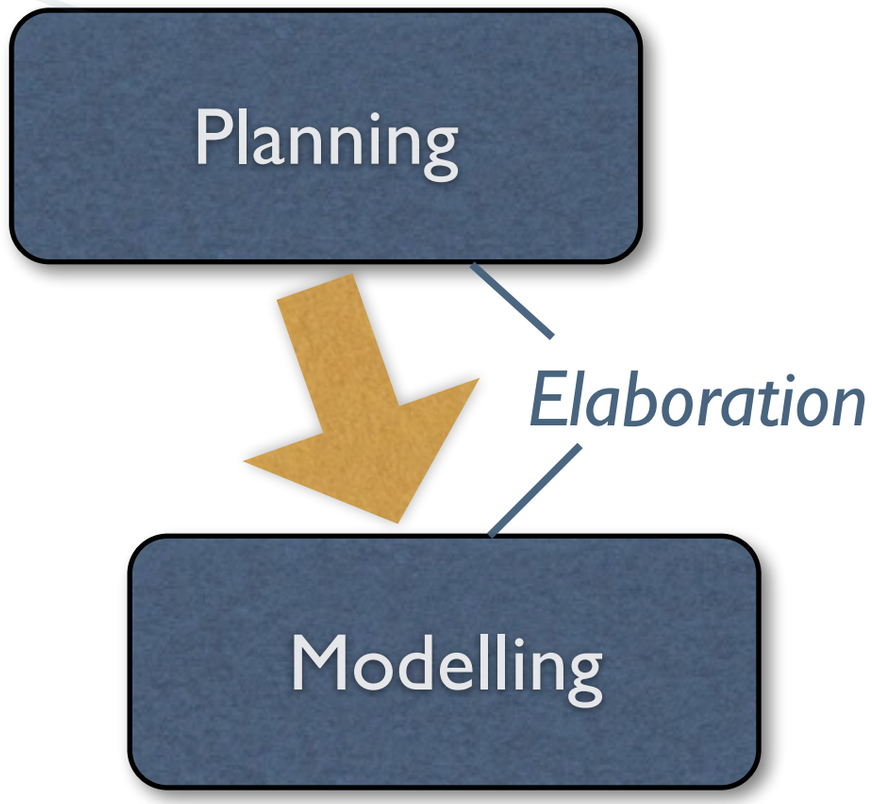
Inception



- Encompasses communication with user + planning
- Results in a set of *use cases*
- Architecture is just a tentative outline

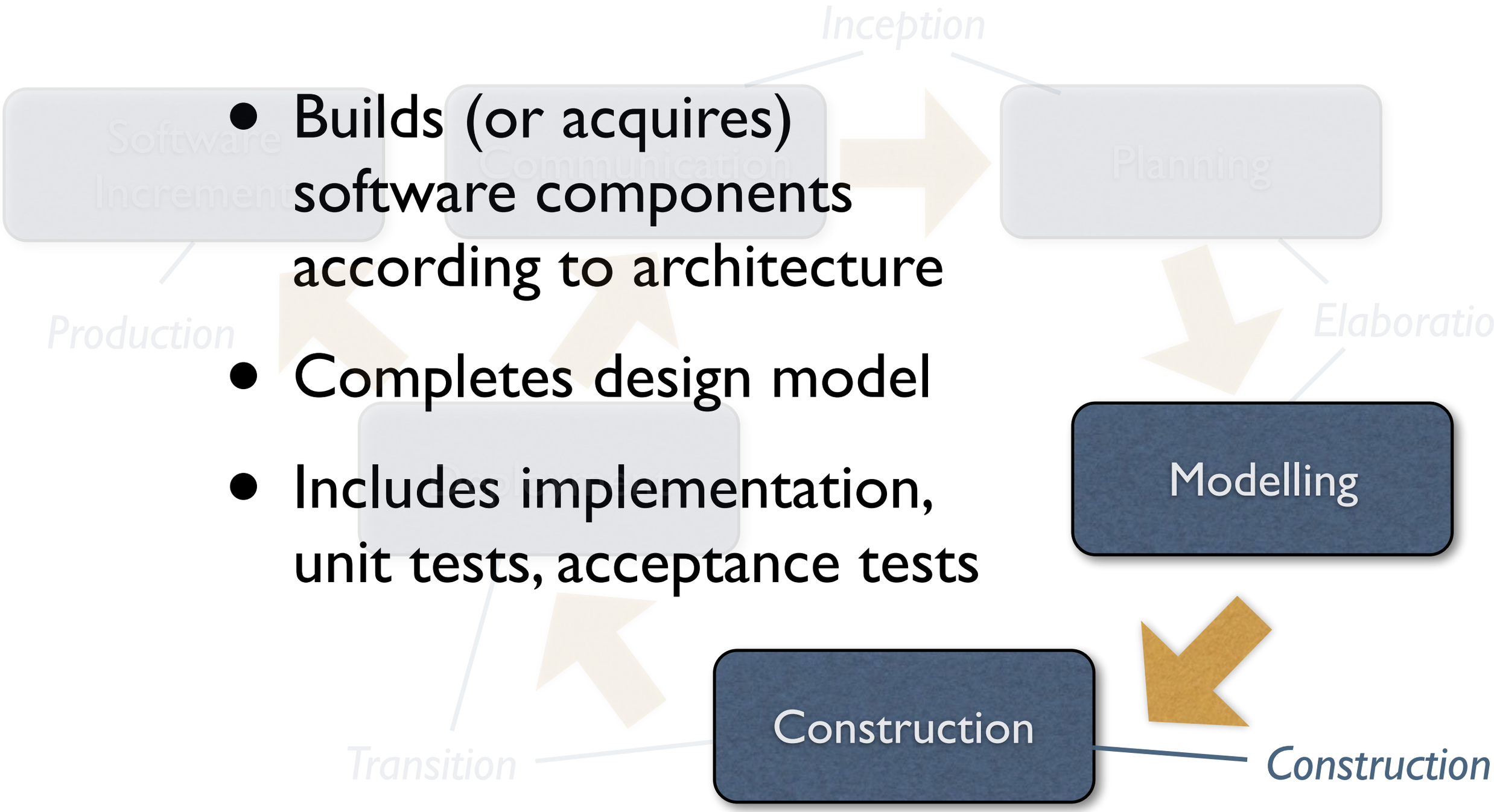
Elaboration

- Refines and expands preliminary use cases
- Provides architecture and initial design model



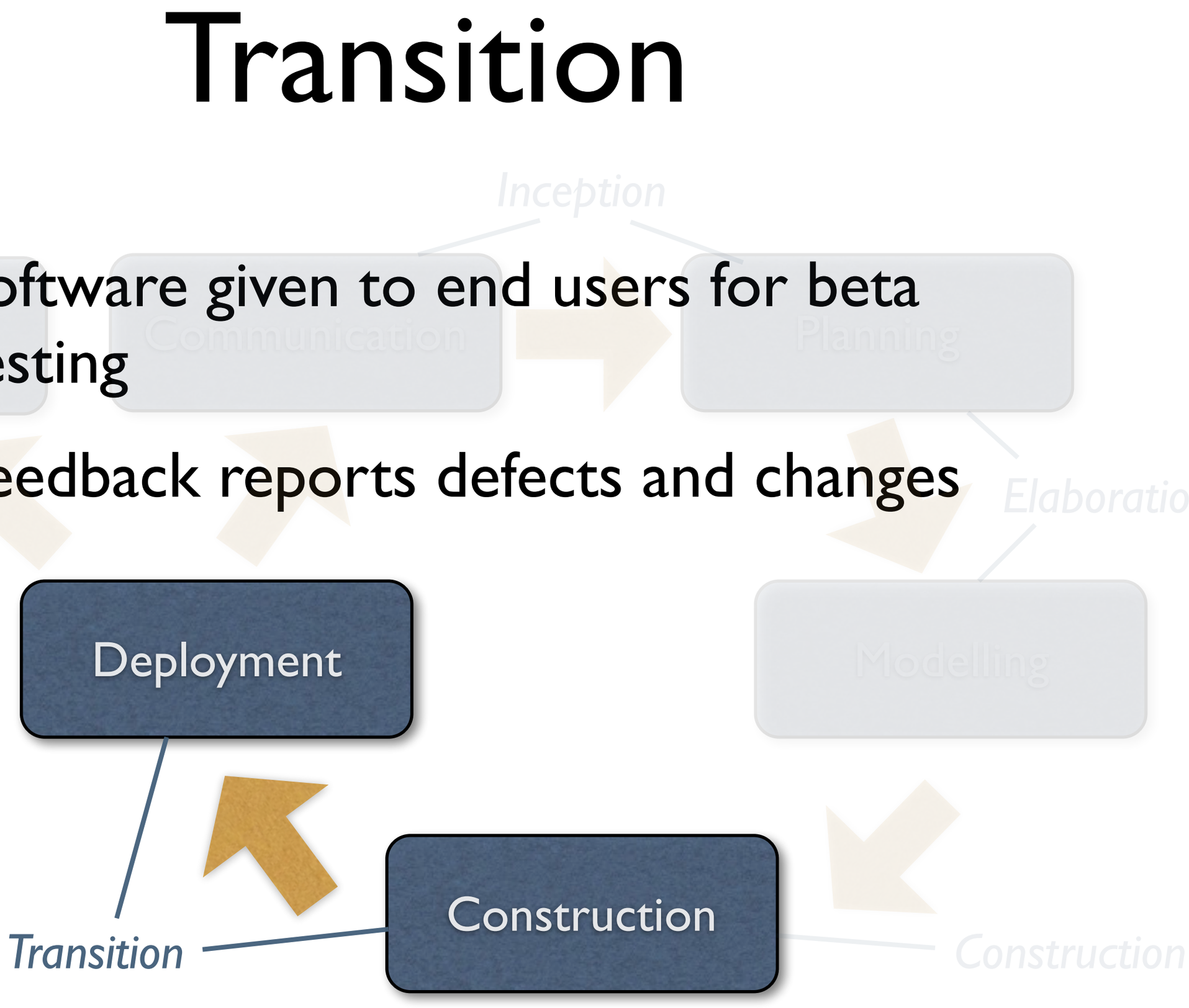
Construction

- Builds (or acquires) software components according to architecture
- Completes design model
- Includes implementation, unit tests, acceptance tests

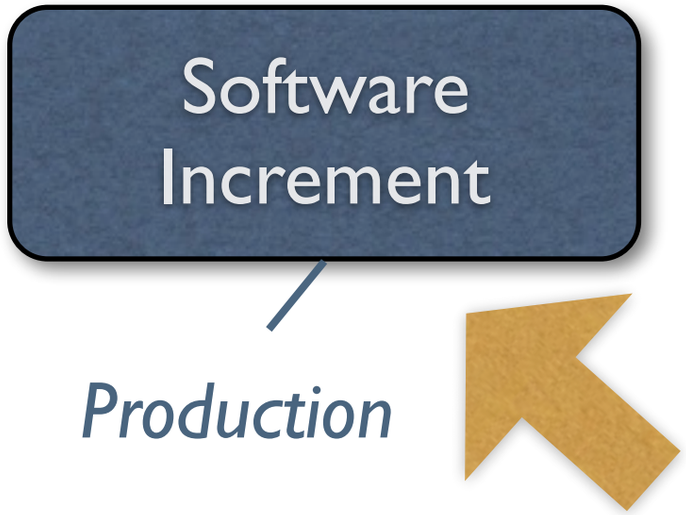


Transition

- Software given to end users for beta testing
- Feedback reports defects and changes

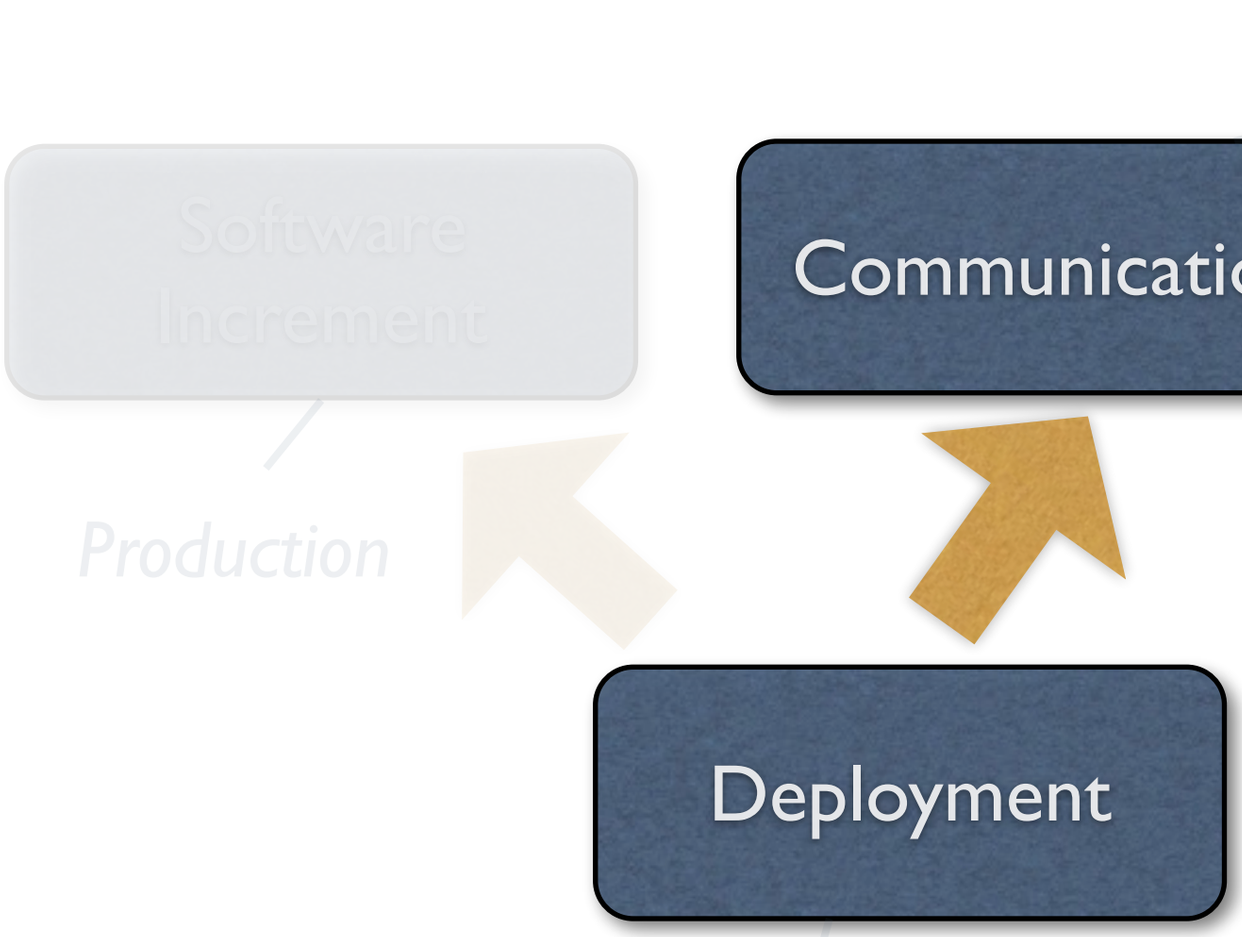


Production



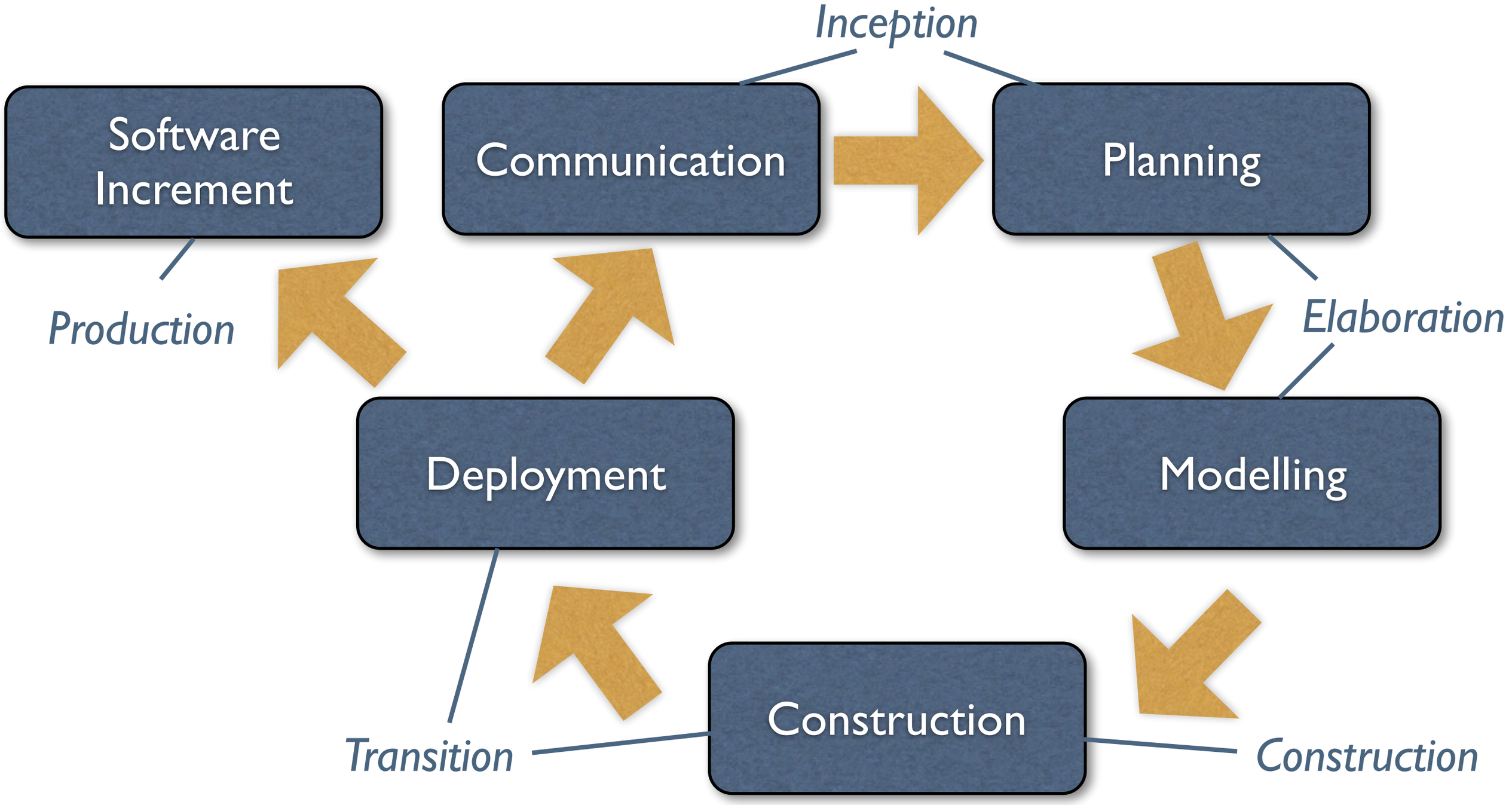
- Software is deployed
- Problems are monitored

Re-Iteration



- Feedback results in new iteration for next

Unified Process



Unified Process

- Draws on best features of conventional process models
- Emphasizes software architecture and design
- Integrates with UML modeling techniques (more on this later)


63


WASSERWACHT
NEURUPPIN
NARWHAL
YAMAHA

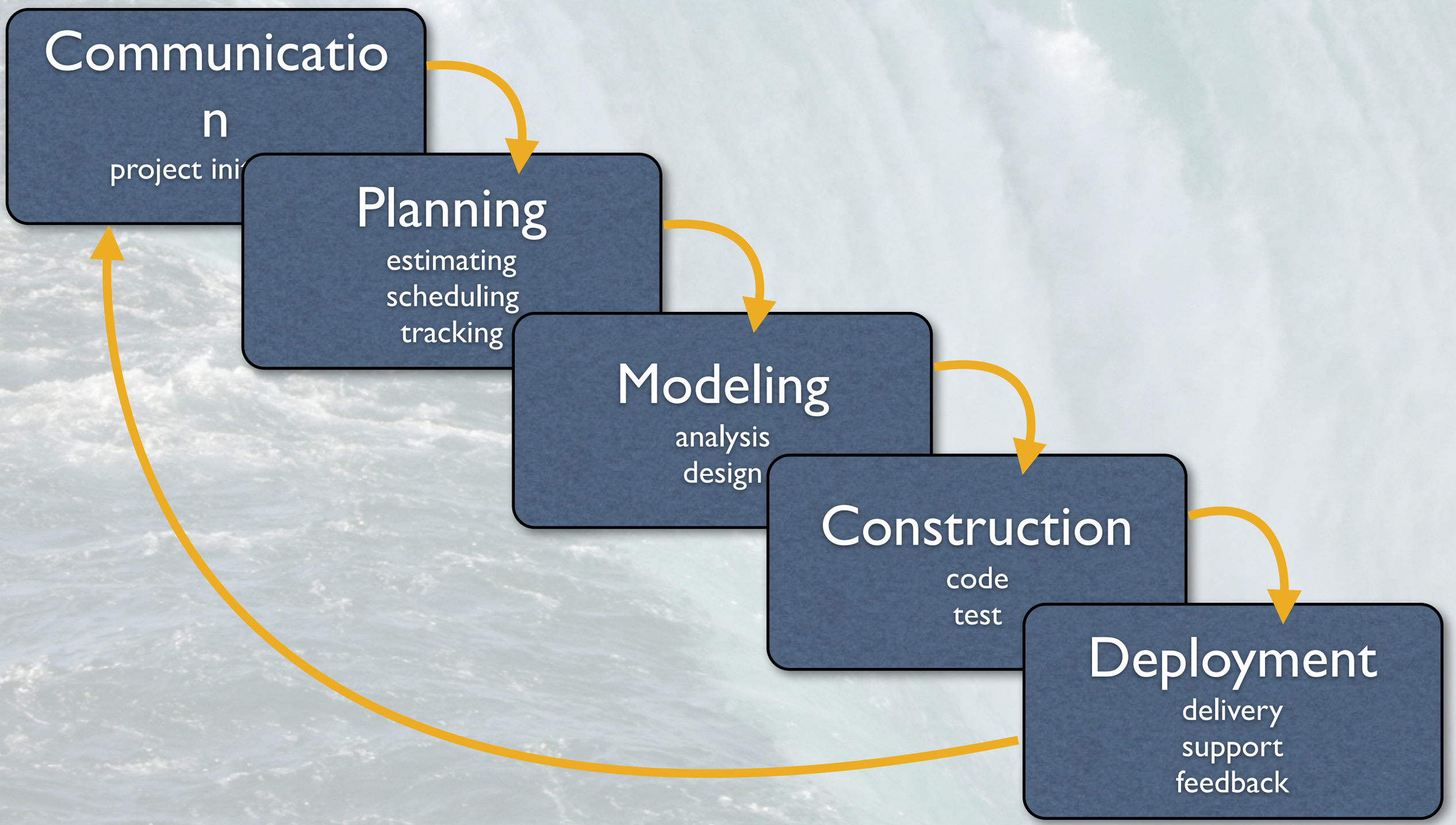
Manifesto for Agile Software Development (2001)

- Individuals and activities over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan..

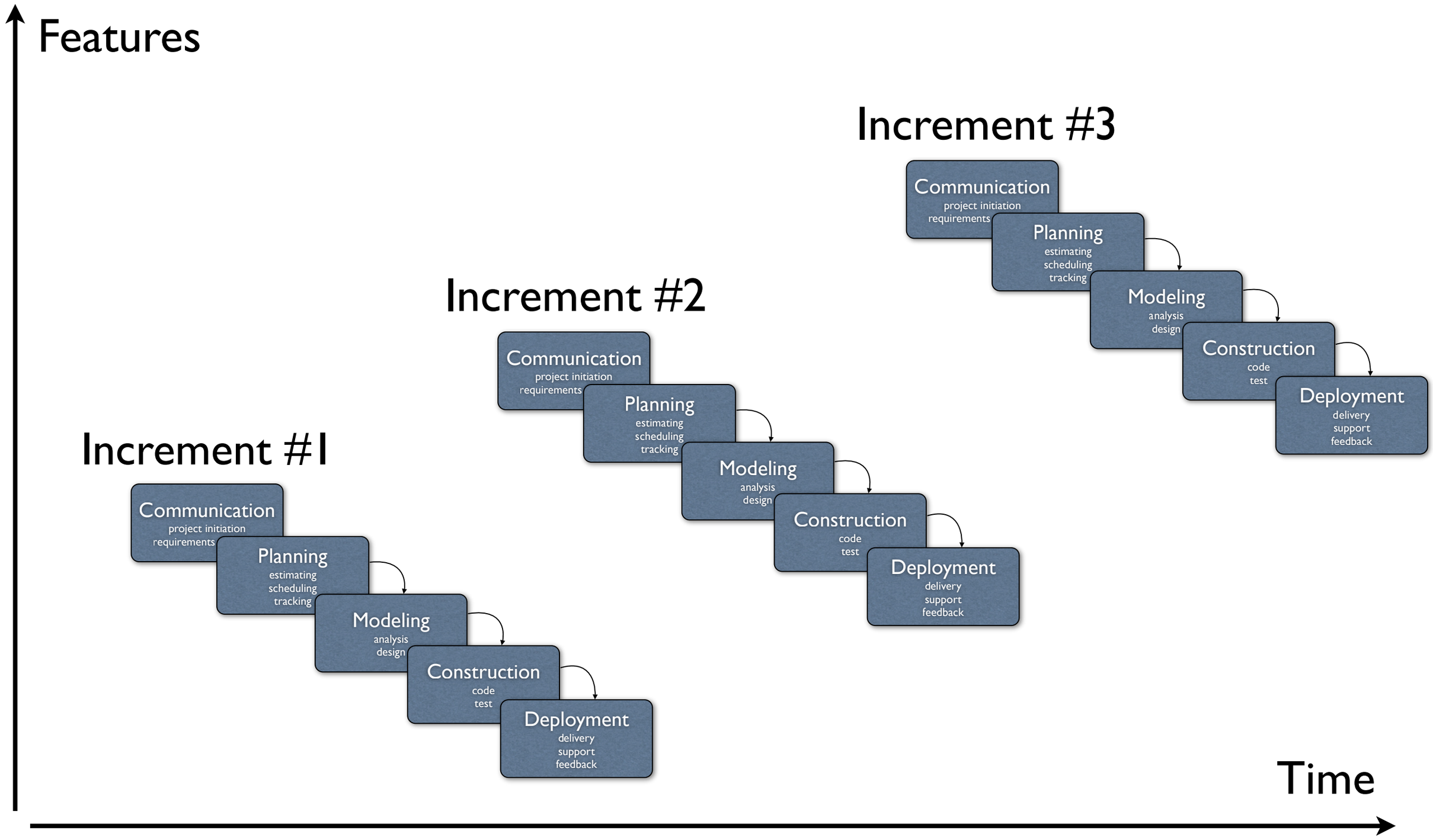
What is Agile Development?

- Fast development? Hacking? Prototyping? Uncontrolled fun? Programmer heaven?
- Agility = ability to react to changing situations quickly, appropriately, and effectively.
  - notice changes early
  - initiate action promptly
  - create a feasible and effective alternative plan quickly
  - reorient work and resources quickly and effectively

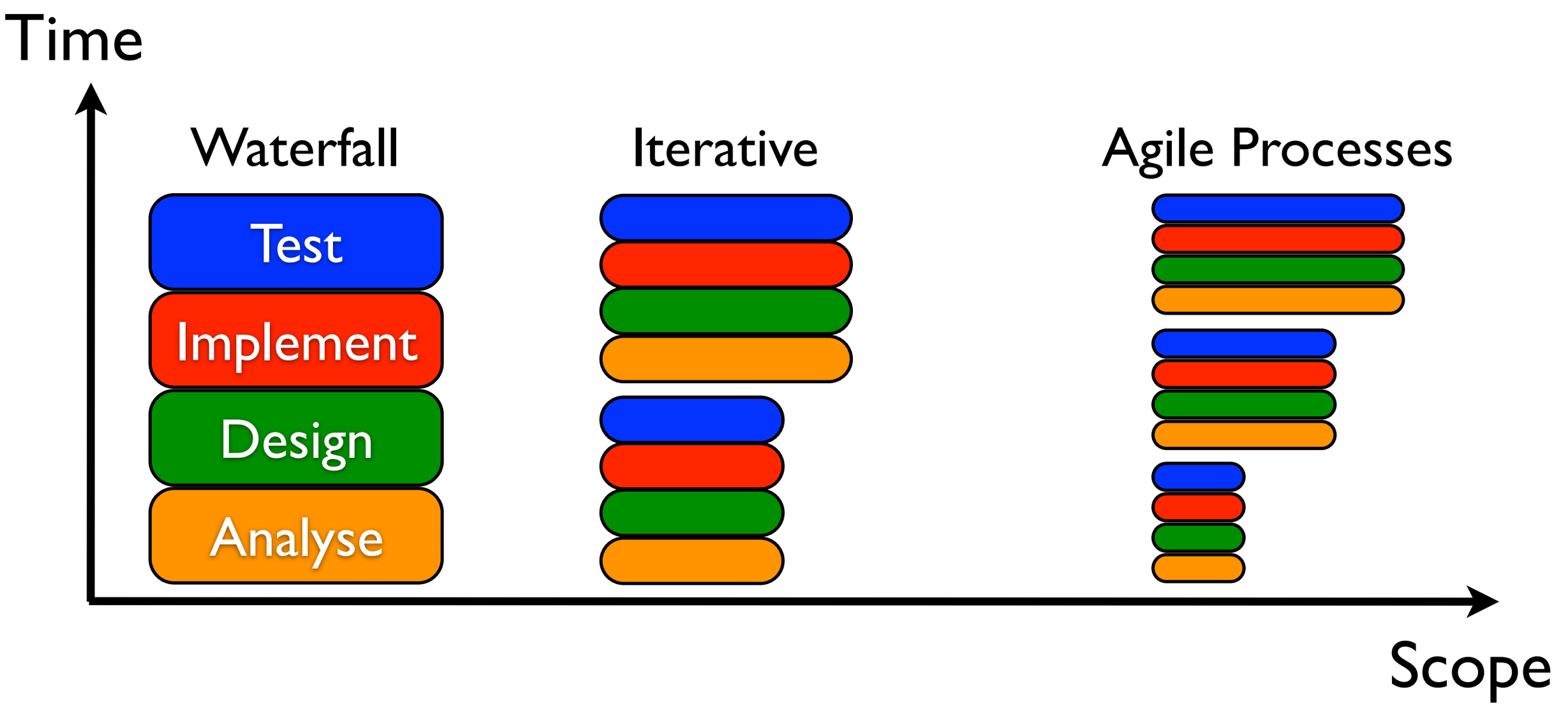
Agile?


Communicatio
n
project ini
Planning
estimating
scheduling
tracking
Modeling
analysis
design
Construction
code
test
Deployment
delivery
support
feedback

Incremental Model



Agile Processes



Credits: Prof. Bodik

Agile vs. Plan-driven

Agile

- Low criticality
- Senior developers
- Requirements change very often
- Small number of developers
- Culture that thrives on chaos

Plan-driven

- High criticality
- Junior developers
- Requirements don't change too often
- Large number of developers
- Culture that demands order

What is an Agile Process?

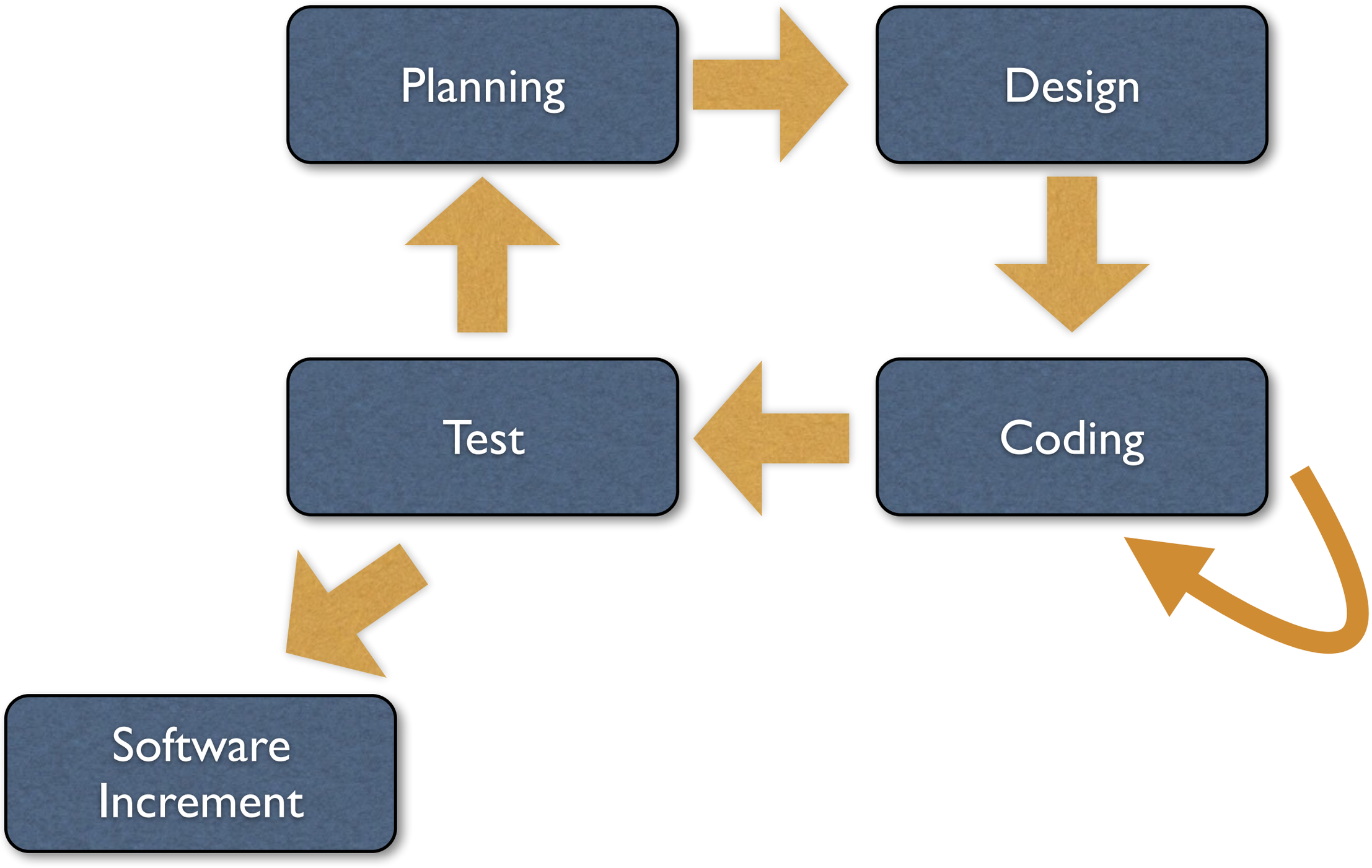
- Difficult to predict which requirements will persist or change in the future.
- For many types of software, design and development are interleaved.
- Analysis, design, construction, and testing are not as predictable.

So, how to tackle unpredictability?

