# Robot Development Cyber and Physical

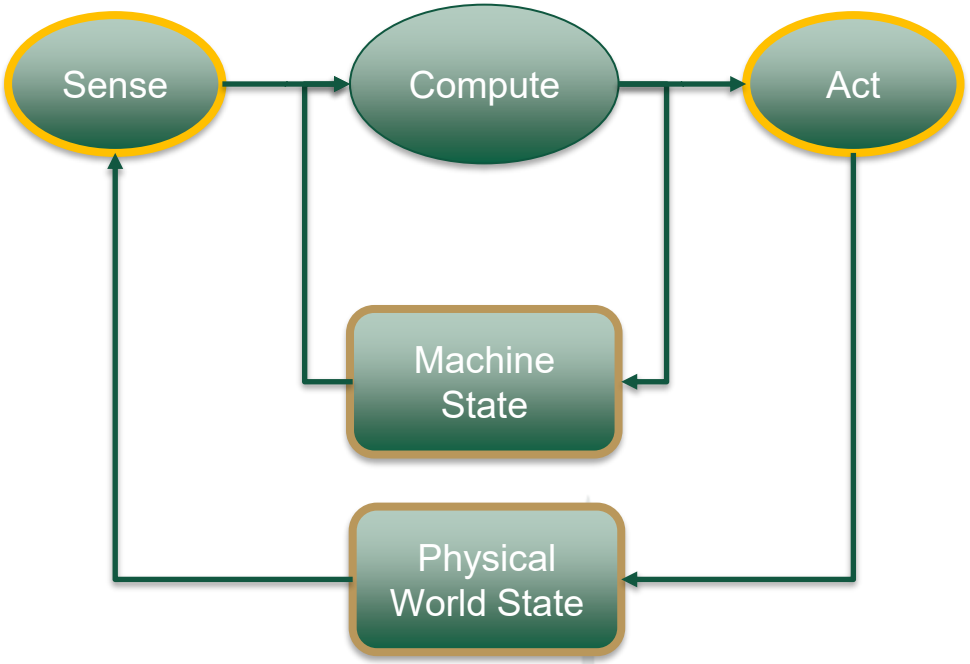
## CSCI 420-04 Robotics

WILLIAM & MARY

CHARTERED 1693

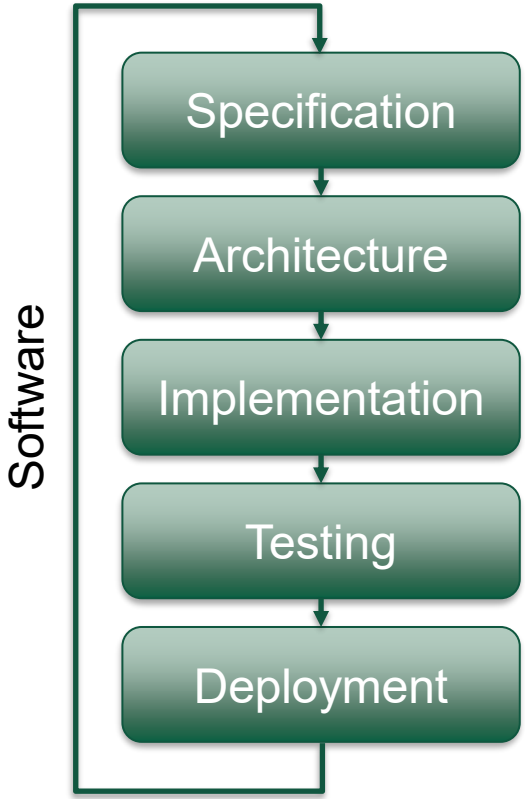

WIRED

https://www.aboutamazon.com/news/operations/amazon-robotics-robots-fulfillment-center



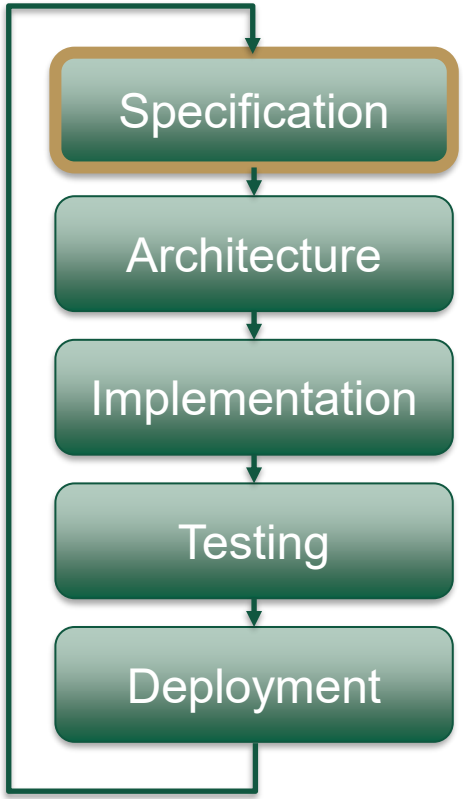

- What does it sense?
- What does it compute?
- How does it act?
- Machine state vs world state?

# Development Lifecycle



# Development Lifecycle

Software




Specification relative to the *world*

Robot

Physical:

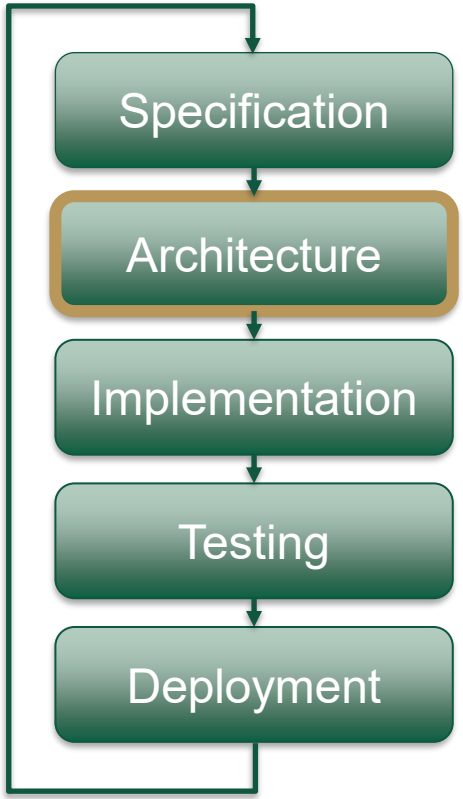
- 39.8”x28.3”x7.8”
- Lift 1250 lbs
- Top speed, battery life, etc.

