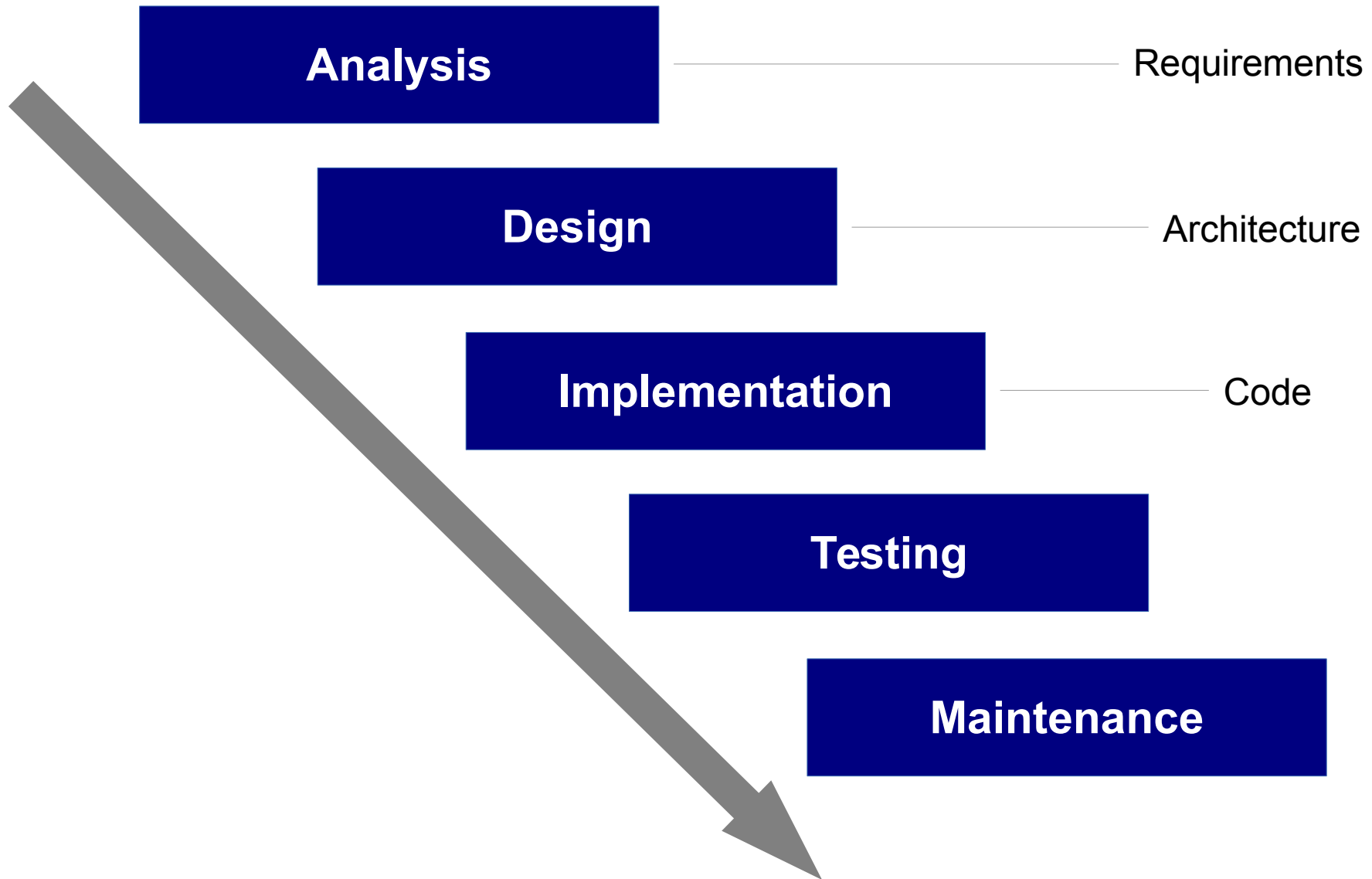


# Iterative and Incremental development

01219245/01219246  
Individual Software Process  
Spring Semester 2014

# Traditional Waterfall Model



# Problems (1)

- Very hard to get all requirement up-front

# Problems (2)

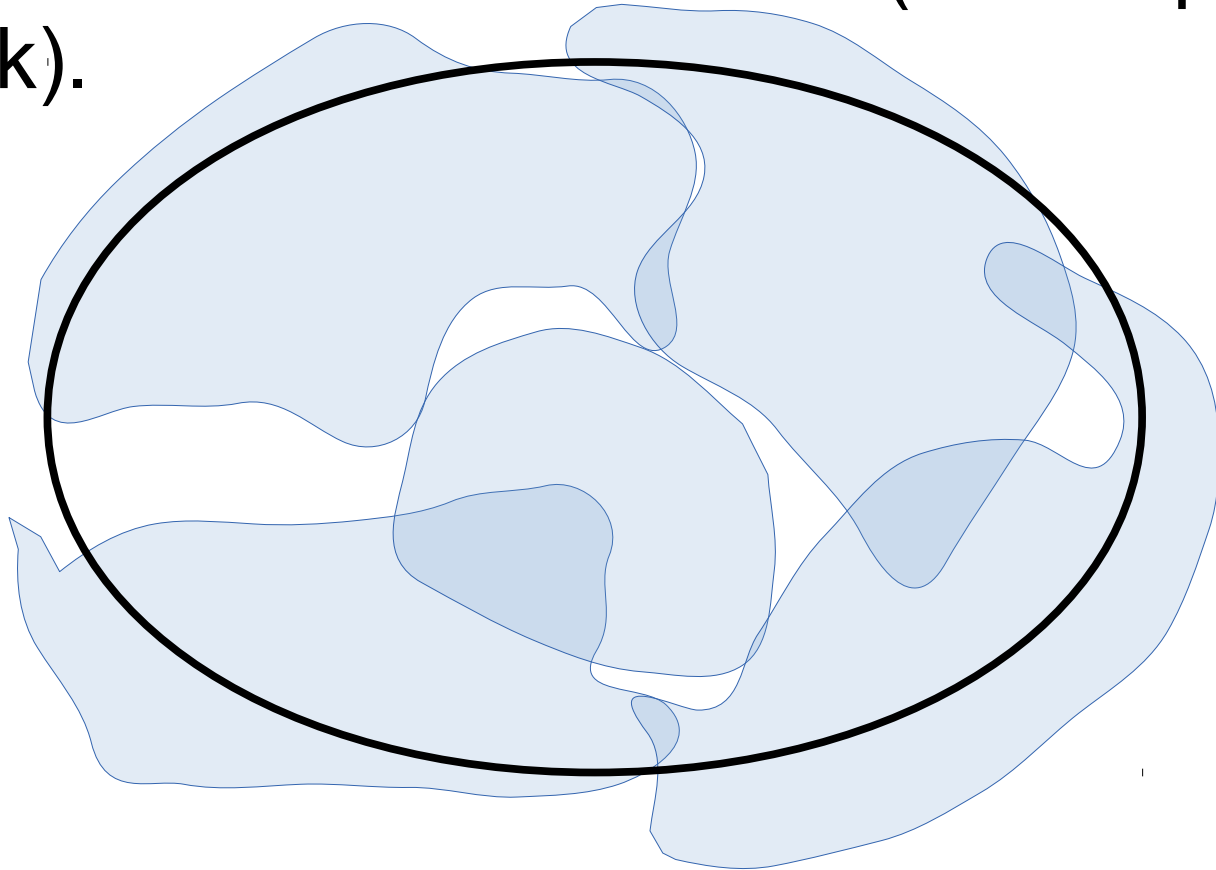
- Requirement changes
  - Changes in business condition
  - Knowledge gained from the development/deployment