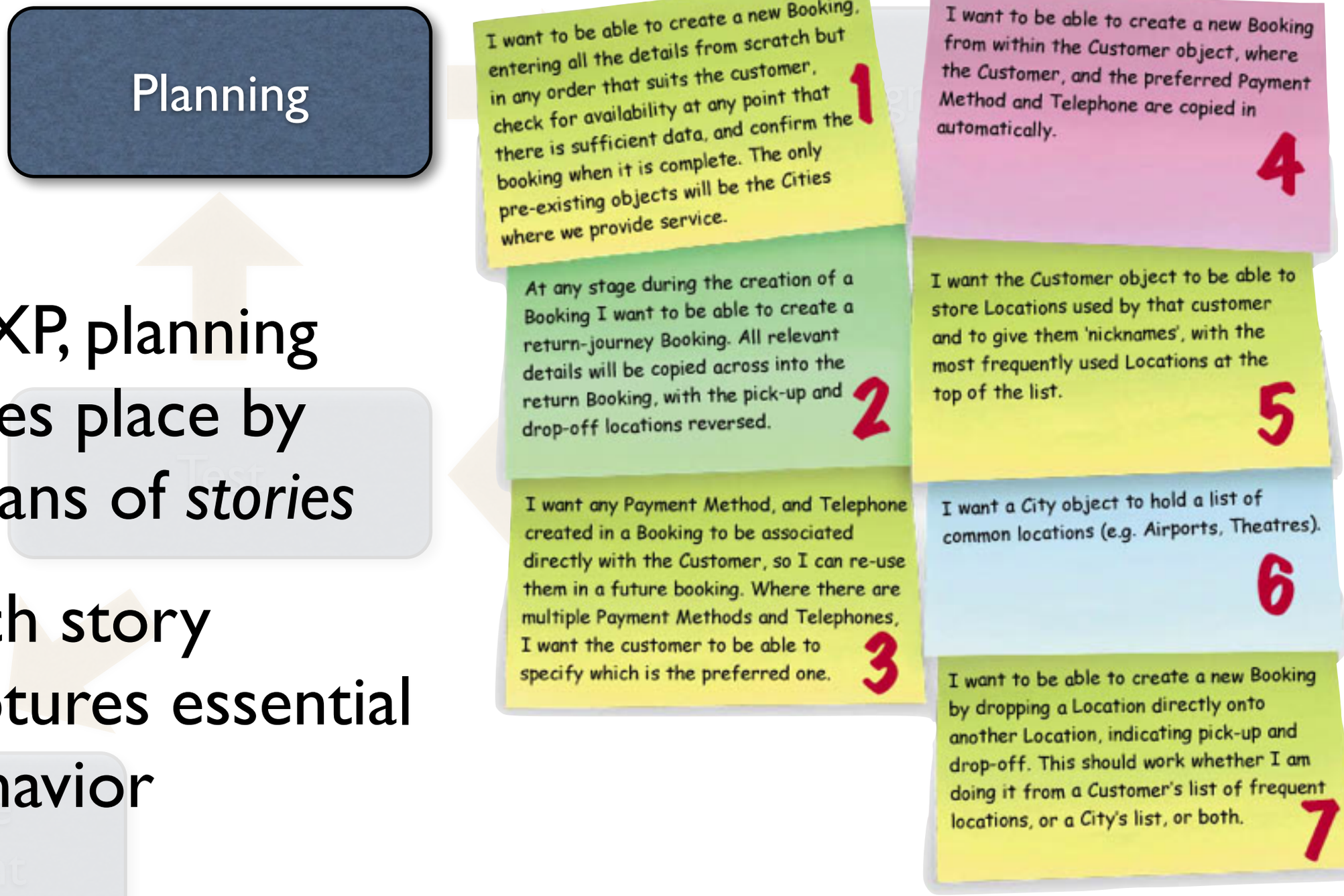
make the process adaptable...

Extreme Programming

(1999–)

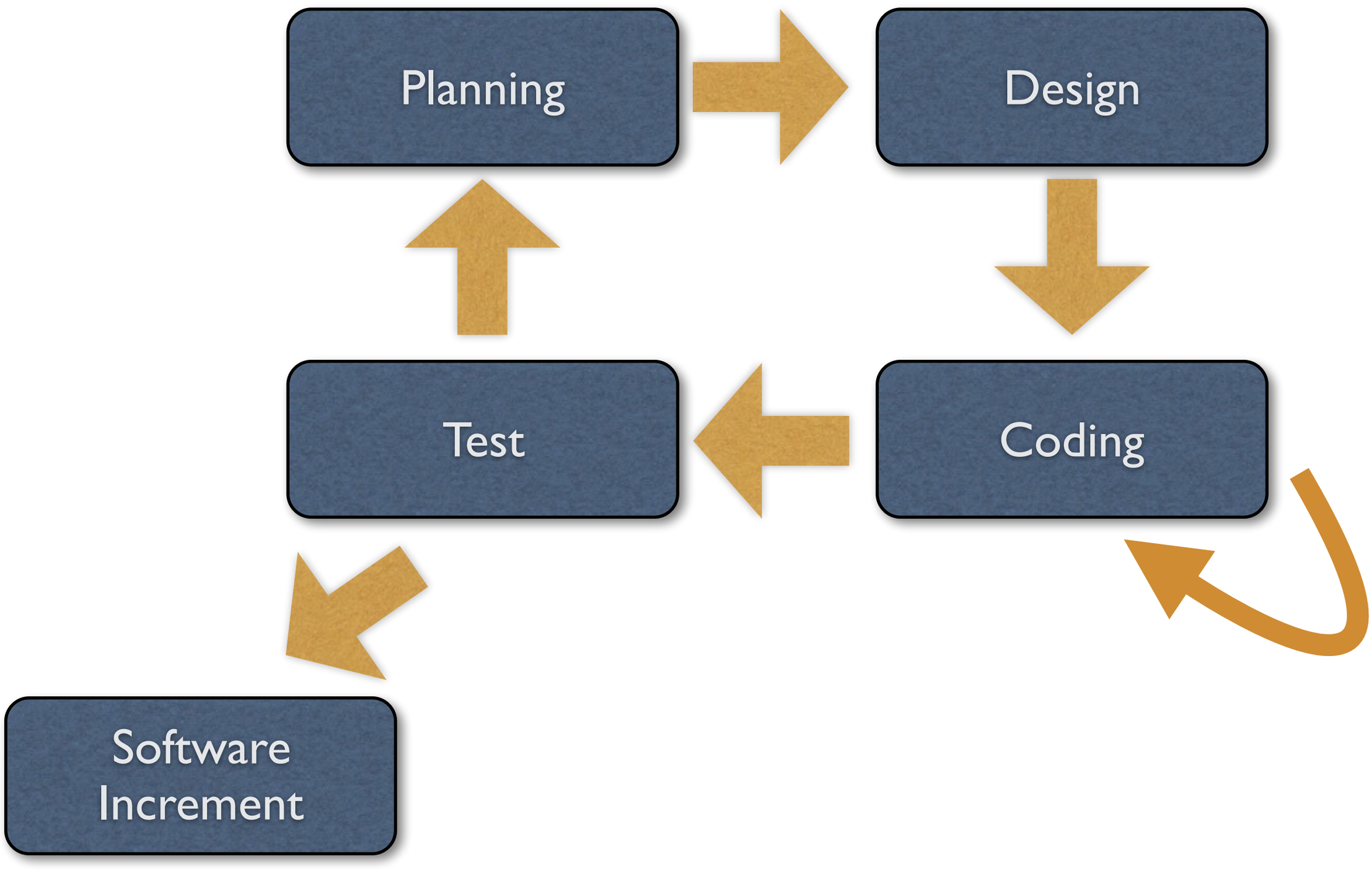


Planning

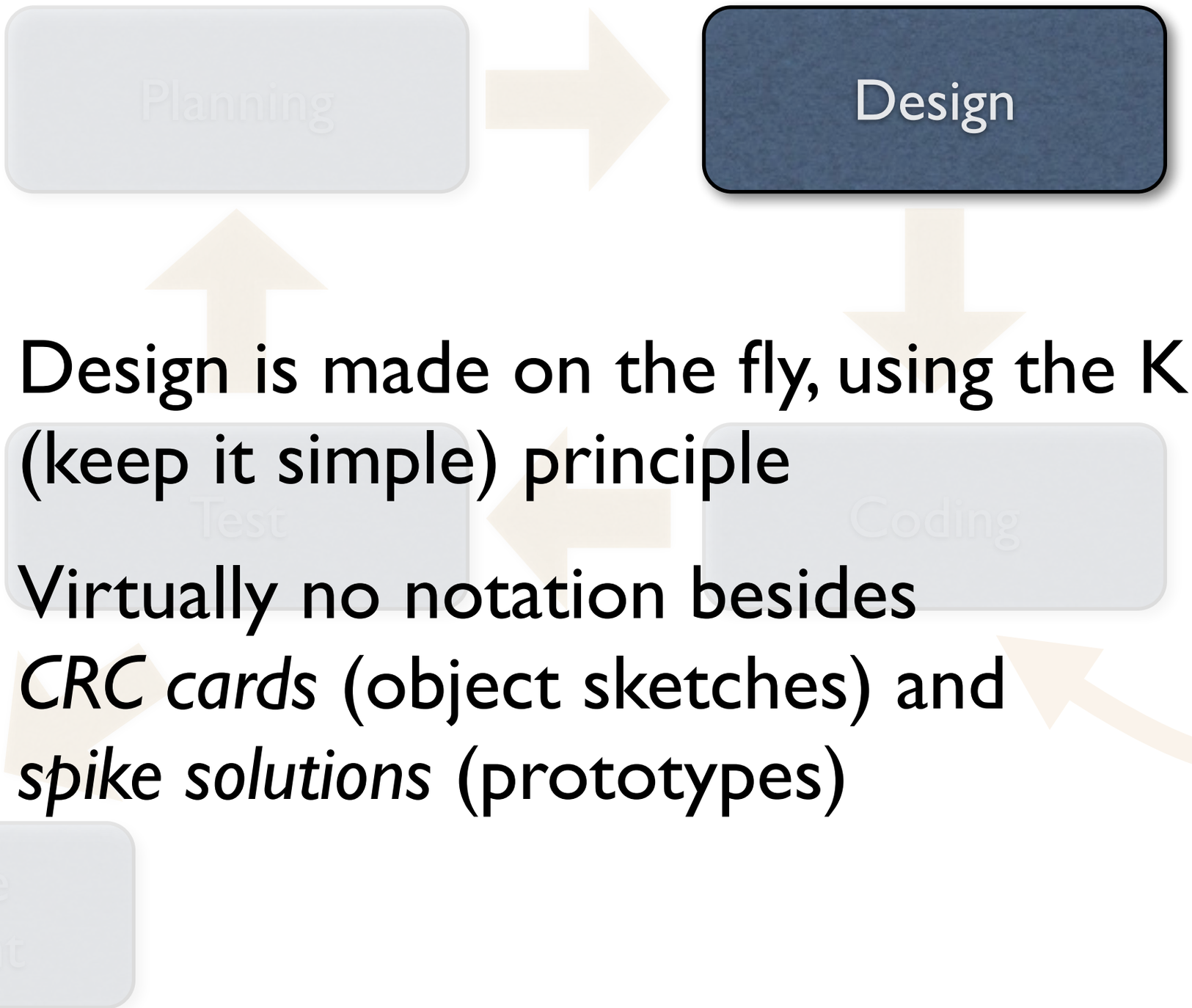


- In XP, planning takes place by means of *stories*
- Each story captures essential behavior

Extreme Programming

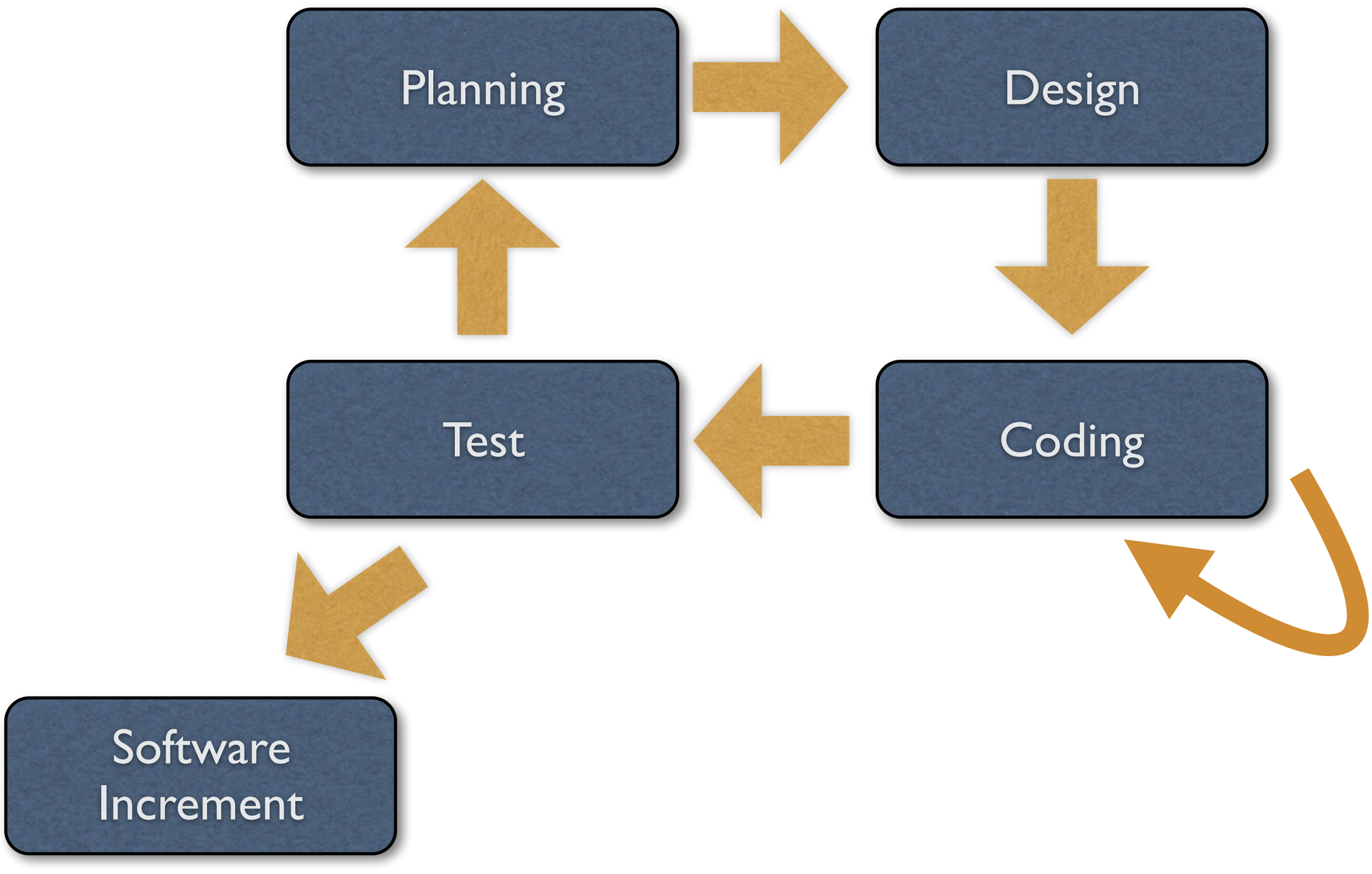


Extreme Programming



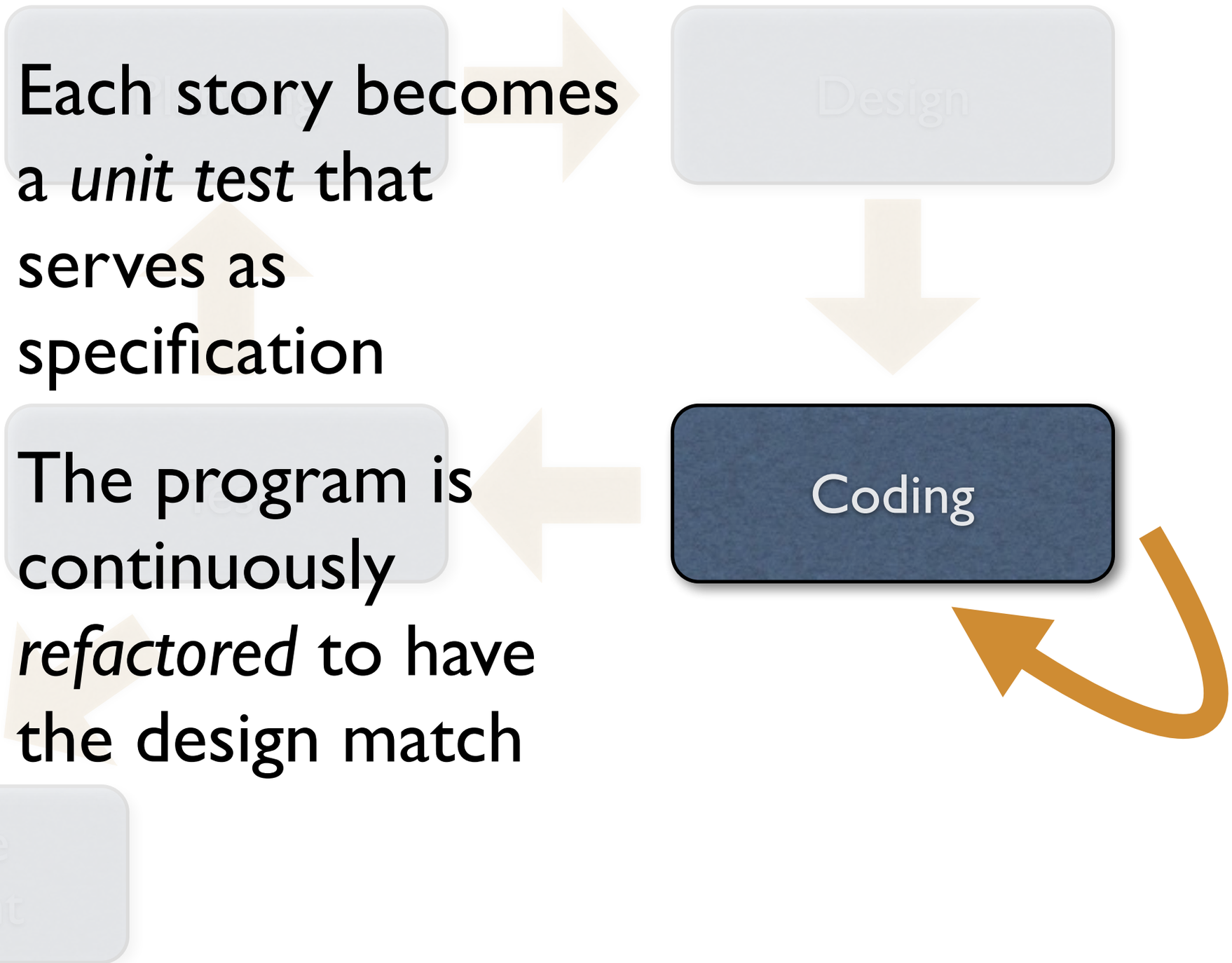
- Design is made on the fly, using the KISS (keep it simple) principle
- Virtually no notation besides *CRC cards* (object sketches) and *spike solutions* (prototypes)

Extreme Programming



Coding

- Each story becomes a *unit test* that serves as specification
- The program is continuously *refactored* to have the design match

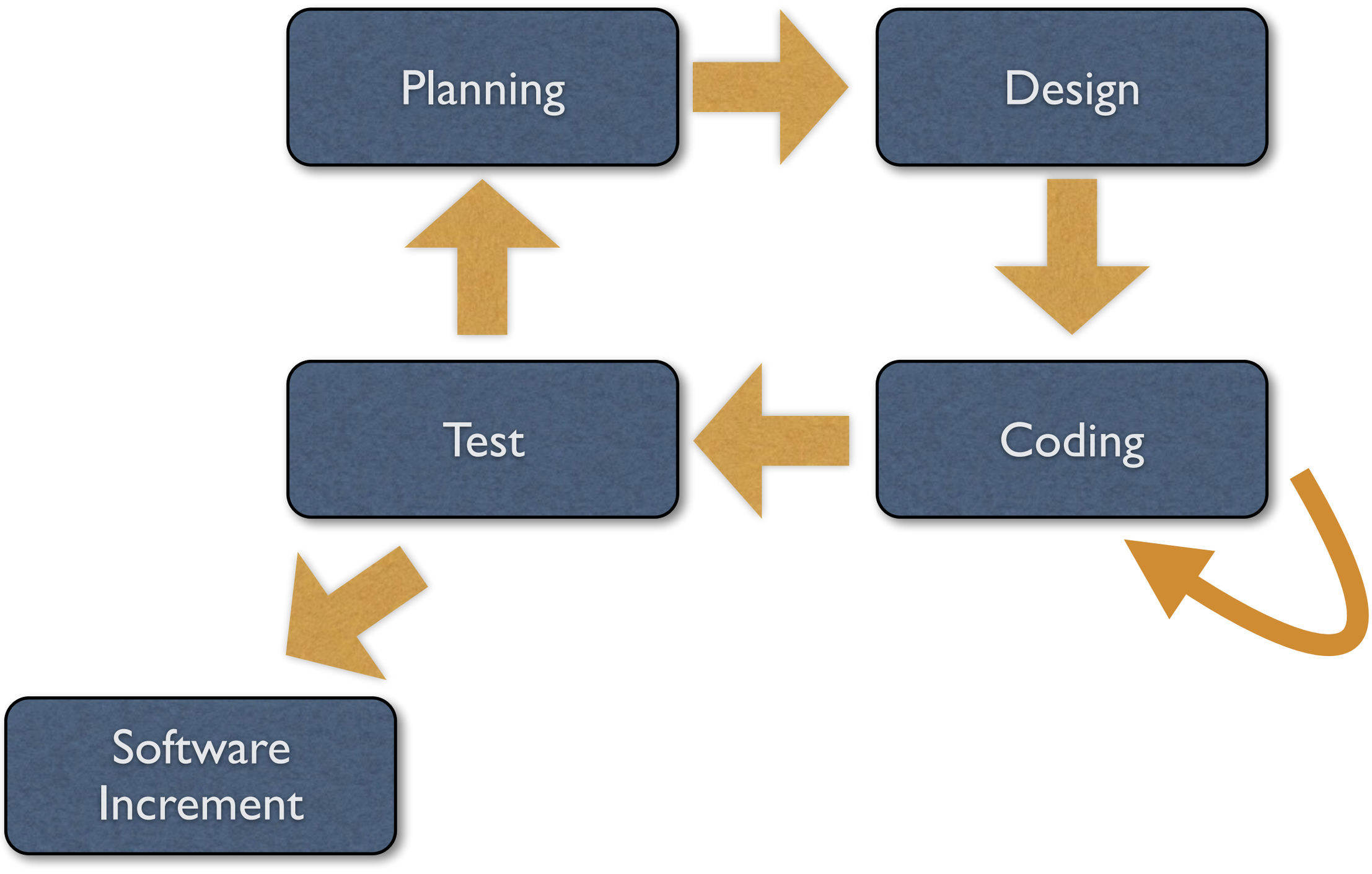


Coding

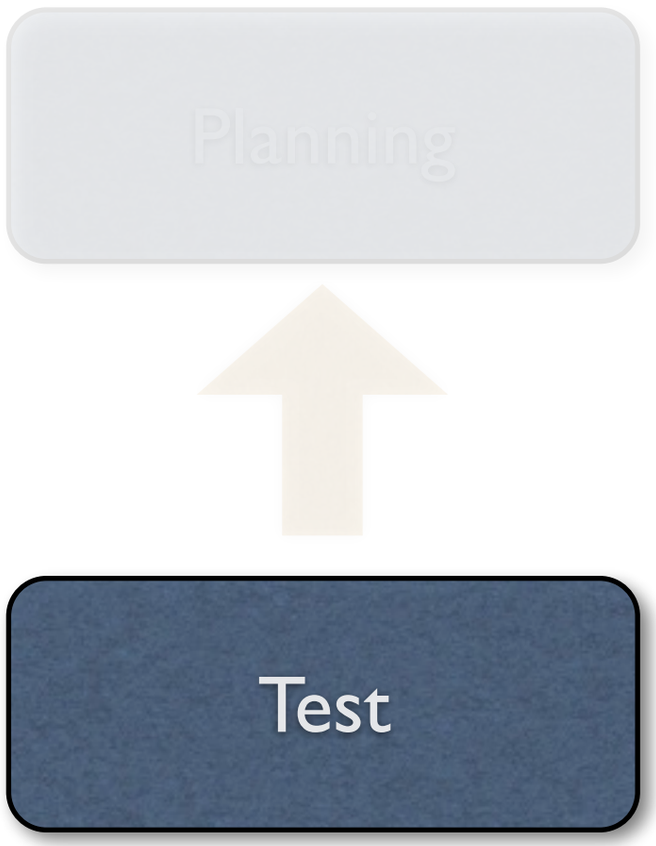
- To ensure continuous review, XP mandates *pair*

Coding

Extreme Programming



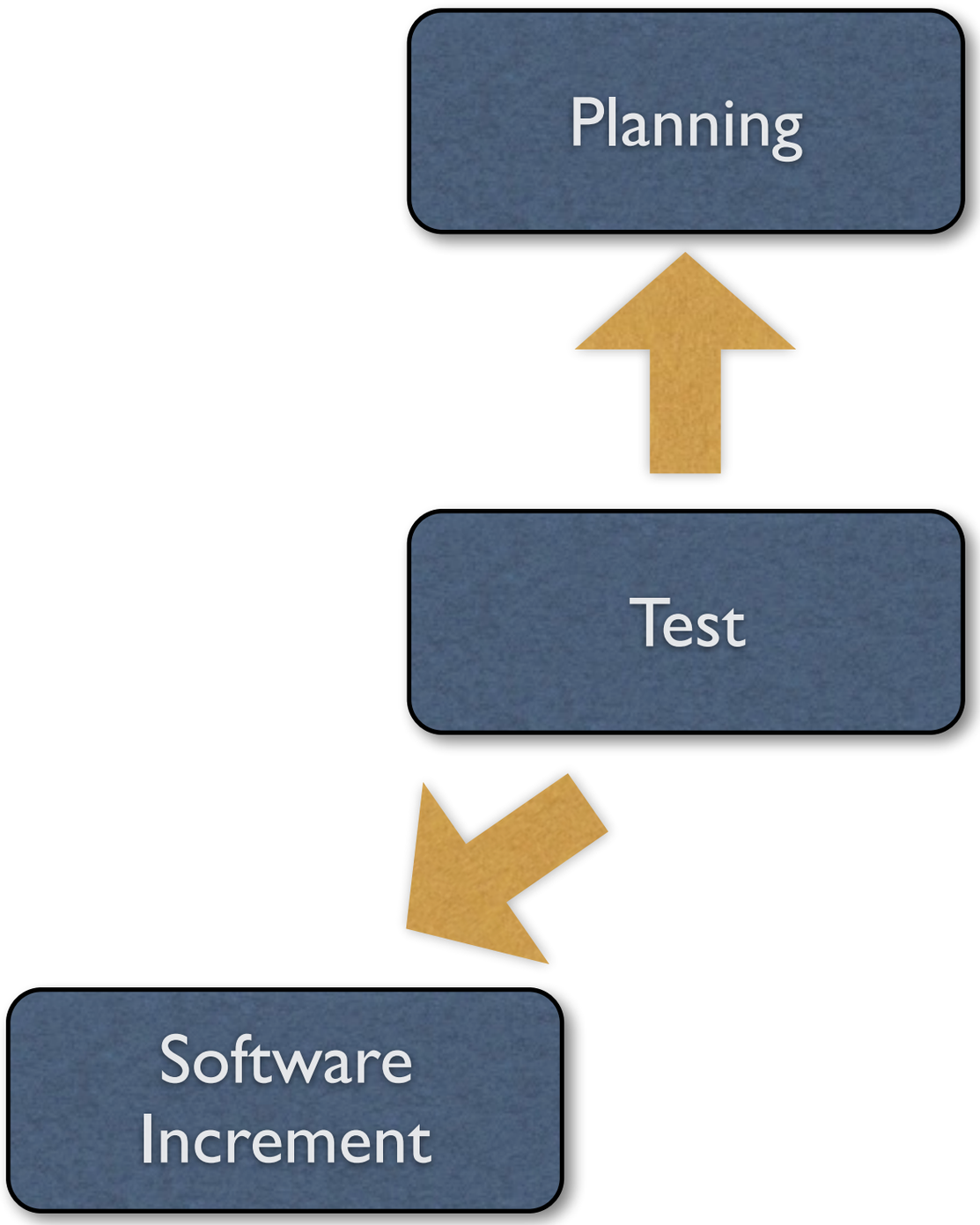
Testing



Unit tests

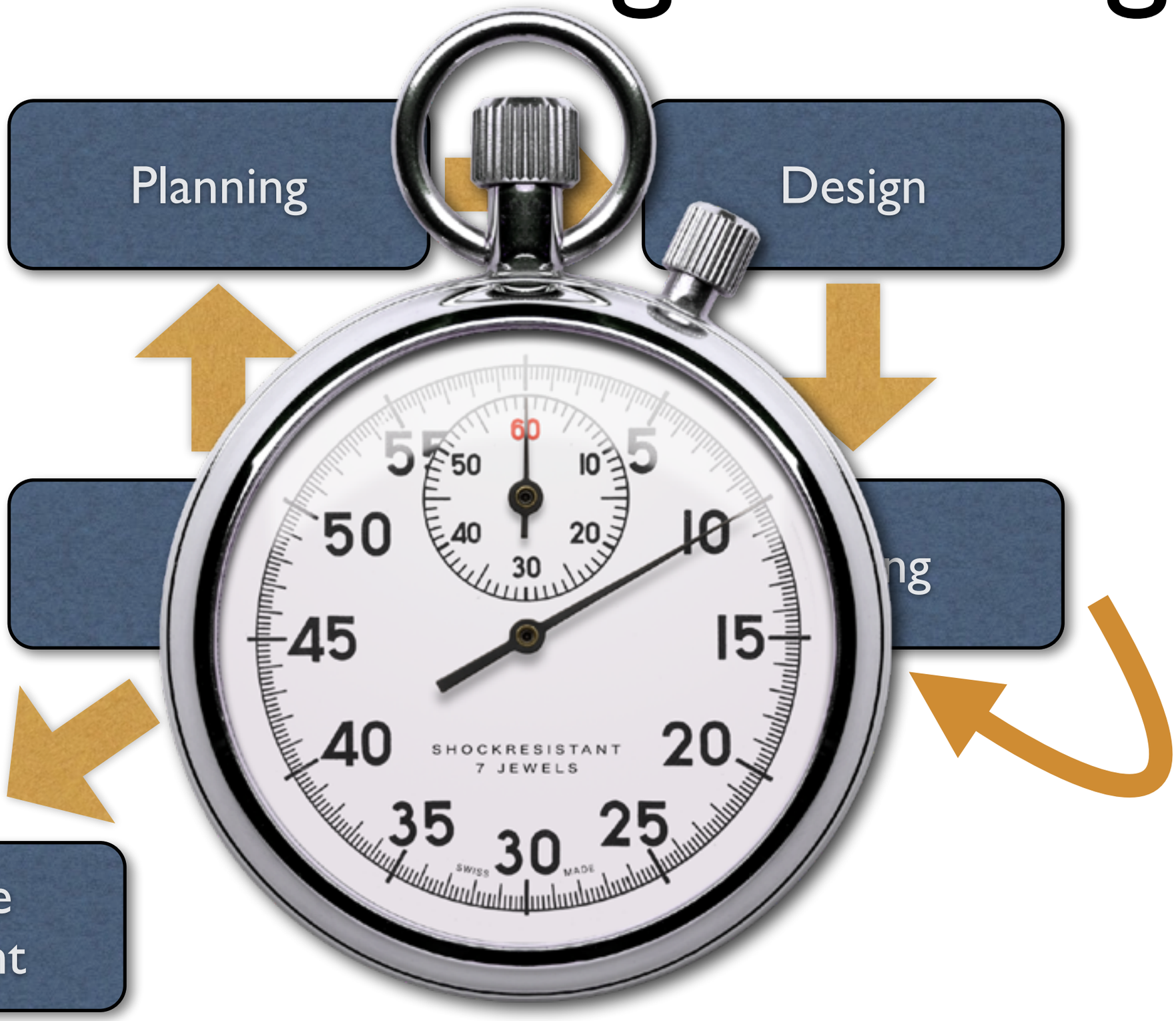
- detect errors
- find missing functionality
- measure progress

Extreme Programming

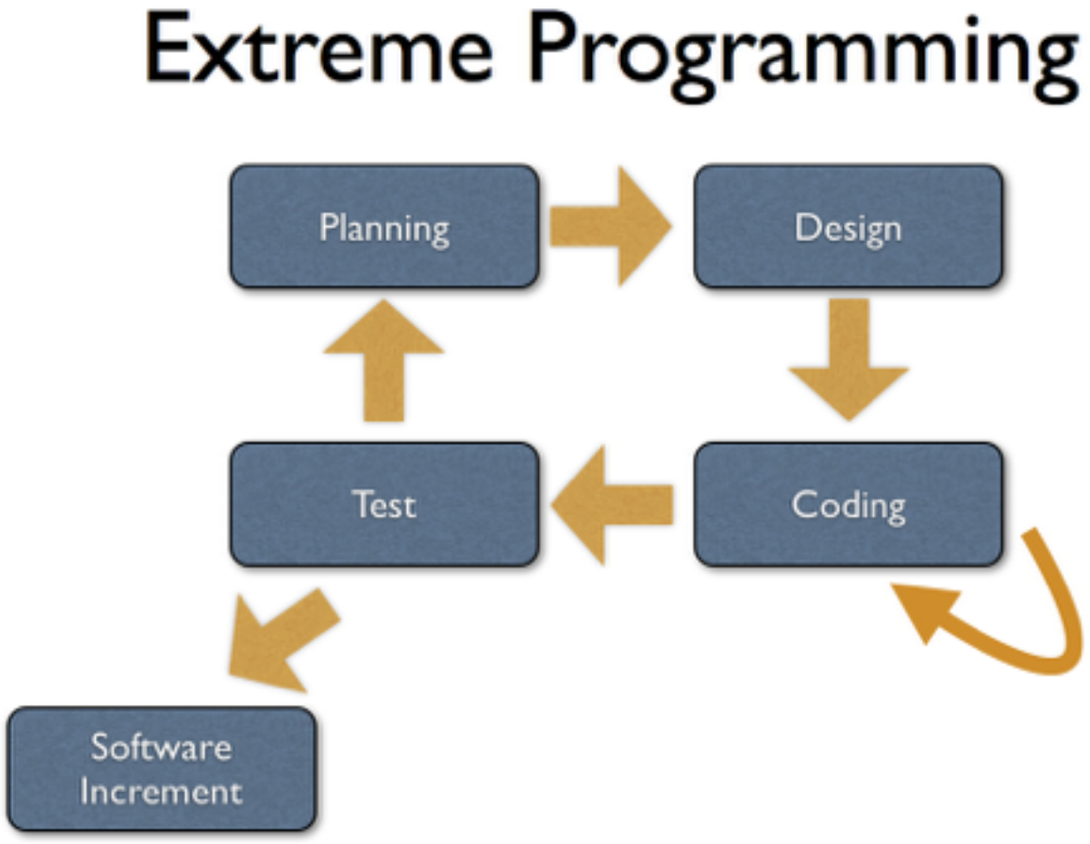


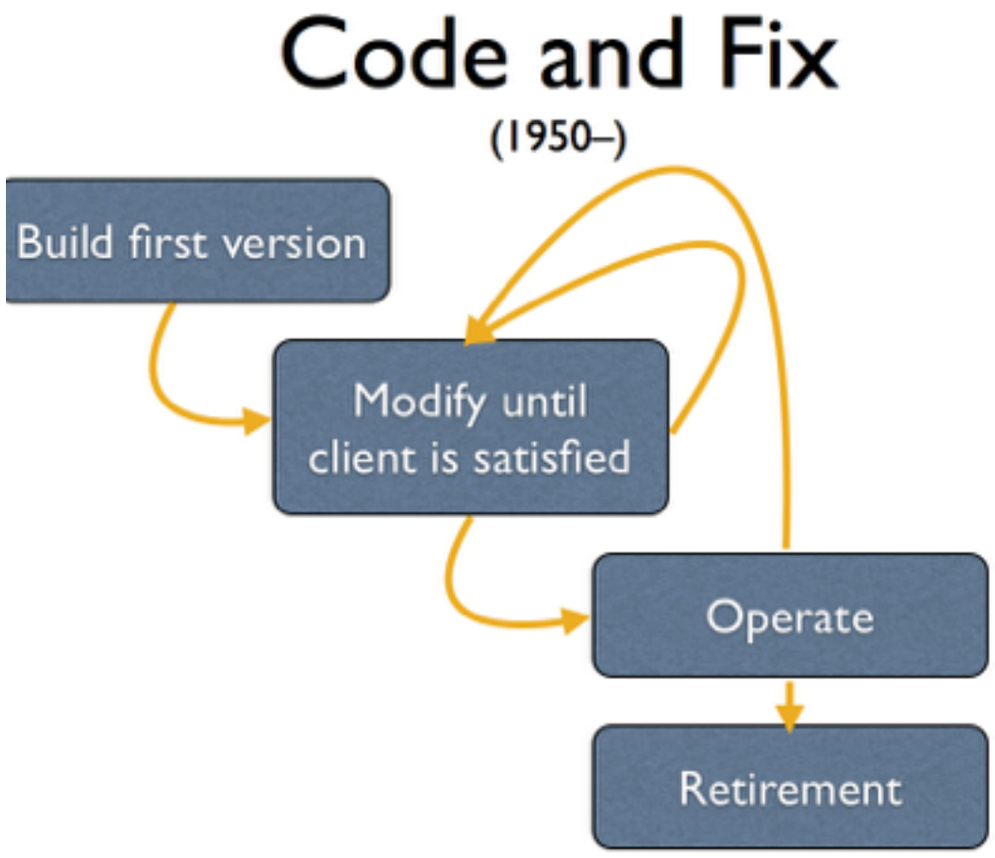
- The resulting prototypes result in new stories

Extreme Programming



Spot the Difference





Scrum

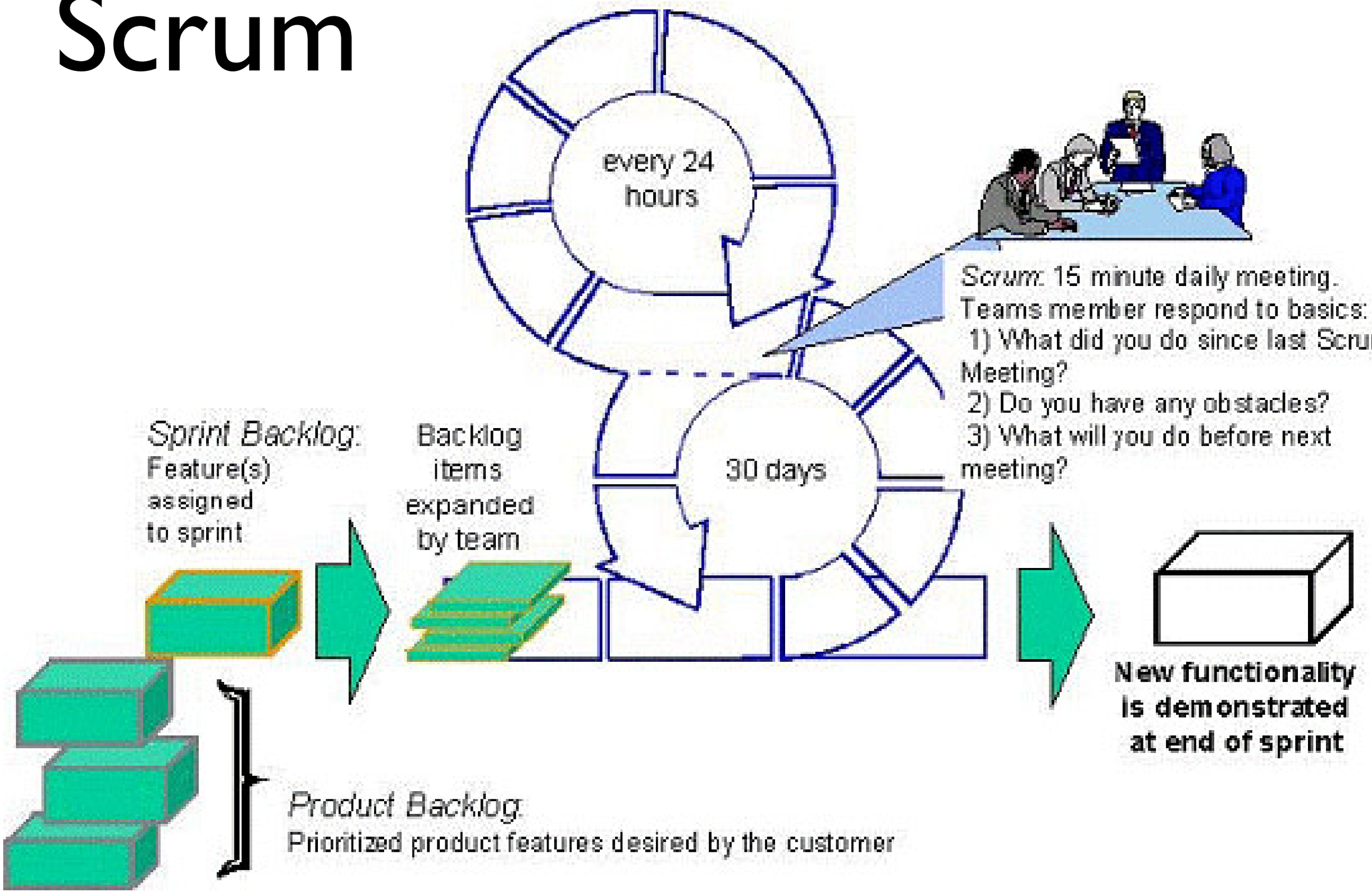
Scrum

- An iterative and incremental agile software development method for managing software projects and product or application development.
- Small working teams to maximize communication, minimize overhead and maximize knowledge sharing.
- Adaptable to technical and business changes.
- Yields frequent software increments that can be inspected.

Scrum

- Development work and the people who perform it are partitioned into clean, low coupling partitions.
- Constant testing and documentation is performed.
- Ability to declare project “done” whenever required.

Scrum


every 24 hours
30 days
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Scrum

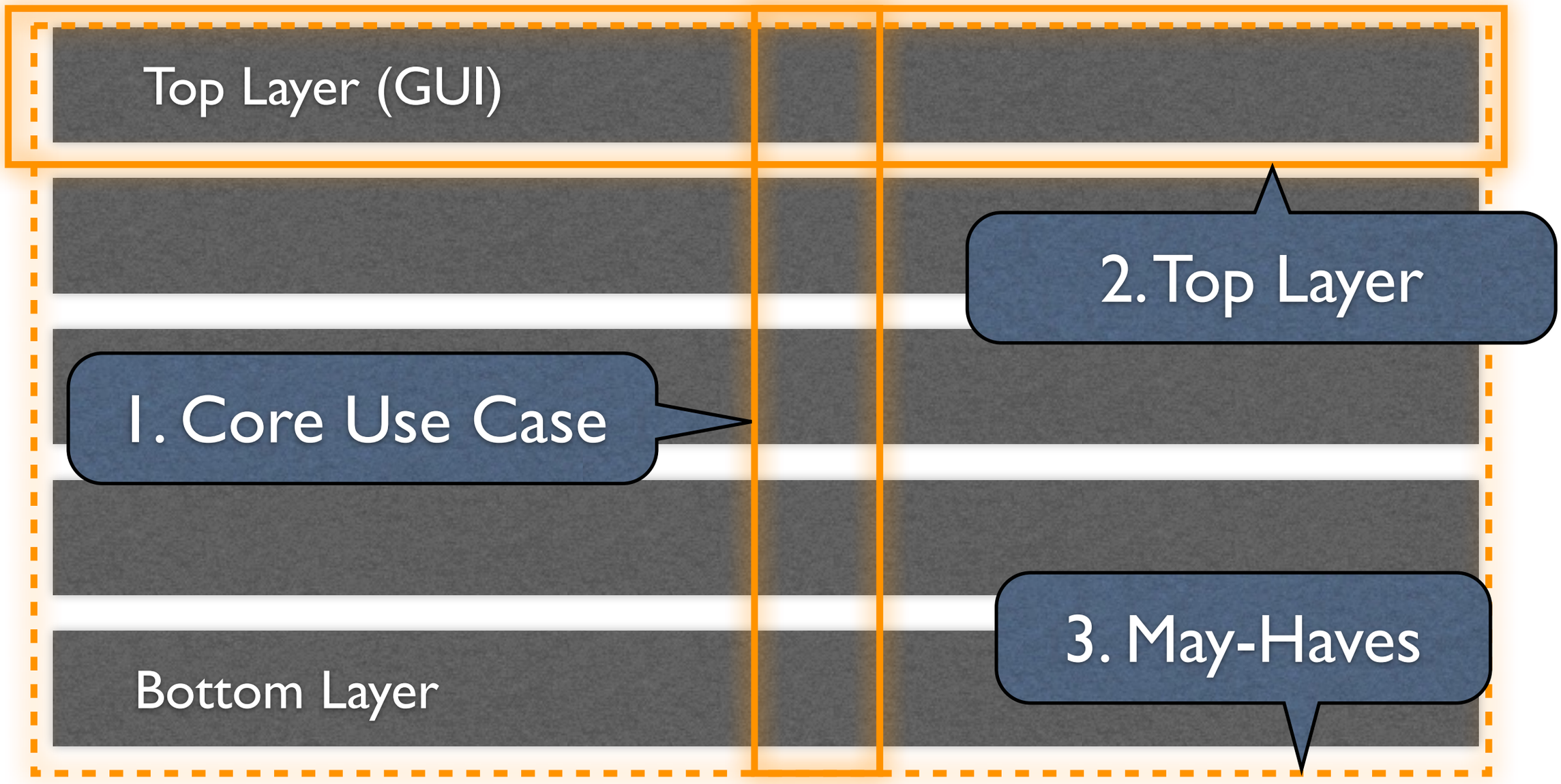
Backlog: A prioritized list project requirements or features that provide business value.

Sprints: Consists of work units that are required to achieve a defined backlog into a predefined time-box (usually 30 days).