World Behaviors:

- Operate over 1 million sq ft (~20 football fields)
- Read floor markers
- Navigate to any pod
- Avoid people, robots, unknown objects

# Development Lifecycle

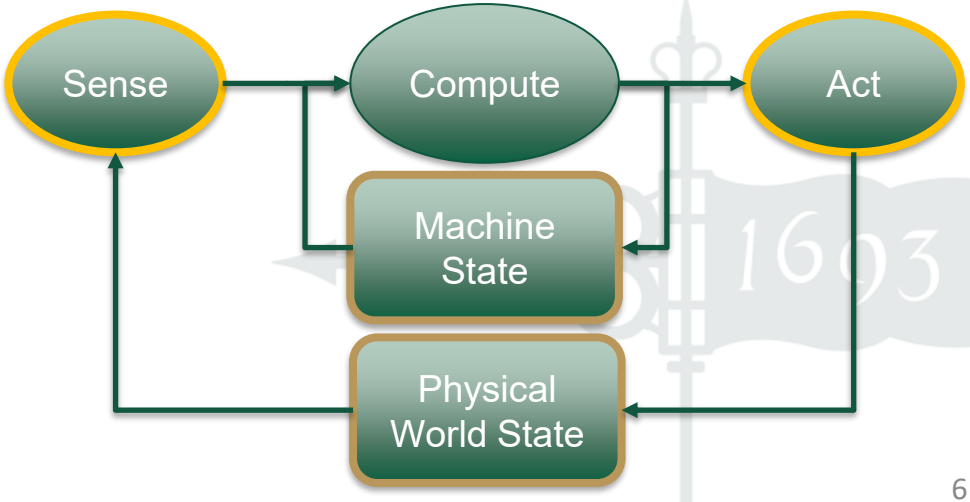
Software



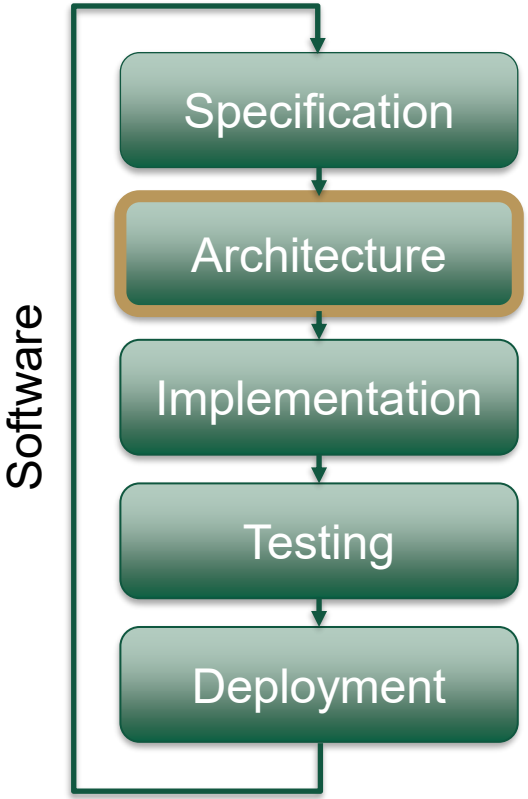

Specification relative to the *world*

Architecture

Robot



# Development Lifecycle



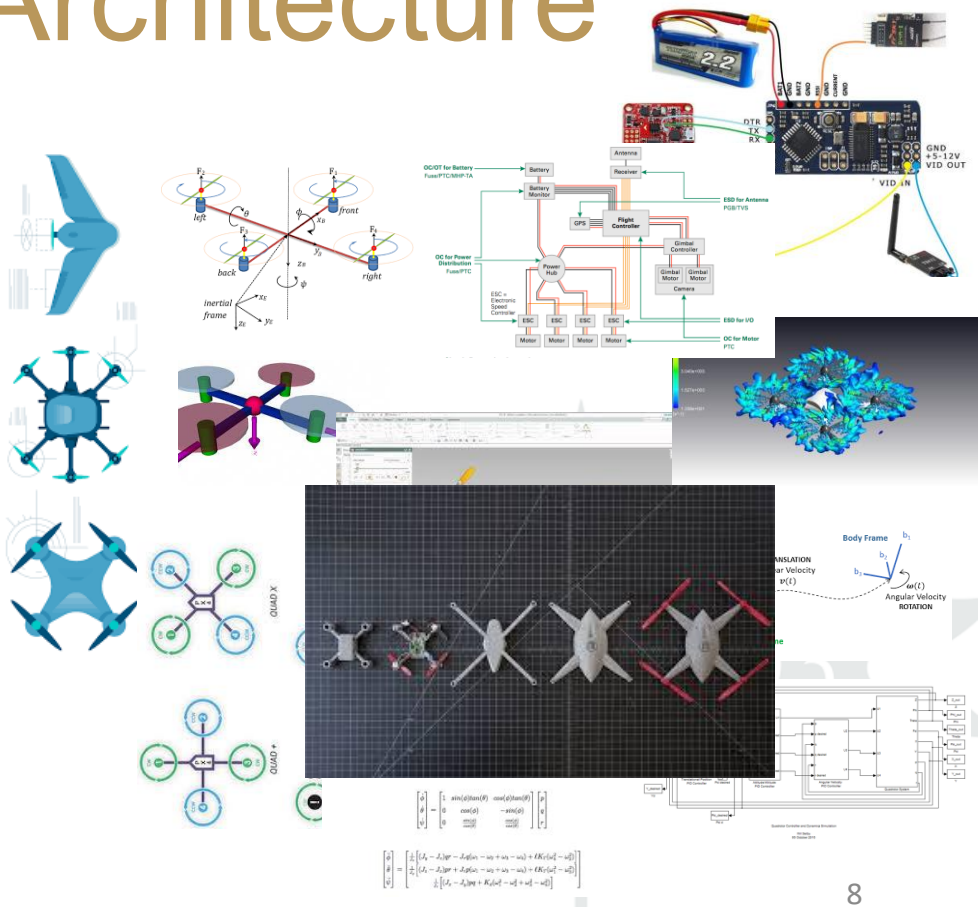

Specification relative to the *world*

Software Architecture

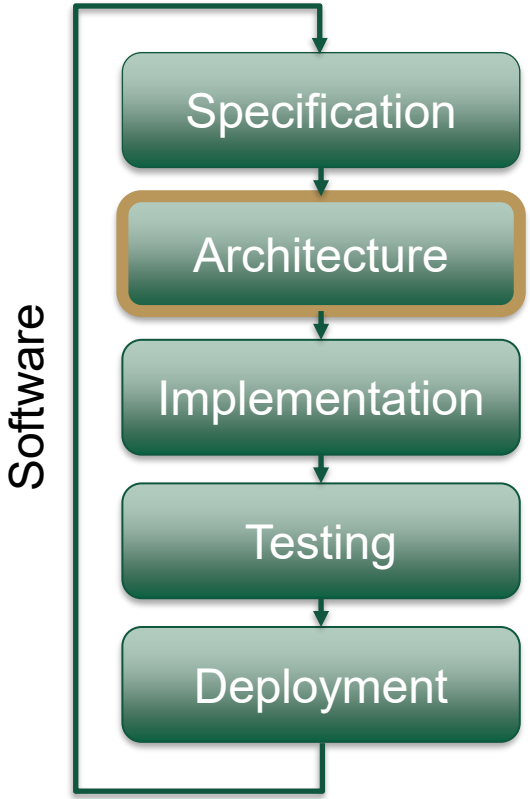
Hardware Architecture

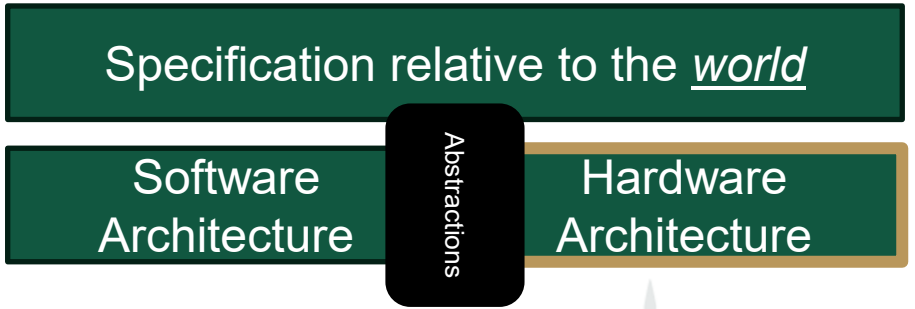
Robot

# Hardware Architecture



# Development Lifecycle



# Development Lifecycle

Software

Specification

Architecture

Implementation

Testing

Deployment

Specification relative to the *world*

Software Architecture

Abstractions

Hardware Architecture

Robot

Localization

Abstraction

Jpg 128x128
Rate > 10 hz
at: /bus/cameras/down/

Abstract

# Development Lifecycle

Software

Specification

Architecture

Implementation

Testing

Deployment

Specification relative to the *world*

Software Architecture

Abstractions

Hardware Architecture

Robot

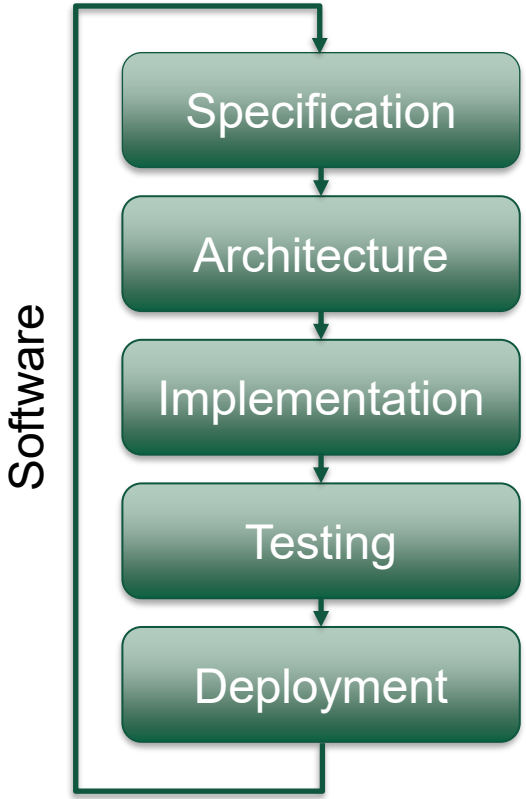
Planning

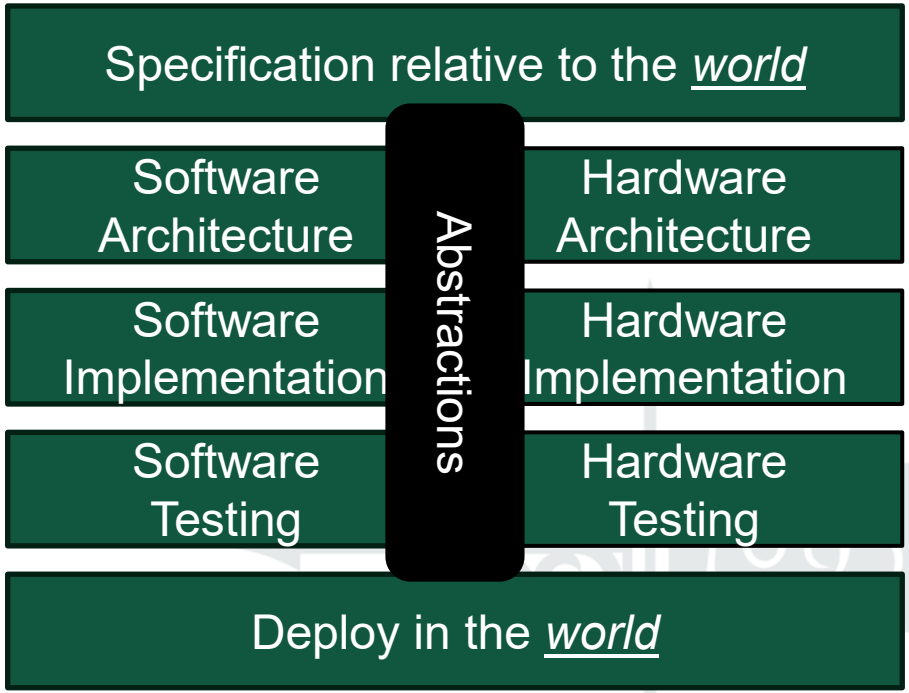
Abstract

Abstraction

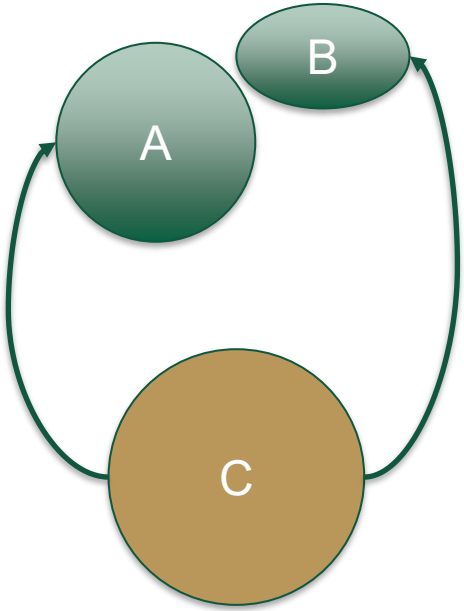
*Linear_Velocity [x,y]*
*Angular_Velocity [x]*

# Development Lifecycle

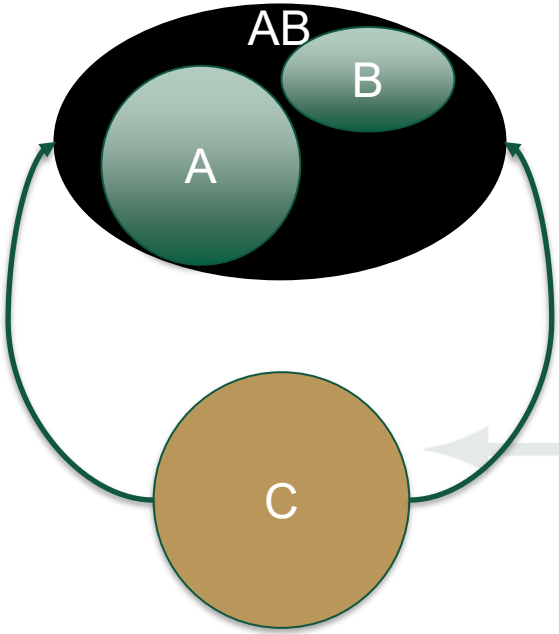


# Abstractions are Imperfect

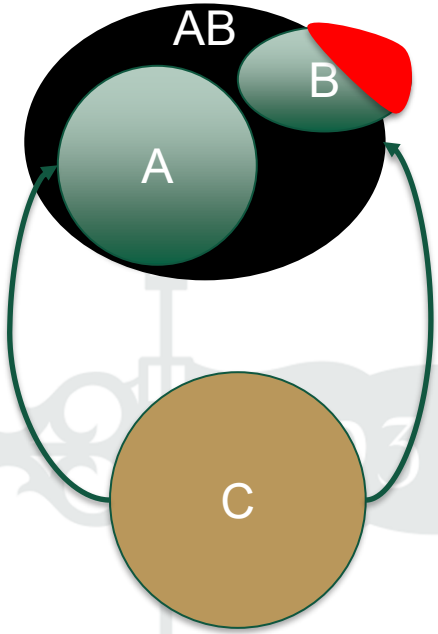



C must understand A & B




C uses abstract interface

Interface may not match implementation!



# Law of Leaky Abstractions

- *Iterating direction on a 2D array does not matter*
- *Accessing virtual memory has a constant speed*
- *SMB  are the same as local file*
- *SQL query with "where a=b and b=c and a=c" = "where a=b and b=c"*

***What leaks in these examples?***

# Why do we use abstractions?

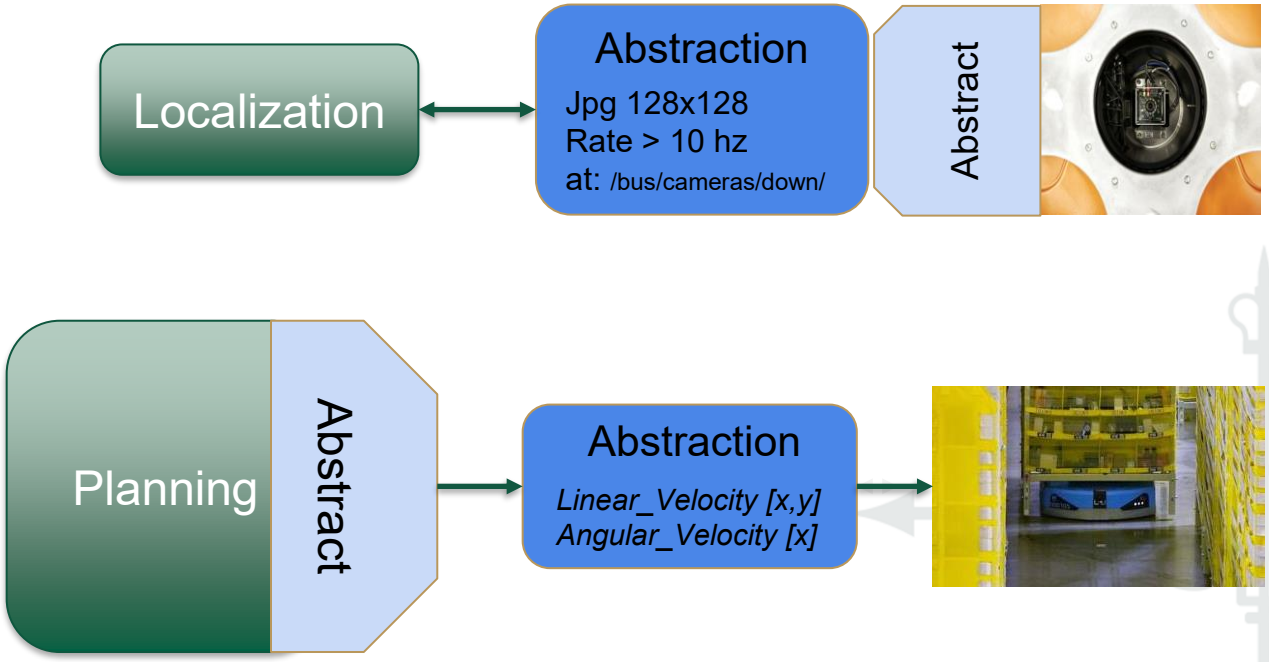
- Abstractions hide “the mess” to keep us organized and distraction free
  - This works until “the mess” is important!
- “**All non-trivial abstractions, to some degree, are leaky.” – Joel Spolsky**
  - An assumption doesn’t hold
  - An exception occurs

# Why do we use abstractions?

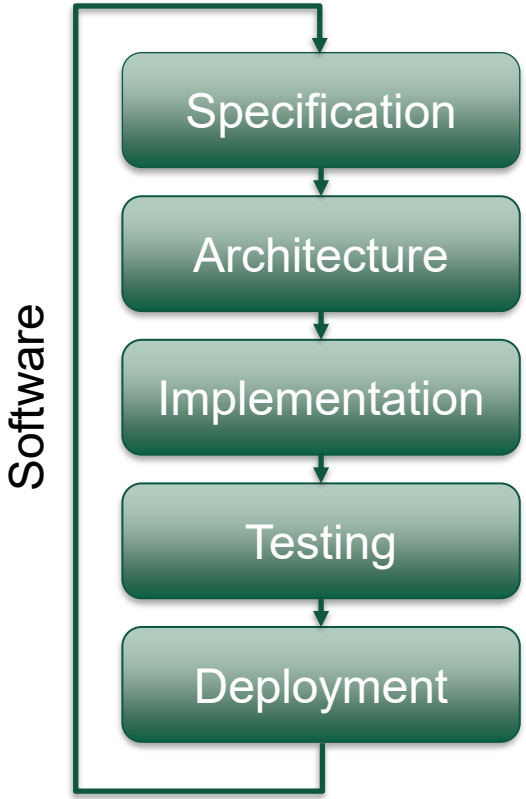
**Abstractions in robotics are particularly leaky!**

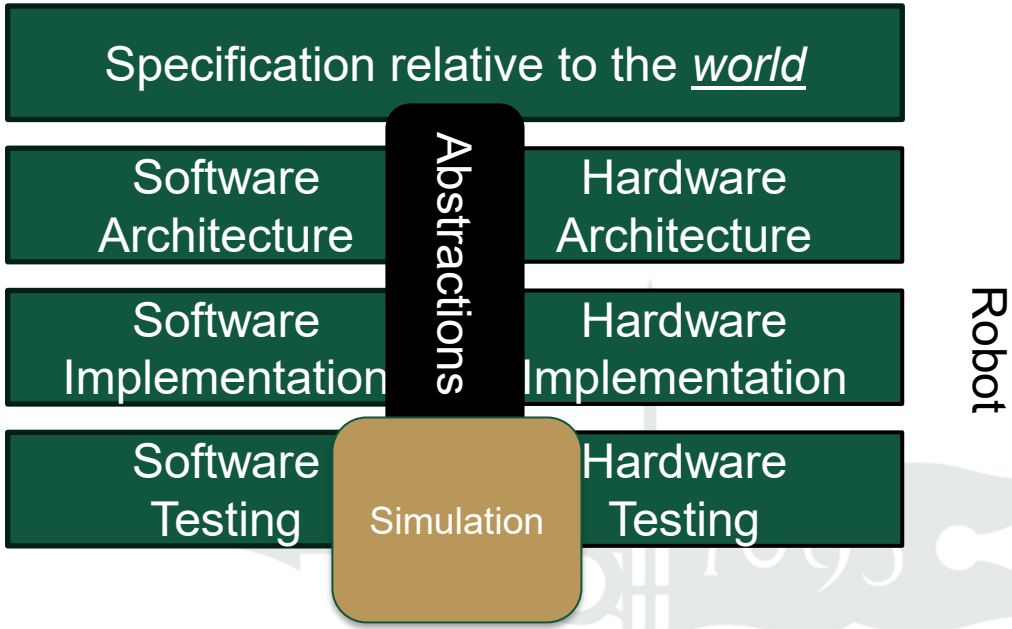
- Sensing
- States
- Actuation
- Communication

# Where do leaks occur?



# Development Lifecycle



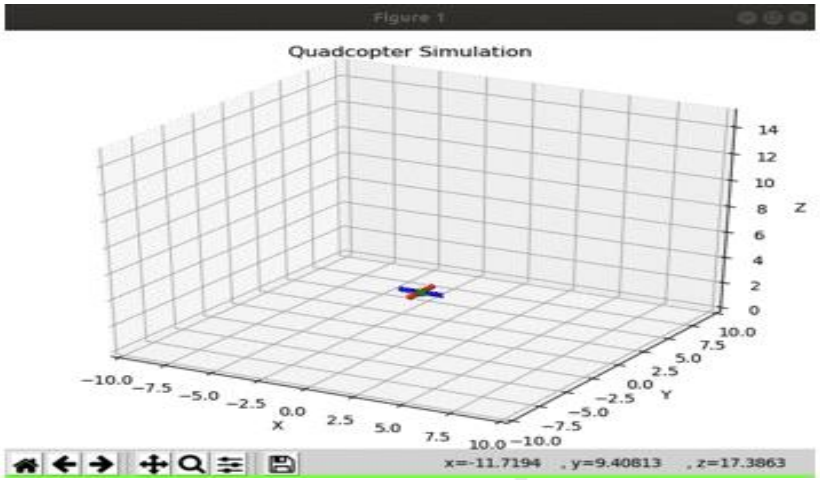



# When do we use simulation?

- In traditional software?
  - Code not available
  - Too Complex/Expensive
- In robotics:
  - The world: complex, noisy, *expensive failures*
  - Other components: sensors, actuators, code

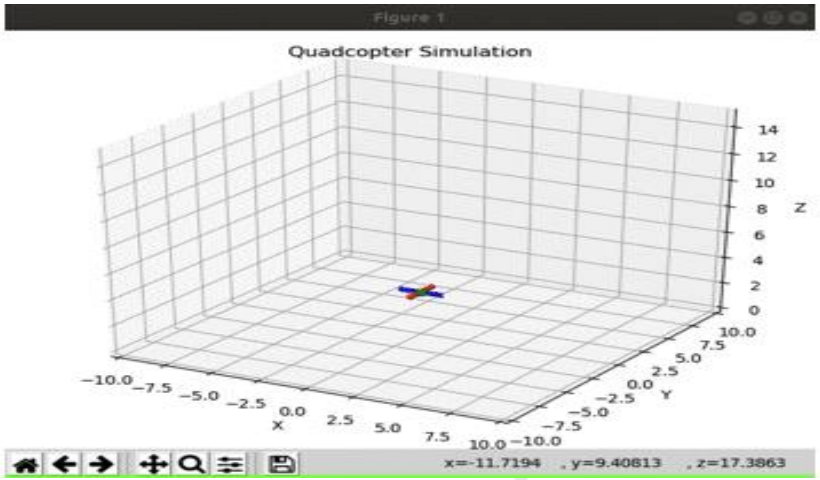
# What do you simulate?

- Develop code to make the drone hover

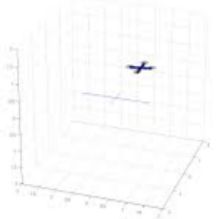


# What do you simulate?

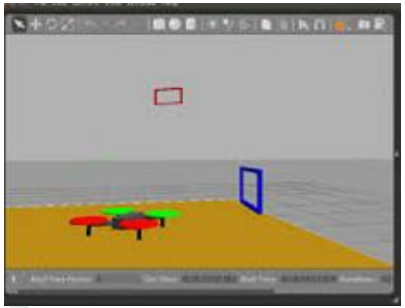
- Develop code to make the drone *collect*



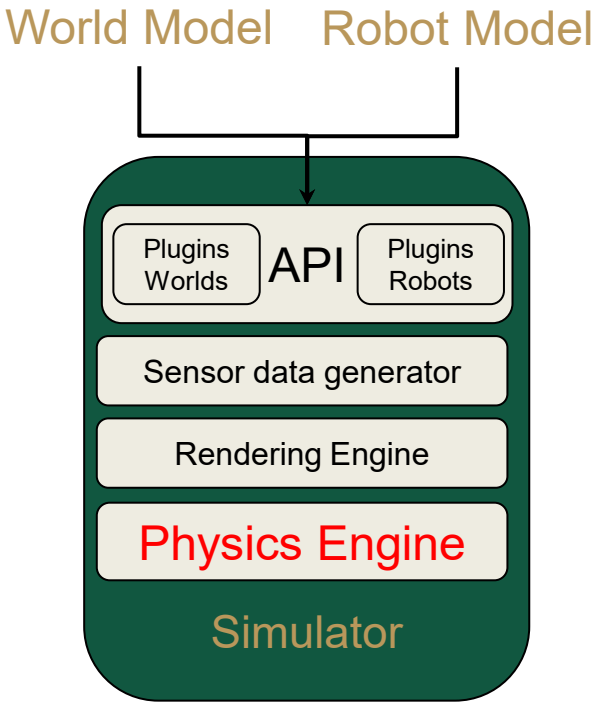
# Why Simulation?

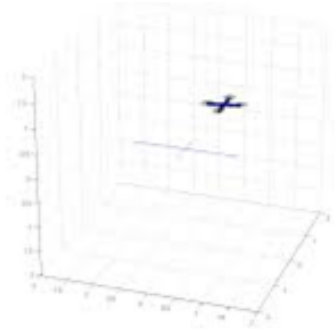


- Uses:
  - Explore designs
  - Test
  - For SW/HW/both
  - Training data
- Benefits:
  - Quick
  - Safe
  - Cheap

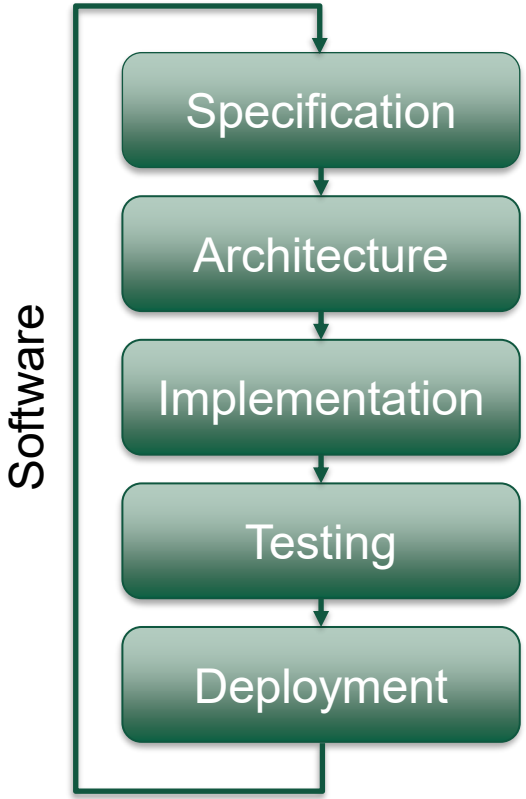


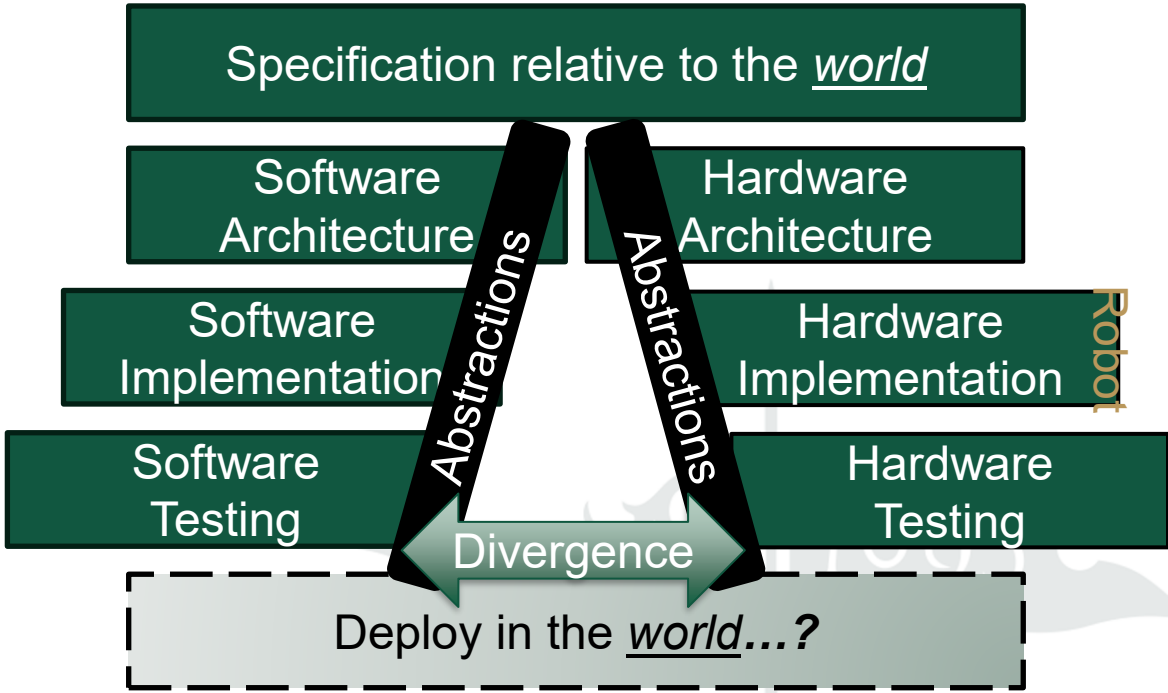
# Anatomy of a Simulator



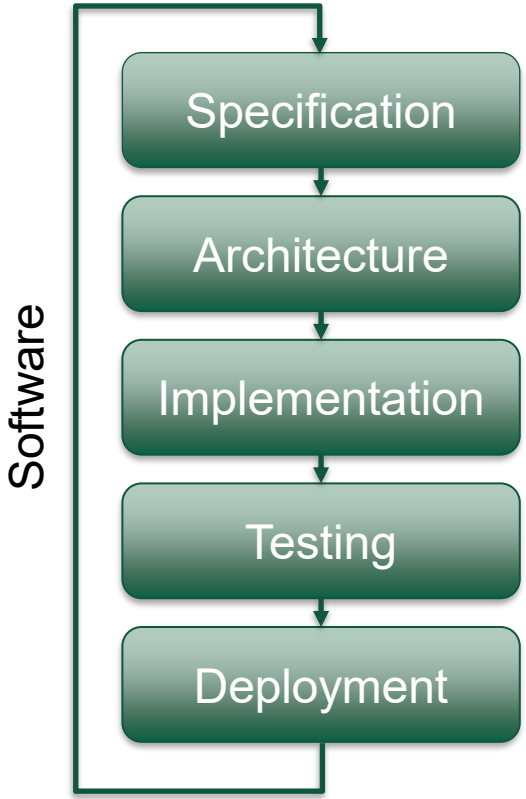



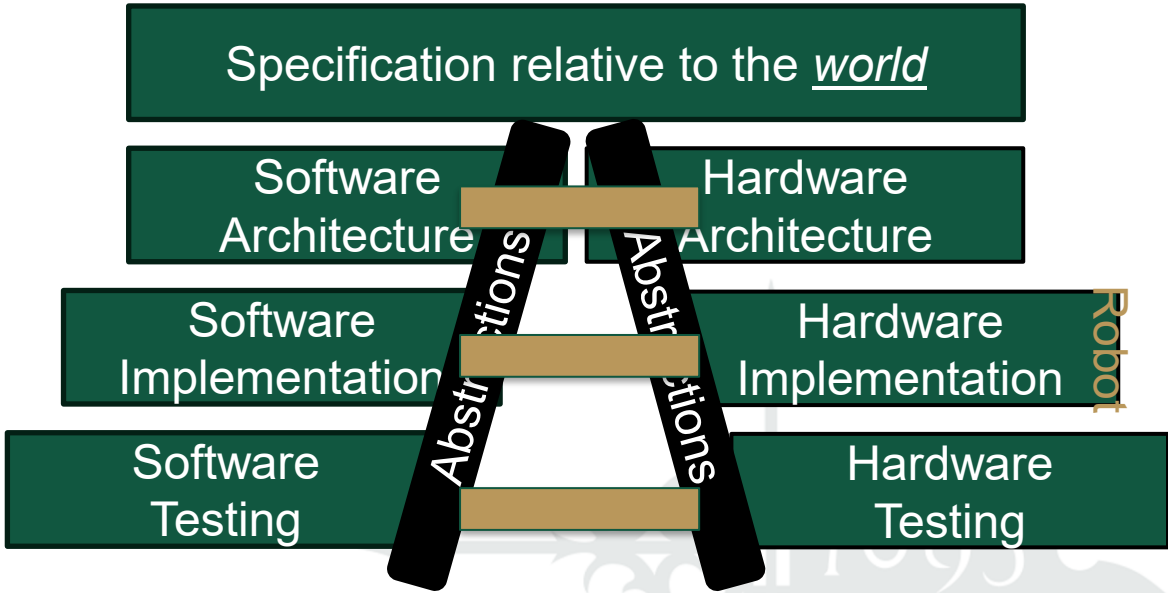
# Development Lifecycle



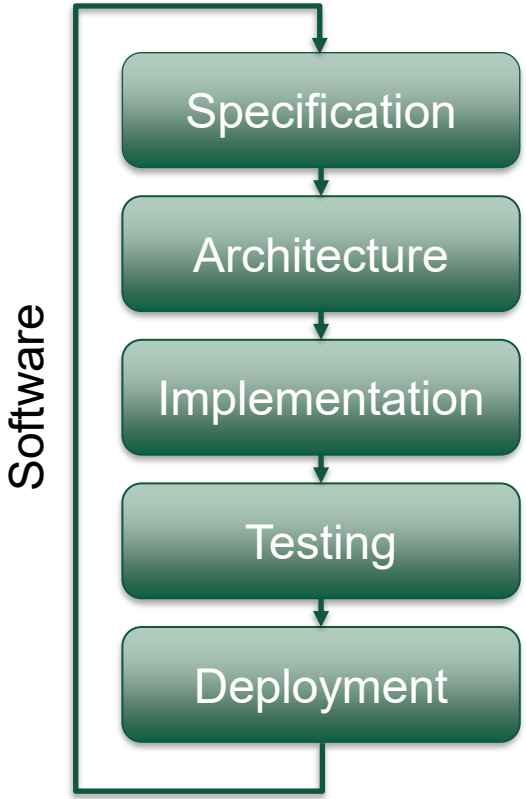
# Development Lifecycle

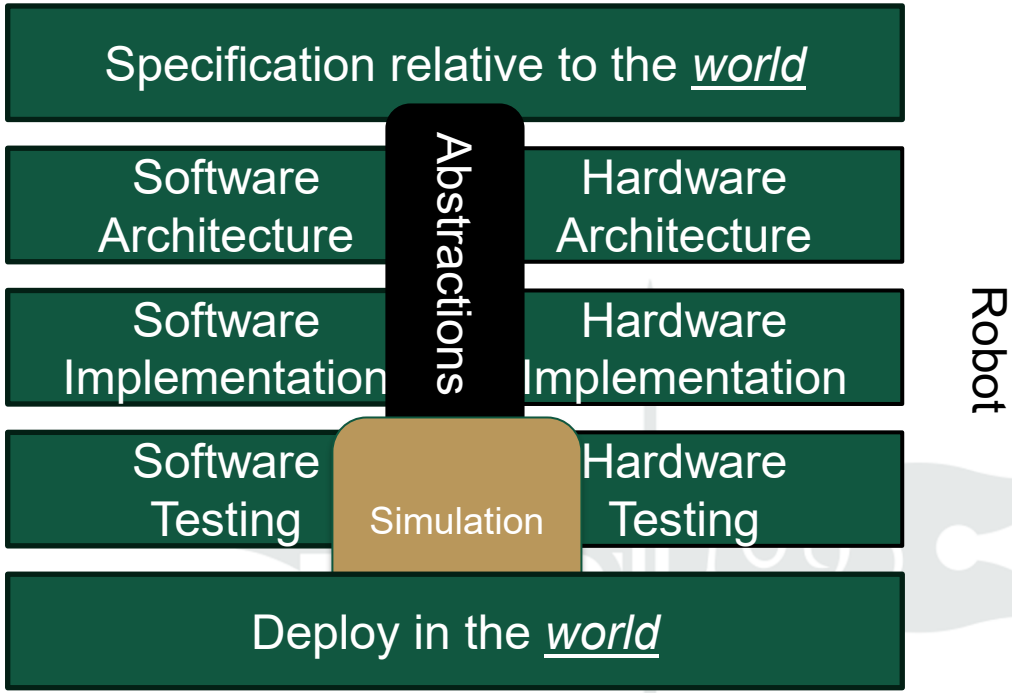


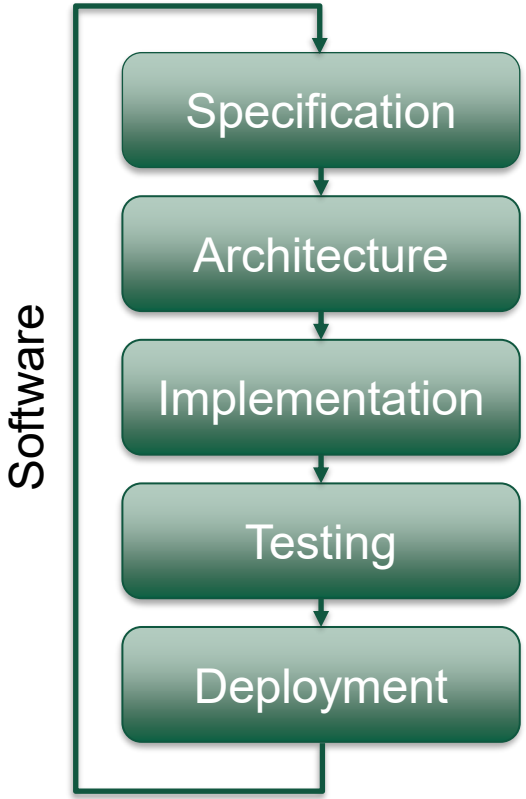

Must ensure alignment at each step!

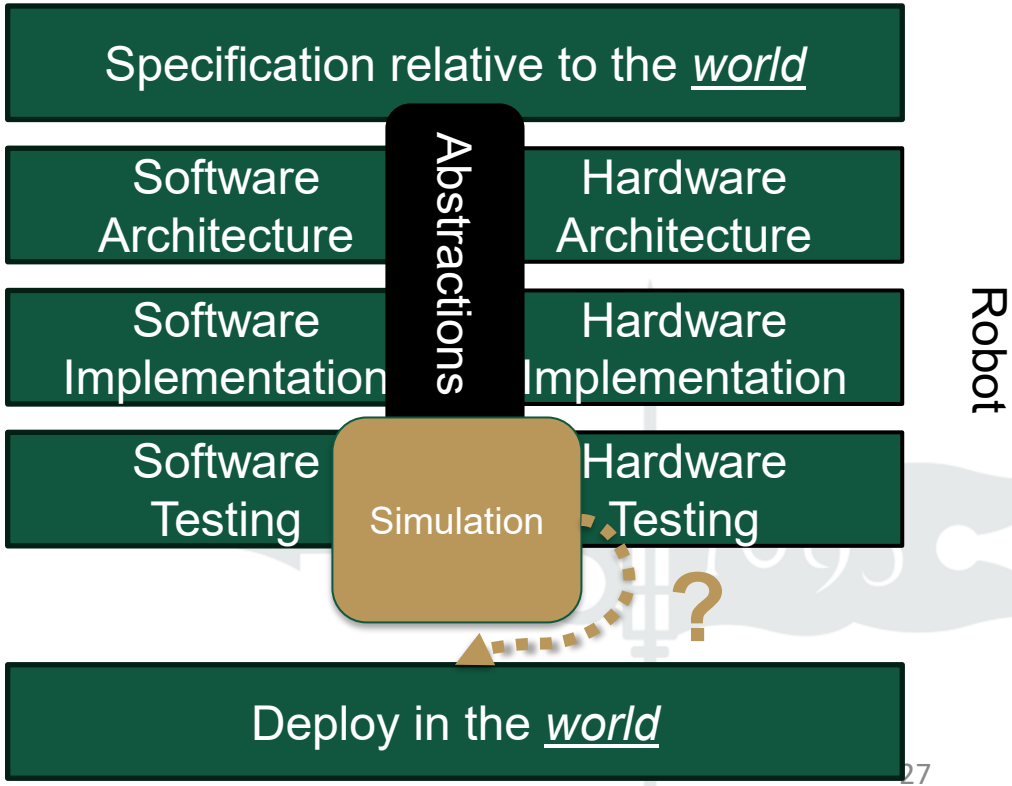
# Development Lifecycle



# Development Lifecycle



# Robot Deployment

- Simulation Reality Gap – Sim2Real
- In deployment:
  - Define initial states
  - Many distributed processes
  - Thousands of configuration parameters
  - Optimization based on scenario

# Robot Development

- Physical requirements (constraint & reqs)
- Leaky Abstractions (world & components)
- Split but coupled SW & HW dev
- Simulation is useful when used correctly
- Iterate design/abstraction/implementation

# Robot Development

- Highly multi-disciplinary
  - Rich vocabulary
  - Multiple points of integration
  - High opportunity for innovation
  - High opportunity for mismatch/breakdown

**Robot development is complex!**