# Problems (3)

- Quality problem
  - Testing at the end might not be sufficient
  - Before testing begins, the team has no clear idea how the project progresses.

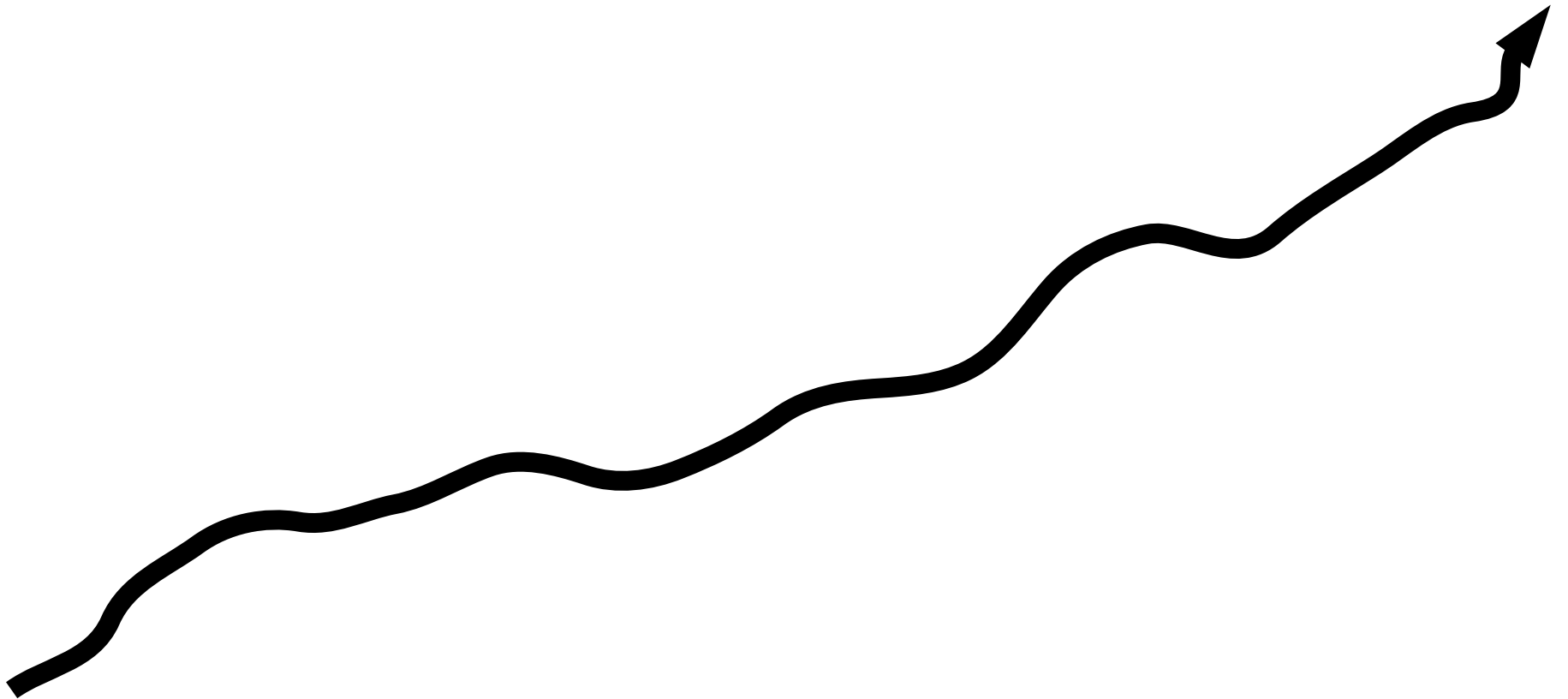
# Big-bang integration

- This is a classic recipe to failure.
- Work on many (large) parts of the project and wait to combine them at the end (and hope that it will work).



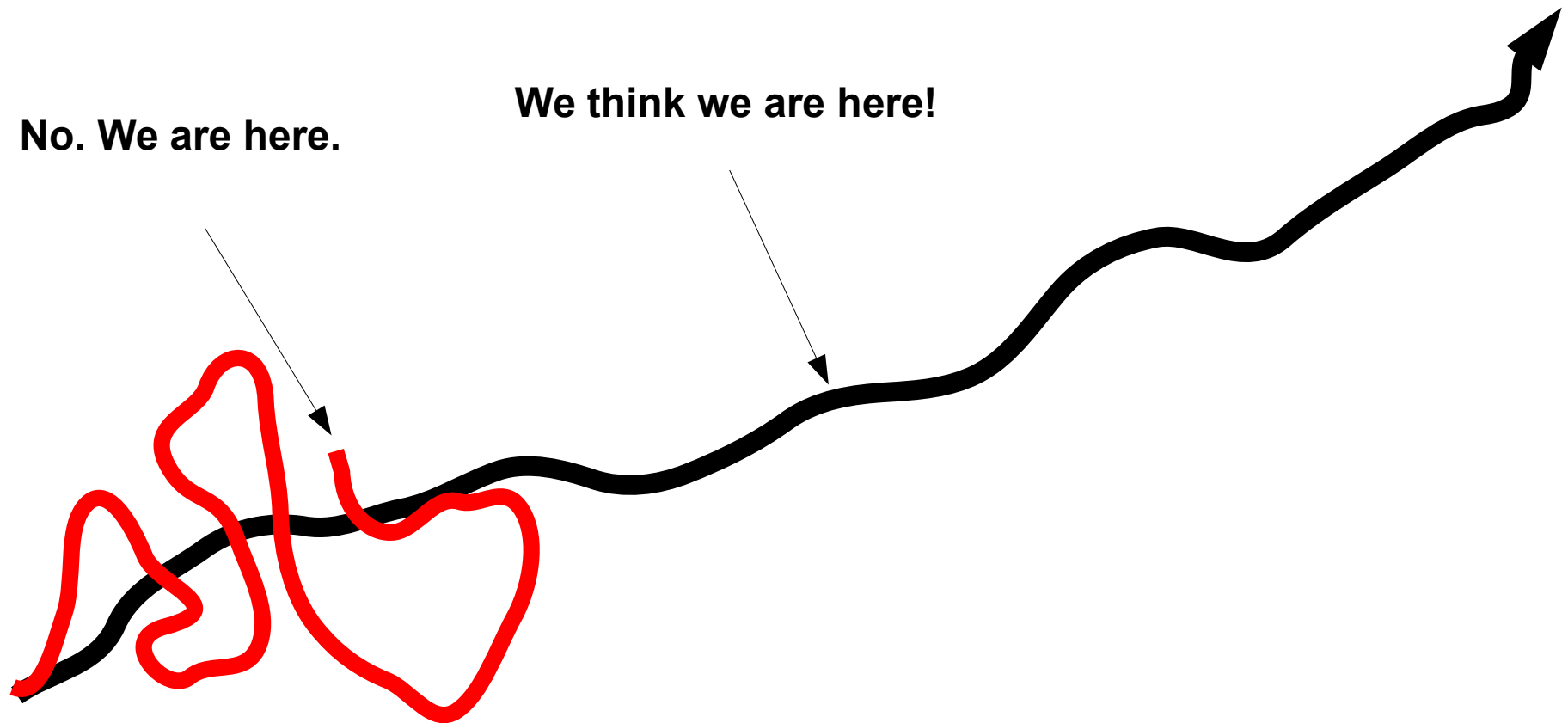
# Software Development as a Journey

- You have (unclear) goals, with limited resource.



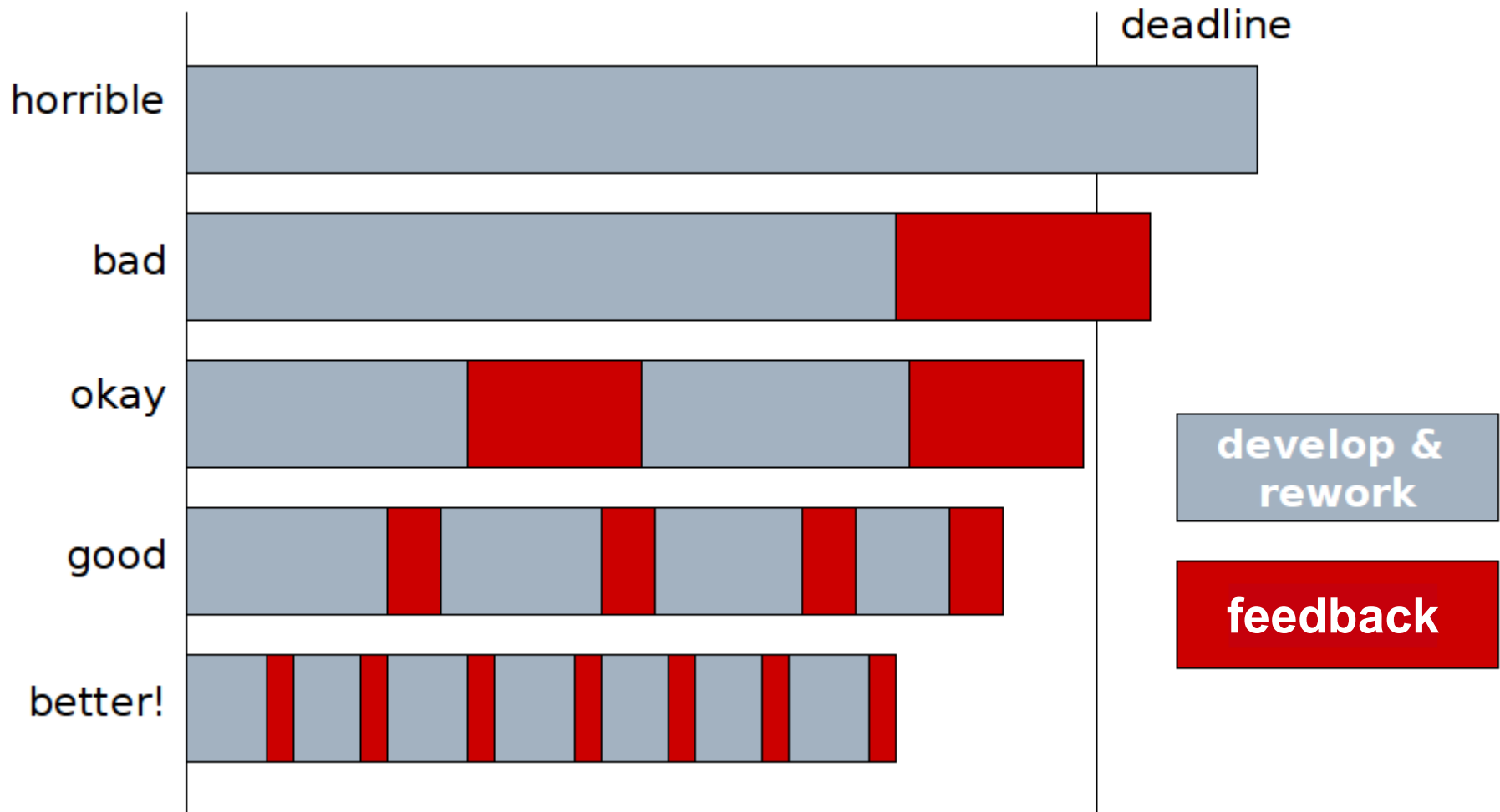
# Software Development as a Journey

- You have (unclear) goals, with limited resource.



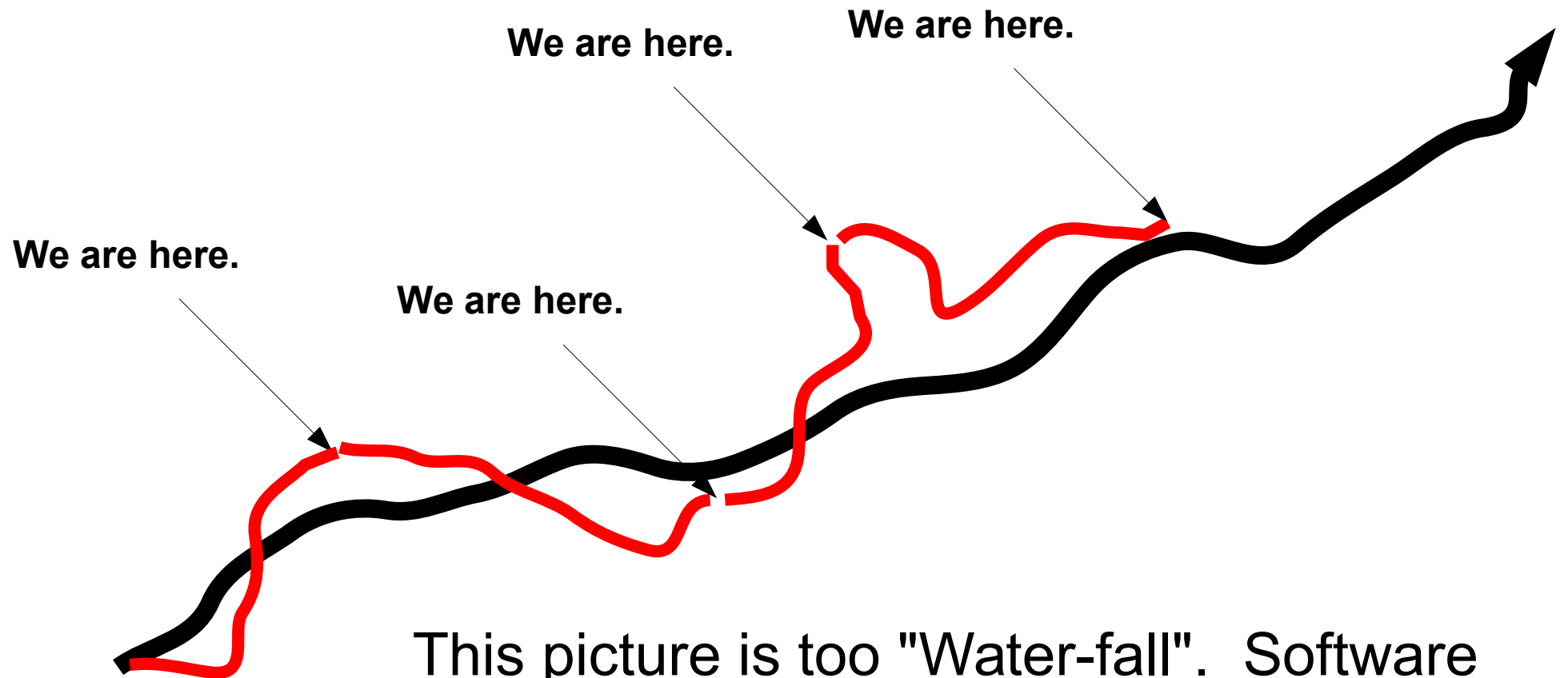


# Early feedback



# Software Development as a Journey

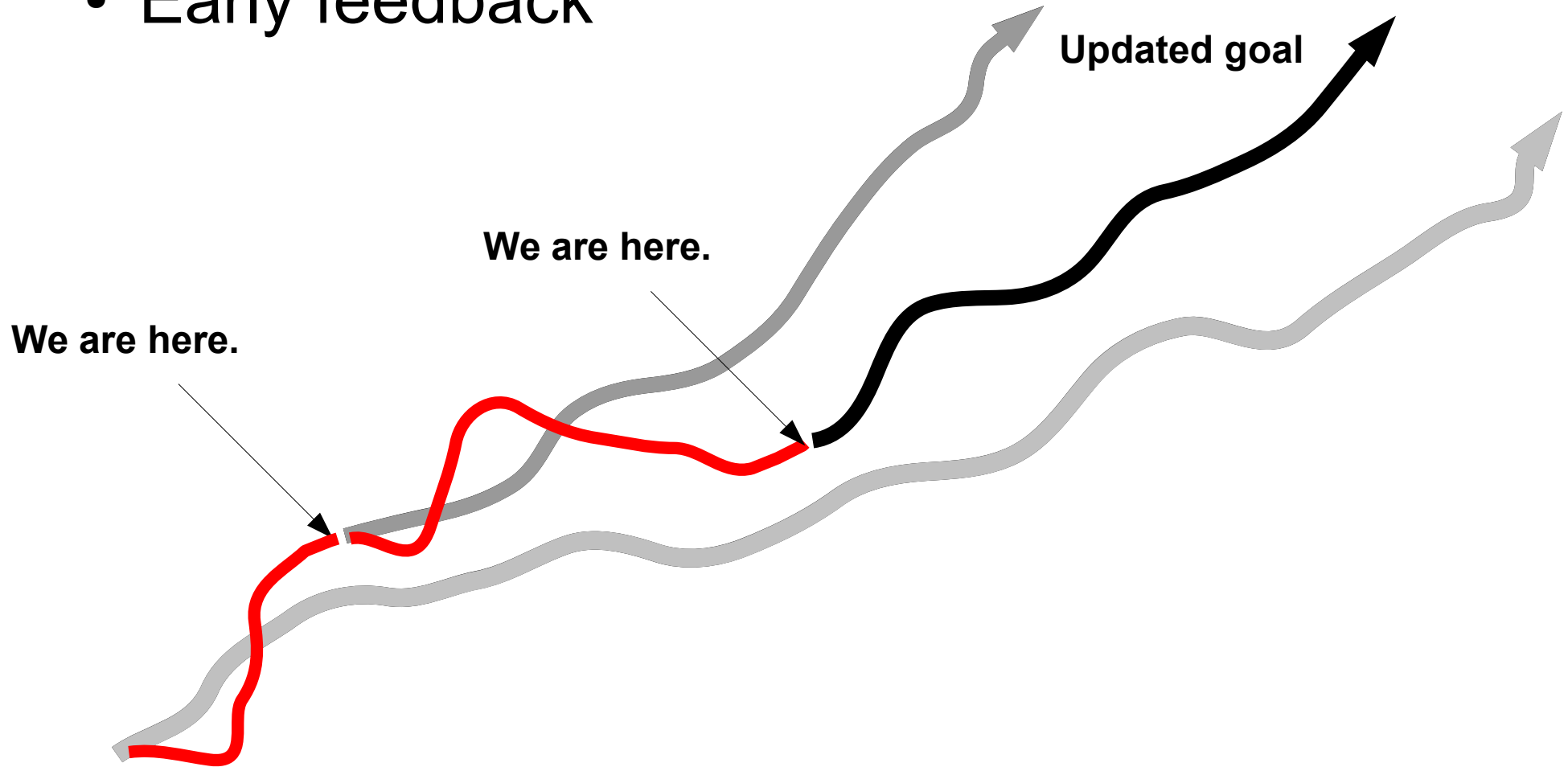
- Early feedback



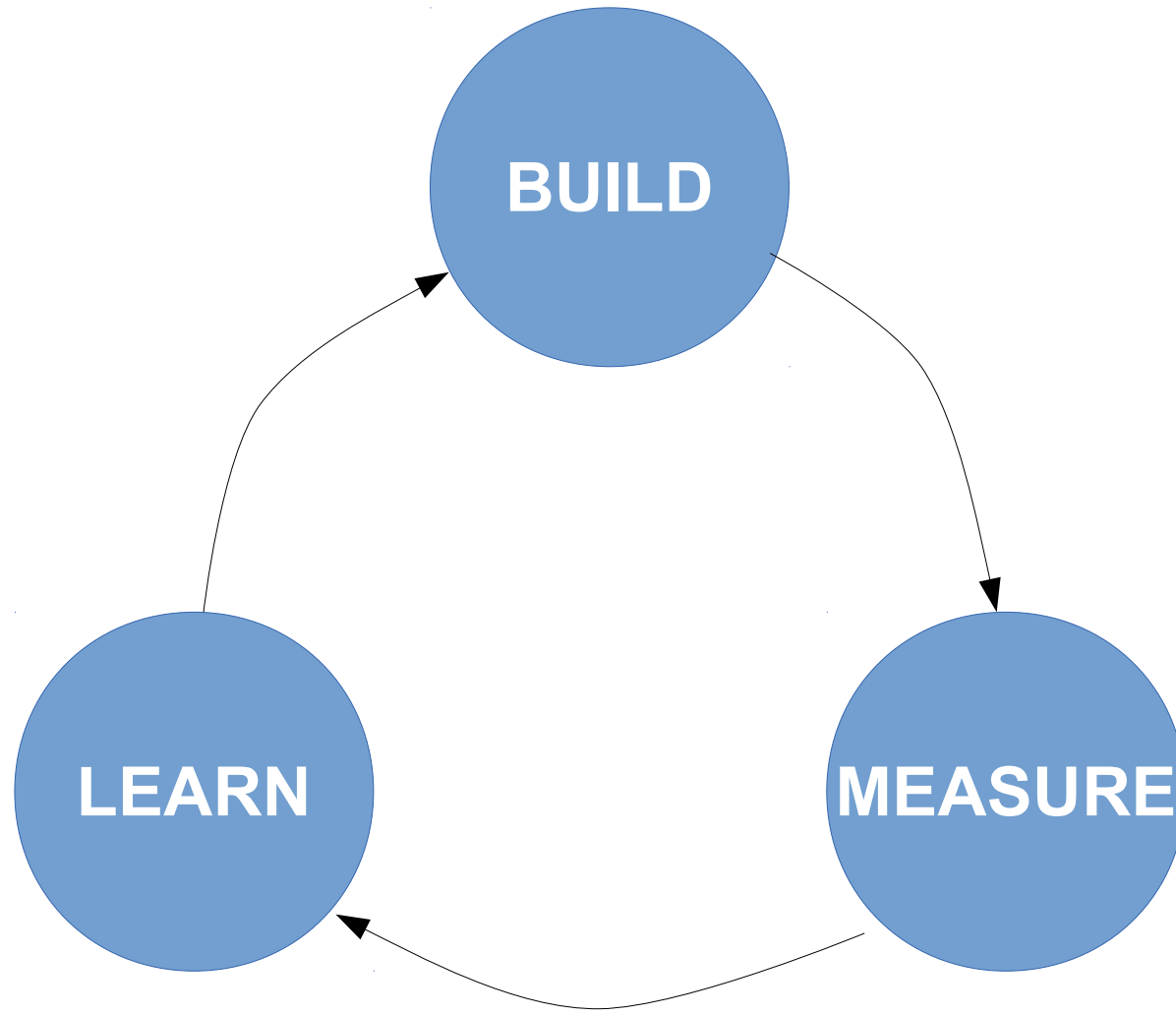
This picture is too "Water-fall". Software development rarely has clear, defined goals.

# Software Development as a Journey (Agile view)

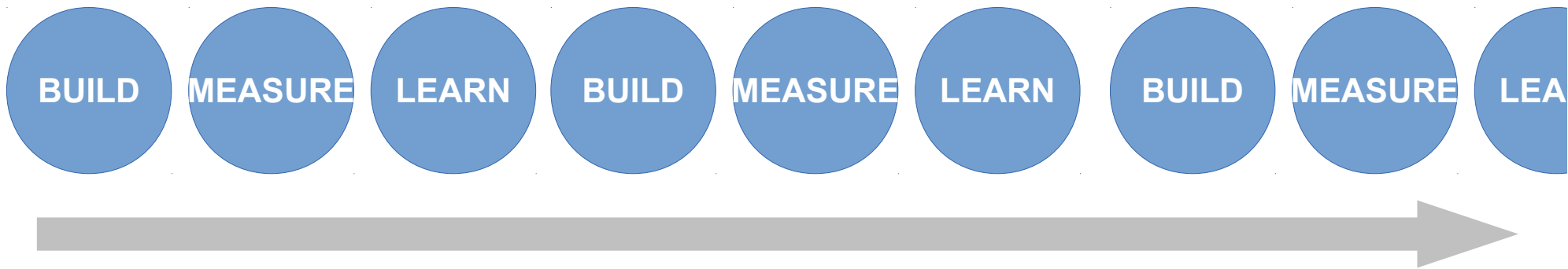
- Early feedback



# The feedback loop: lean startup



# The feedback loop: lean startup



- Continuous building and learning loop

# How can you get feedback?

- Are we building the right product?
- Talk with the customer:
  - **Developer:** This is my piece of code (500 lines) that still doesn't do anything much. Can you give us feedback?
  - **Customer:** ???
- You need to be able to show a concrete unit of work to the customer.

# How can you get feedback?

- Are we building the product right?
- Software Quality (given that the requirement is known and is clear)
- Use various testing activities
  - unit testing
  - automated functional testing
  - manual testing

# Build

- What to build?
- When to build?
- How to build?



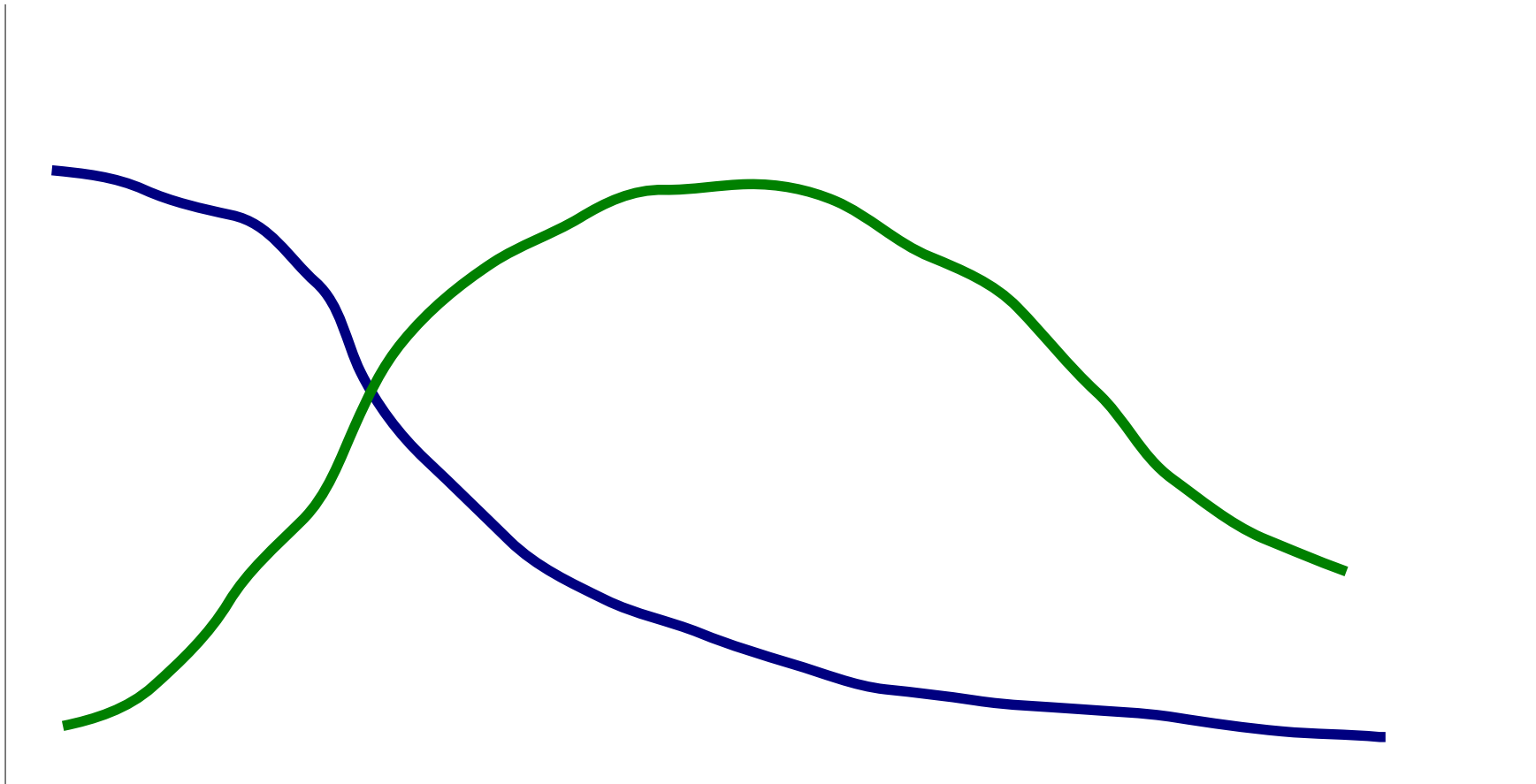
# What to build?

- **Small**
  - doesn't take too much time to build
- **Measurable**
  - should be able to obtain feedback after we build
- **Goal-based**
  - should provide us with the right kind of feedback

# Many kinds of feedback

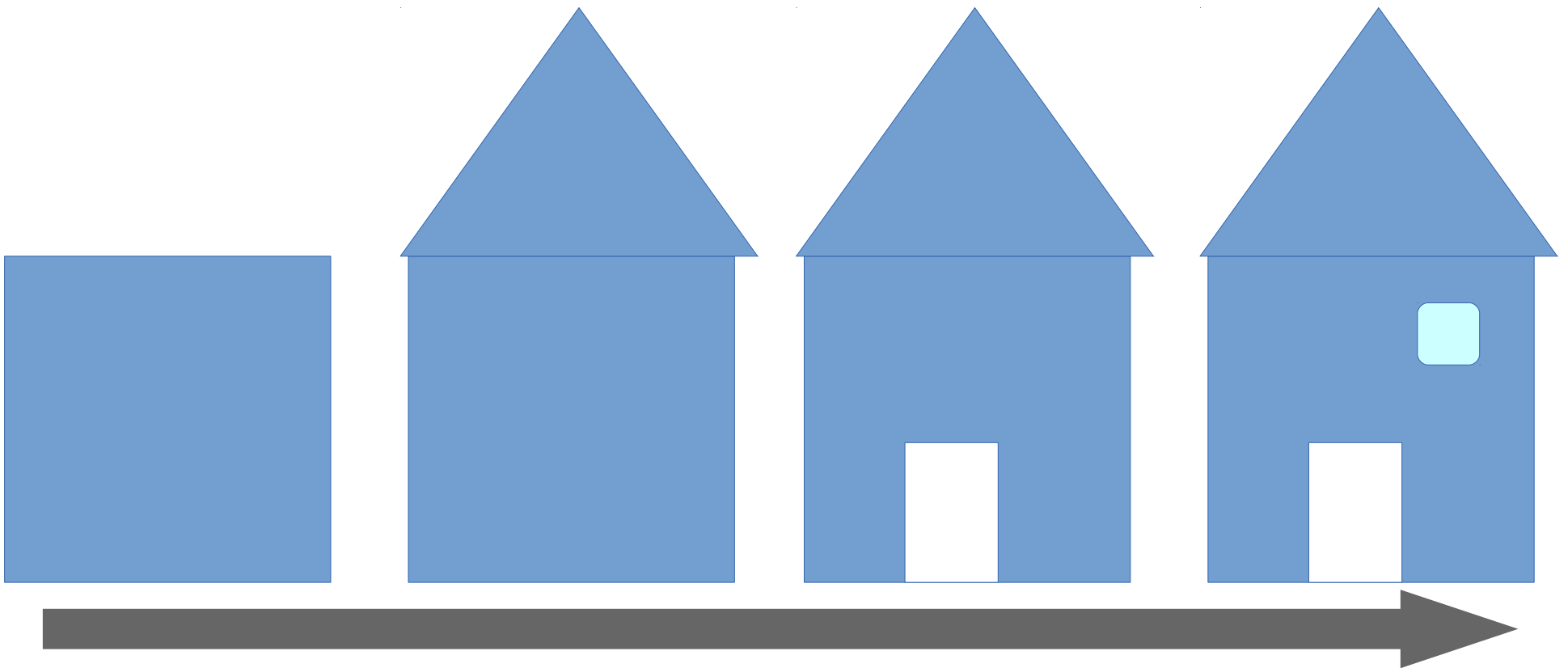
- **Functionality**
  - Does our product solve the user's problems?
  - Does it produce correct answers?
- **Usability**
  - Is it easy to use?
  - Does it look nice? Is it beautiful?
- **Efficiency**
  - Can the product support 100 concurrent users?
- **Security, Compatibility, etc**

# Different feedback needed at different time



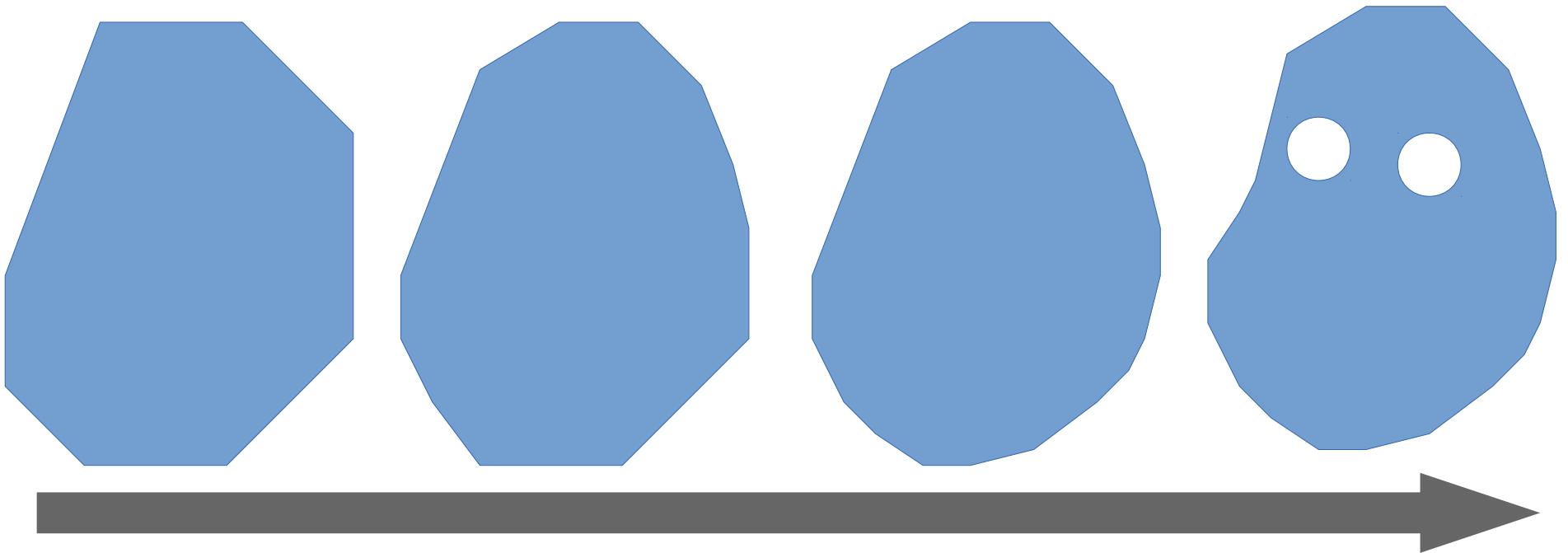
# Incremental development

- To build the whole thing, we work by building its parts, one by one.



# Iterative development

- We repeatedly build something in rounds.
- We know that we will not get it right the first time, but we keep working on it many times.



# Tasks, tasks, tasks

- Breakdown the whole project into a set of tasks
- Think about incremental development
  - what are the parts that you need to build?
- Think about iterative development
  - for each part, what are the smaller steps we can take to complete it?

# Complete?

- The goal is not to have a complete task breakdown.
  - (That's nearly impossible, and usually leads to a waste of time and effort.)
- But to have enough product idea to start building.
- The task list will be refined over time.

# Prioritization

- Too many tasks to do, which one to build first?
- Work on the one with the highest "return".
  - What are the possible "returns"?



# Practice

- Return to your group from last week.
- Think about the game that you have chosen.
- Breakdown the tasks you need to do if you want to build that game.
  - Don't try to get all the details in the first round.

# Example: Prince of Persia (Round 1)

- Show map
- Show main character
- Move the main character according to the keyboard input
- Animate the main character movement

# Example: Prince of Persia (Round 2)

- Show map
  - Show map with just one row
  - Show map with many rows with free space for climbing (without the traps)
  - Show map with many rows with free space for falling (without the traps)
  - Show map with moving cutters
  - Show map with moving spikes

# Example: Price of Persia (Round 2)

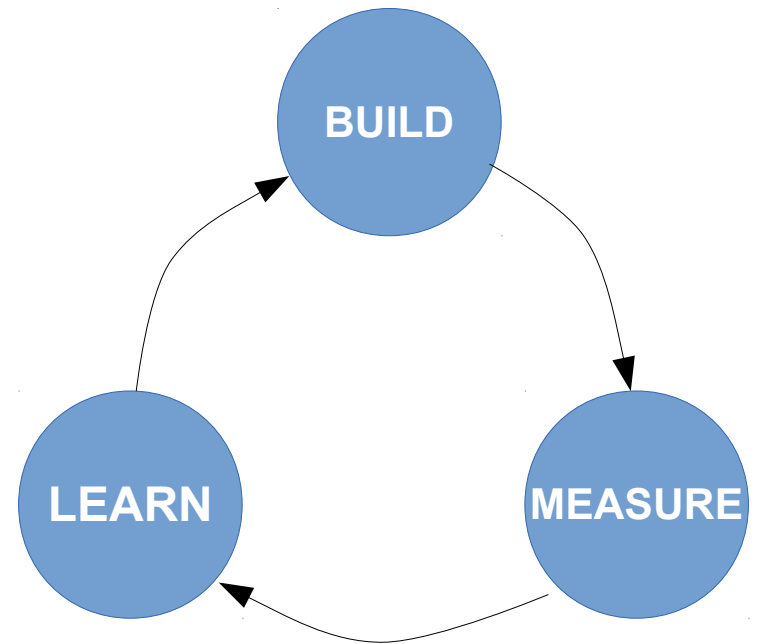
- Move the character
  - Move the character in a simple row
  - Let the character climb the steps
  - Let the character fall if it move to a free space
  - Move the character through moving cutters
  - Move the character through moving cutters and get killed

# Practice

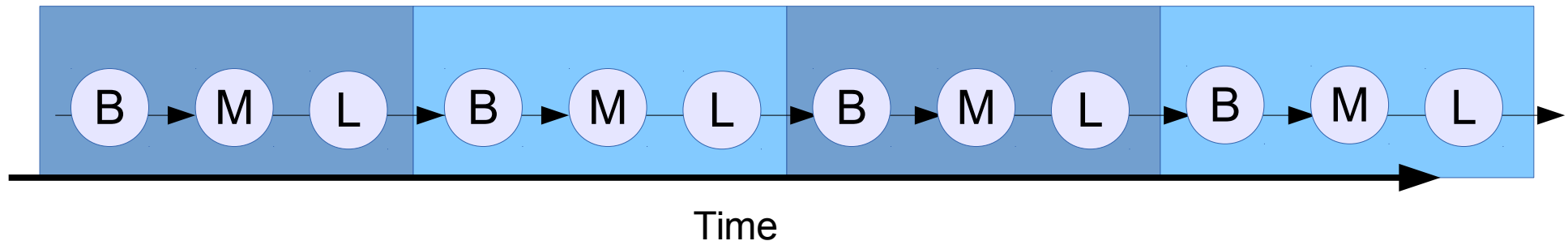
- From the tasks that you have listed, prioritize them.
- We will have discussions after you finish.

# When to build?

- You want to iterate over this loop quickly.
- There are many ways to force you to get moving in this loop.
- Simplest one is to use fixed time-boxing.



# Fixed time-boxing



- Work in a fixed time frame, e.g. one to four week periods.
  - In Scrum, this is called a **sprint**.
- In that period, build something and get feedback.

# How to build?

- We need engineering practices that "support" incremental and iterative development.
- How to make sure that \_\_\_\_\_ over a long period of time.
  - We remember what we did
  - The features that already work remain working
  - The feature can be added
  - The code can be modified, fixed, changed
  - The code is understandable