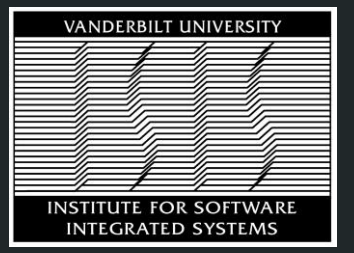


ROSMOD: A Toolsuite for Modeling, Generating, Deploying, and Managing Distributed Real-time Component-based Software using ROS

Pranav Srinivas Kumar, William Emfinger, and Gabor Karsai
Institute for Software-Integrated Systems
Vanderbilt University

This work was supported by DARPA under contract NNA11AB14C and USAF/AFRL under Cooperative Agreement FA8750-13-2-0050, and by the National Science Foundation (CNS-1035655). The activities of the 2014-15 Vanderbilt Aerospace Club were sponsored by the Department of Mechanical Engineering and the Boeing company. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA, USAF/AFRL, NSF, or the Boeing Company.

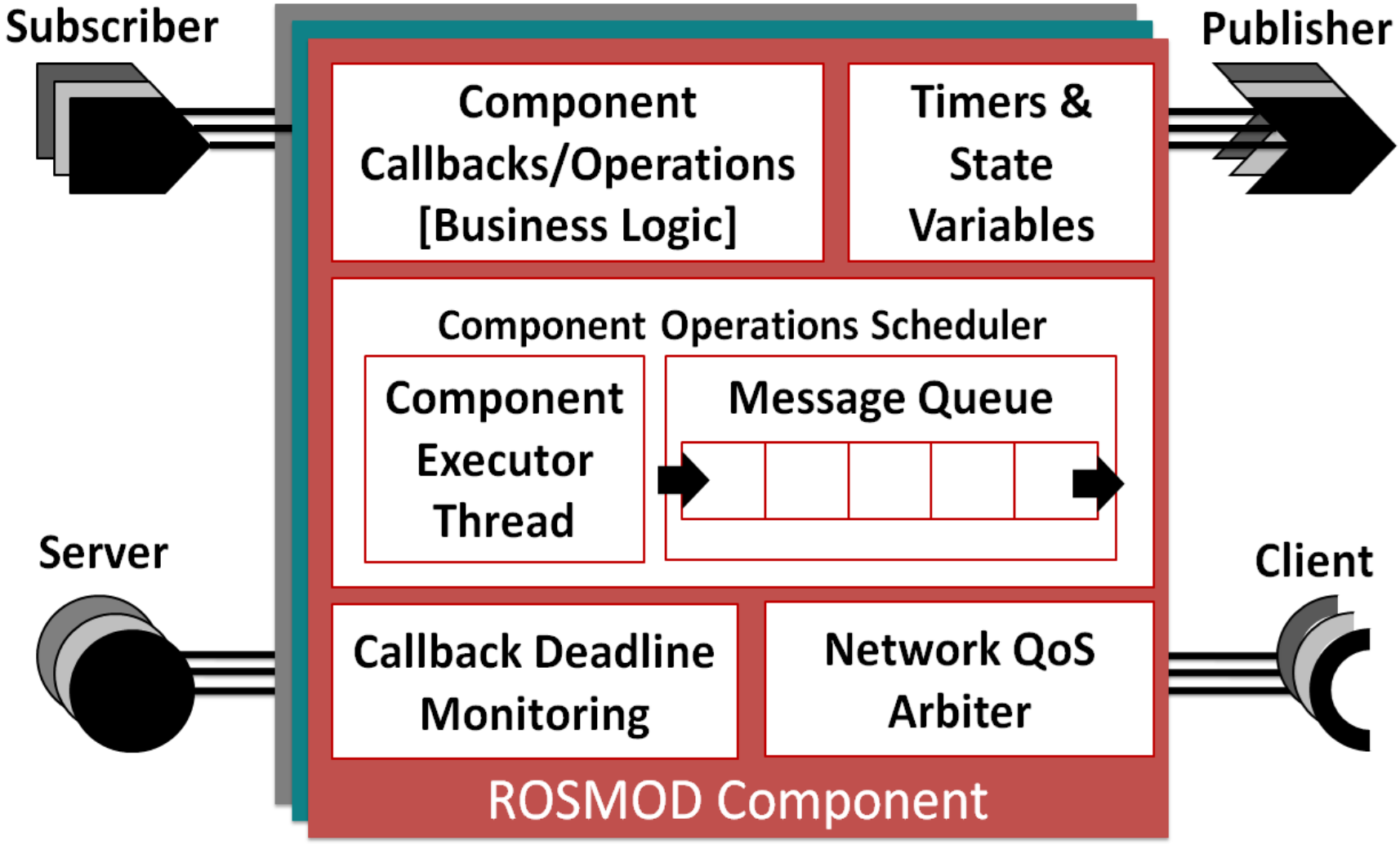
Introduction

- **ROSMOD** is a Model-driven Development (MDD) Toolsuite
- Designed for Rapid Prototyping Component-based Distributed Real-time Embedded (DRE) Applications with the Robot Operating System (ROS)
- Well-suited for large-scale cyber-physical applications on distributed embedded devices
- Includes a Graphical User Interface & