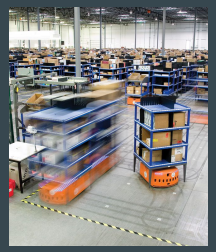
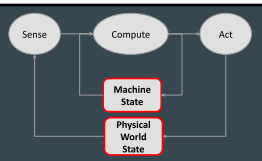
CS4501
Robotics for Soft Eng

•••

Distinguishing Features Of Robot System Development

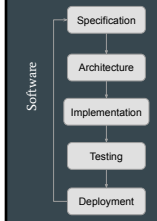


www.amazonrobotics.com

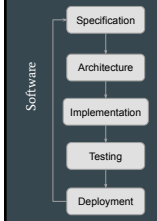


What does it sense?
What does it compute?
How does it act?
Machine state vs world state?

Development lifecycle



Development lifecycle



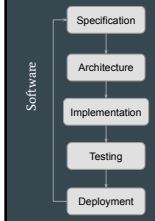
Specification relative to the World

Robotics

Physical attributes
- *Size: 40" x 40" X 6"*
- *Lift 500 pounds*
- *Speed up to 2 mph*

World Behaviors
- *Follow ground markers*
- *No crashes against stationary objects*
- *…*

Development lifecycle



Robotics

Specification relative to the World

Architecture

Development lifecycle

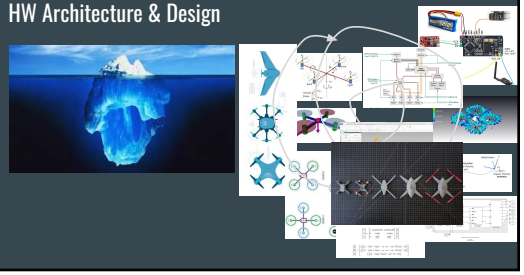
Software

- Specification
- Architecture
- Implementation
- Testing
- Deployment

Robotics

- Specification relative to the World
- Architecture SW
- Architecture HW

HW Architecture & Design



Development lifecycle

Software

- Specification
- Architecture
- Implementation
- Testing
- Deployment

Robotics

- Specification relative to the World
- Architecture SW
- Abstractions
- Architecture HW

Development lifecycle

Software

- Specification
- Architecture
- Implementation
- Testing
- Deployment

Robotics

- Specification relative to the World
- Architecture SW
- Abstractions
- Architecture HW
- Localization
- Abstraction
  - Jpg 128x128
  - Rate > 10 hz
  - at: /bus/camera/down/
- Abstract

Development lifecycle

Software

- Specification
- Architecture
- Implementation
- Testing
- Deployment

Robotics

- Specification relative to the World
- Architecture SW
- Abstractions
- Architecture HW
- Planning
- Abstract
- Abstraction
  - *Linear_Velocity [x,y]*
  - *Angular_Velocity [x]*

Development lifecycle

Software

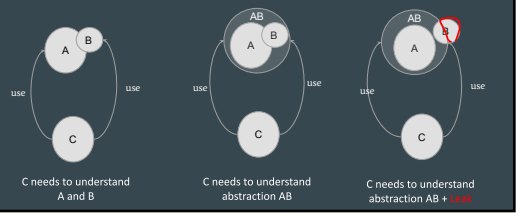
- Specification
- Architecture
- Implementation
- Testing
- Deployment

Robotics

- Specification relative to the World
- Architecture SW
- Implementation SW
- Testing SW
- Abstractions
- Architecture HW
- Implementation HW
- Testing HW

Law of Leaky Abstractions - G. Kiczales

Noticeable between Cyber to Physical



Law of Leaky Abstractions - J. Spolsky examples

- *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"*
- *VMs emulate an OS like it's running on real hardware*

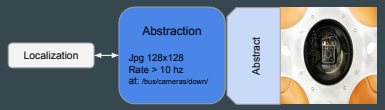
What is leaking?

Law of Leaky Abstractions

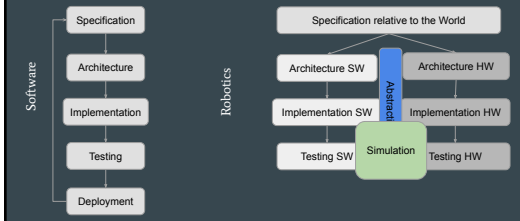
- Abstractions makes us more efficient, until they leak
- All good abstractions leak
  - They have exceptional behaviors
  - They break underlying assumptions

Law of Leaky Abstractions

Where could it leak?



Development lifecycle



Simulation in Robotics

Developing Software

- Mock when
  - Relying on other components
    - Not available yet
    - Too complex/expensive
    - Functions to Databases

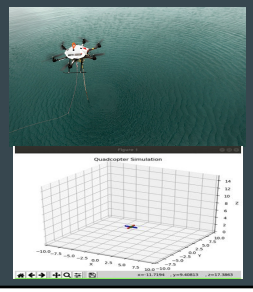
Developing Robots

- Mock when
  - Relying on world
    - Too complex
    - Failures too expensive
  - Relying or other components
    - Sensors
    - Actuators
    - ...
    - Software

Simulation in Robotics

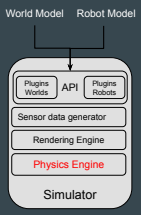
Test hovering functionality

What do you mock?

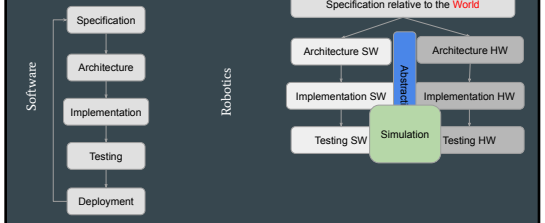


Simulation in Robotics

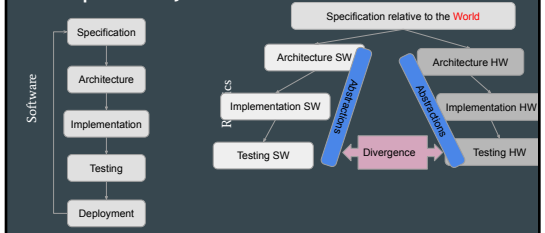
- Applications
  - Design Exploration
  - Testing
  - For SW/HW/Both
  - Training data
- Benefits
  - Accelerates development
  - Low cost



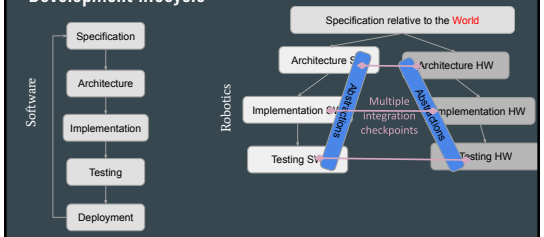
Development lifecycle



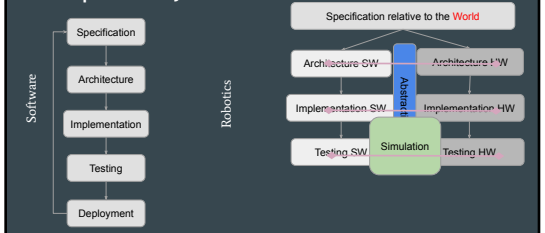
Development lifecycle



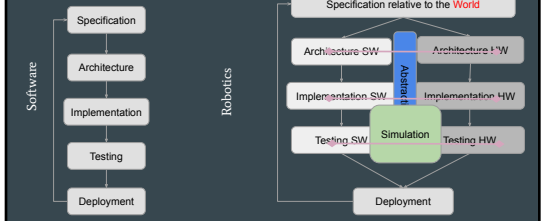
Development lifecycle



Development lifecycle



Development lifecycle



Robot deployment

- Define acceptable initial states
- Multiple distributed processes
- Thousands of configuration parameters
- Optimization for scenarios

Programming the deployment

Robot development lifecycle

- Physical Requirements
- Multi-level Abstractions
- Parallelize synchronized SW/HW development
- Simulation is key tool
- Decomposition is interleaved with discovery
- Highly-multidisciplinary
  - Richer vocabulary
  - Higher opportunity for innovation
  - Higher opportunity for breakdowns

More Complex Development Process

Differences on the SW side

SW Specification Differences - Trickling of Physical World

- State Properties
  - Rate of descent < 3m/s
  - Angle < 17 degrees
  - Calibrated = True

Properties may include physical terms

SW Specification Differences - Trickling of Physical World

- State Properties
  - Rate of descent < 3m/s
  - Angle < 17 degrees
  - Calibrated = True
- Conditional State Properties
  - If approaching, then speed < delta
  - If taking off, proximity sensor should be false

Properties may include physical terms

Properties are state-dependent

SW Specification Differences - Trickling of Physical World

- State Properties
  - Rate of descent < 3m/s
  - Angle < 17 degrees
  - Calibrated = True
- Conditional State Properties
  - If approaching, then speed < delta
  - If taking off, proximity sensor should be false
- Timeliness properties
  - Frequency Heartbeat = 20hz
  - Abort sequence takes less than 2s
- Temporal properties
  - Battery > 30% before Takeoff
  - Translation can only occur after takeoff

Properties may include physical terms

Properties are state-dependent

Timeliness matters

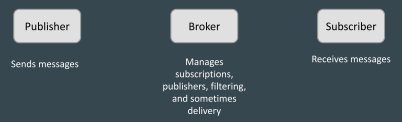
SW Architectural and Design Differences

- Asynchronous
- Loosely coupled
- Abstracted
- Close-loop

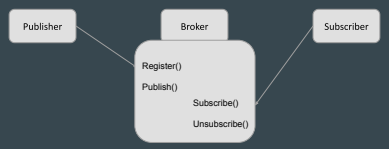
SW Architectural and Design Differences

- Asynchronous, event-driven -- world operates that way
- Loosely coupled -- parallelization, reuse
- Abstraction -- manage complexity
- Close loop -- need to assess/respond to changes

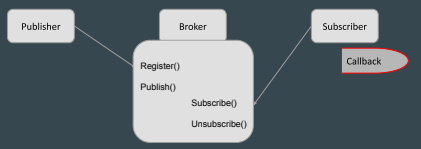
SW Differences: Publish/Subscribe



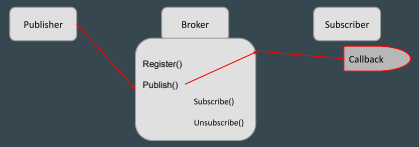
SW Differences: Publish/Subscribe



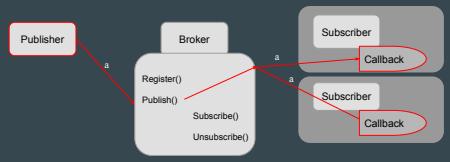
SW Differences: Publish/Subscribe



SW Differences: Publish/Subscribe

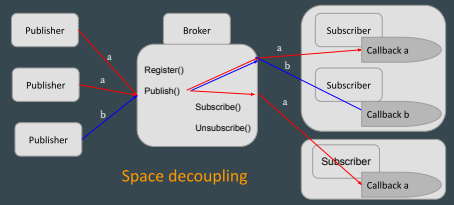


SW Differences: Publish/Subscribe



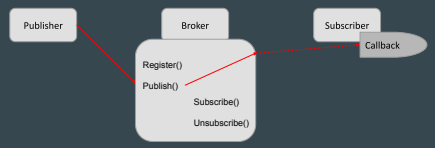
Space decoupling

SW Differences: Publish/Subscribe



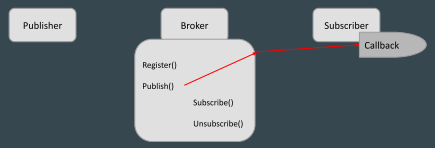
Space decoupling

SW Differences: Publish/Subscribe



Time decoupling

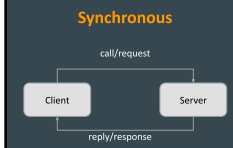
SW Differences: Publish/Subscribe

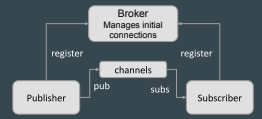


Time decoupling

Publish/Subscribe vs. Client-Server

Synchronous



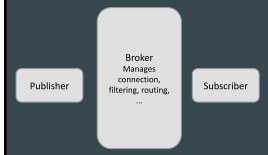


Asynchronous

SW Differences: Publish/Subscribe Functionality

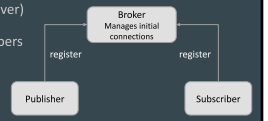
- Filtering
  - Which subscribers get what messages
    - Topic-based
    - Content-based
- Routing
  - Getting those messages to subscribers
  - Alternatives: Unicast / Multicast / Push-pull

SW Differences: Publish/Subscribe ROS



SW Differences: Publish/Subscribe ROS

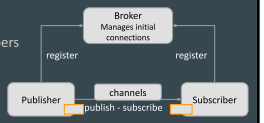
- Broker is a Manager (core+param server)
- Nodes can be Publishers and Subscribers



ROS also offers Client-Server, and Actions

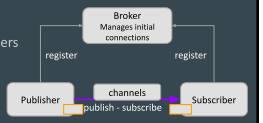
SW Differences: Publish/Subscribe ROS

- Broker is a Manager
- Nodes can be Publishers and Subscribers
- Topic-based filtering with buffering



SW Differences: Publish/Subscribe ROS

- Broker is a Manager
- Nodes can be Publishers and Subscribers
- Topic-based filtering with buffering
- Peer-to-peer routing



SW Differences: Publish/Subscribe ROS

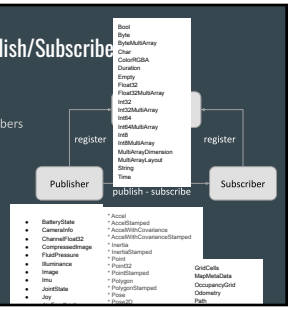
- Broker is a Manager
- Nodes can be Publishers and Subscribers
- Topic-based filtering with buffering
- Peer-to-peer routing
- Standardized common msg formats

SW Design Differences: Publish/Subscribe

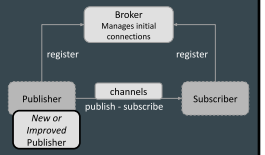
- Broker is a Manager
- Nodes can be Publishers and Subscribers
- Topic-based filtering with buffering
- Peer-to-peer routing
- Standardized common msg formats

SW Design Differences: Publish/Subscribe

- Broker is a Manager
- Nodes can be Publishers and Subscribers
- Topic-based filtering with buffering
- Peer-to-peer routing
- Standardized common msg formats
  - Standard types
  - State
  - Sensors
  - Actuators
  - Navigation

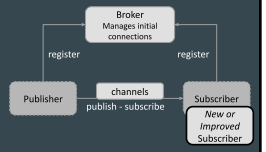


SW Design Differences: Publish/Subscribe ROS



Plug-Play REUSE

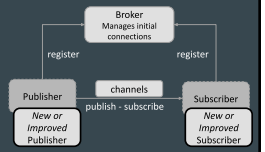
SW Design Differences: Publish/Subscribe ROS



Plug-Play REUSE

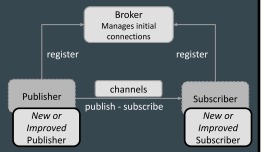
Law of Leaky Abstractions - ROS Pub/Sub

- Not everyone is bound to use standard messages or use them in the same way
- Channels do not offer real-time guarantees
- When underlying core hangs the results are unknown
- Size of the buffer matters
  - Implication of being too short?
  - Implication of being too long?



Law of Leaky Abstractions - ROS Pub/Sub

How can it leak?



Takeaways

- More complex development process, branch / sync / integrate
- Richer specifications that must include the physical world
- Many abstractions, many of them Leaky
- Simulation is a big part of modeling and testing
- Programming the deployment
- Asynchronous, event-driven, loosely coupled architectures
- Publish/Subscribe architecture, P/S ROS

Next - ROS Lab P/S and Simulation