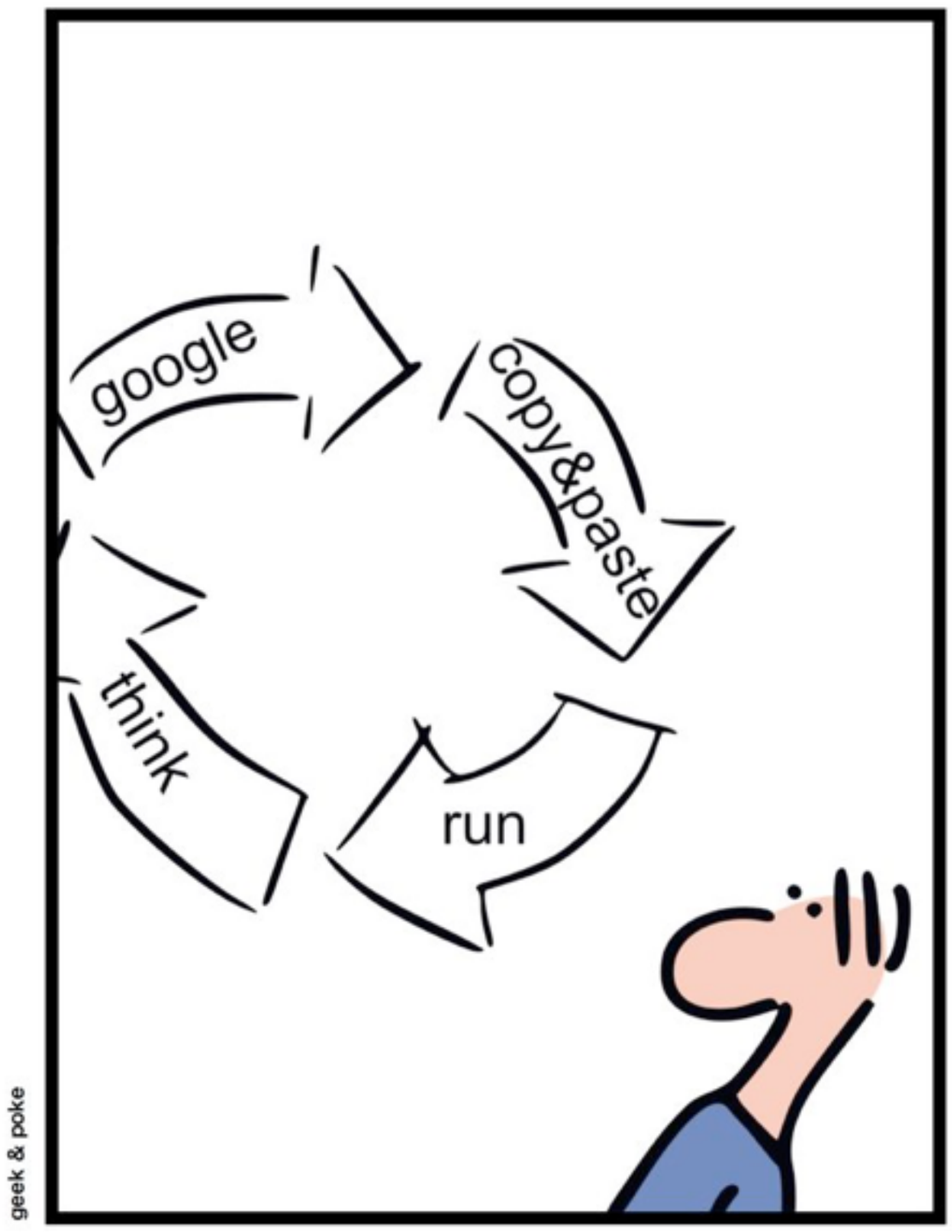
Scrum Meetings: Short 15 mins. meetings held daily by the scrum team. The Scrum master leads the meeting.

Demos: Demonstrate software increment to the customer for evaluation.

Your Sprints

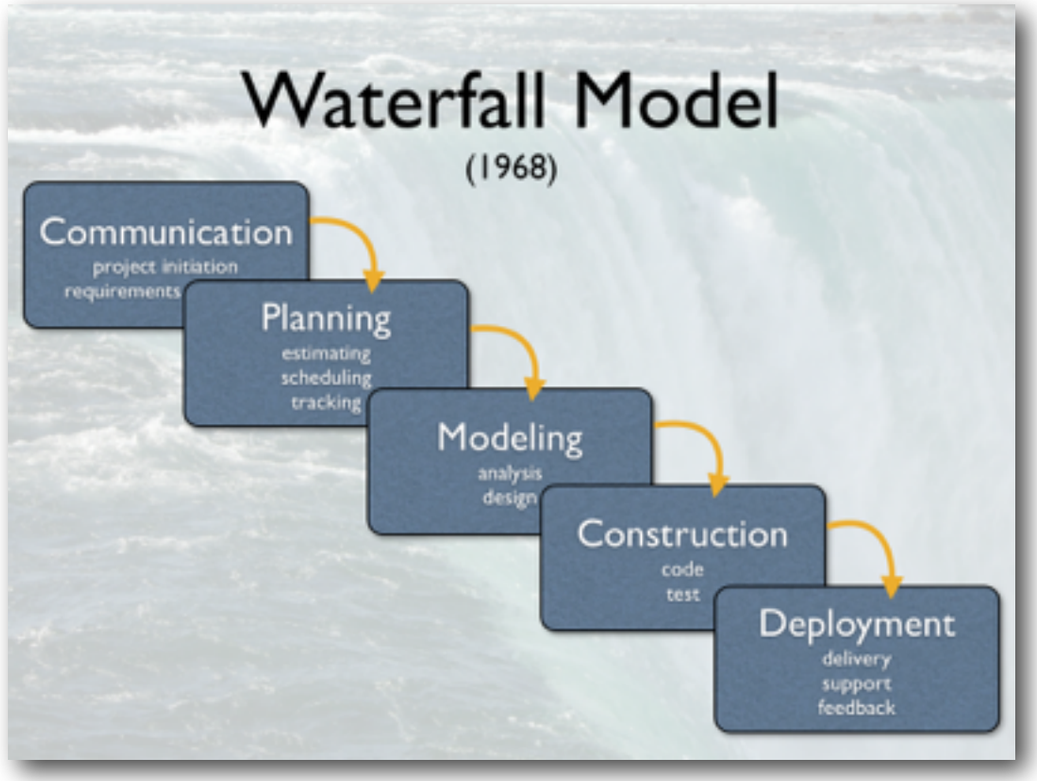


SIMPLY EXPLAINED

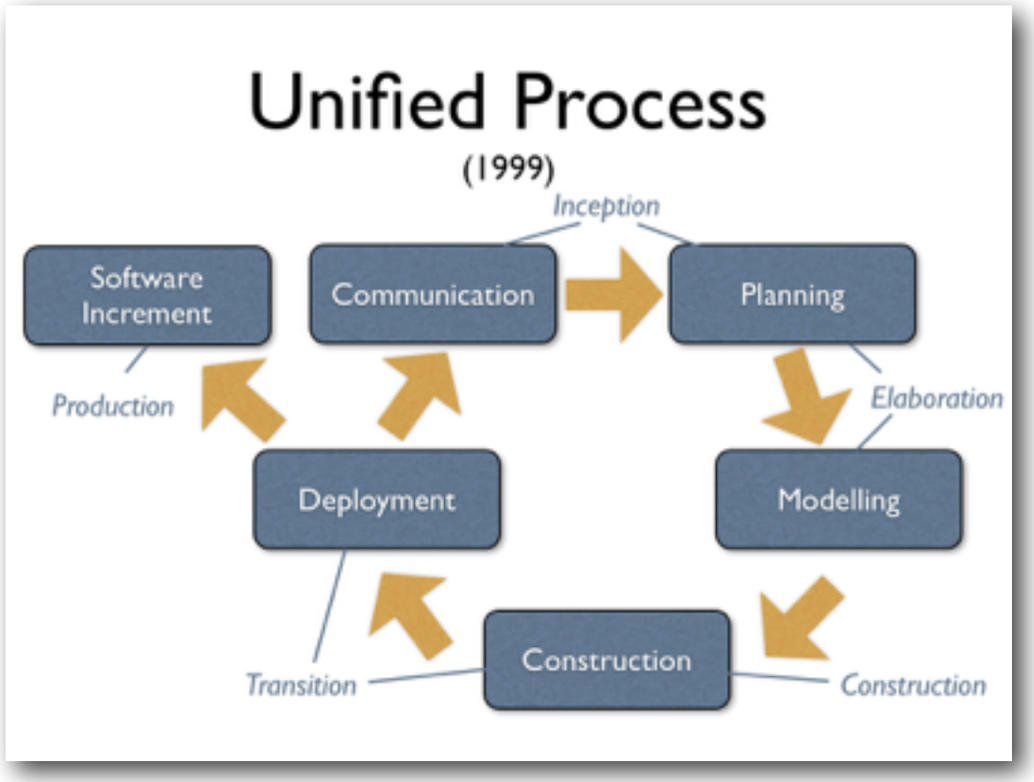


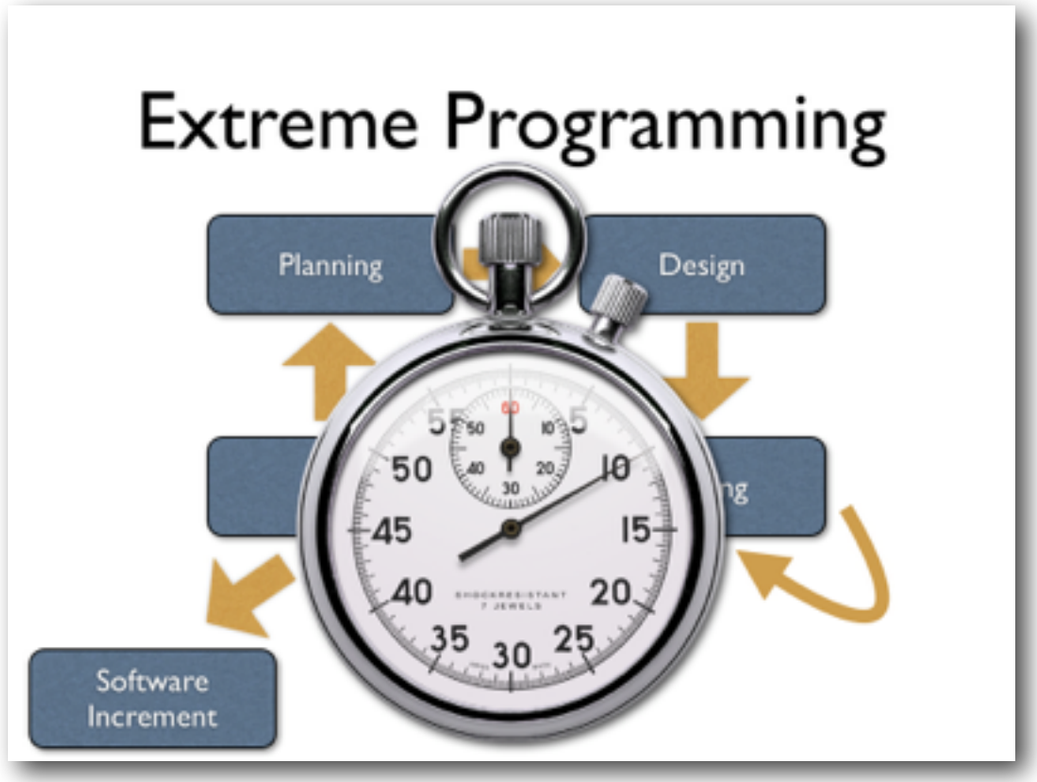
DEVELOPMENT CYCLE


Code and Fix
CODING HORROR


Waterfall Model
(1968)
Communication
project initiation
requirements
Planning
estimating
scheduling
tracking
Modeling
analysis
design
Construction
code
test
Deployment
delivery
support
feedback

Summary


Unified Process
(1999)
Inception
Software Increment
Communication
Planning
Production
Elaboration
Deployment
Modelling
Transition
Construction
Construction


Extreme Programming
Planning
Design
Software Increment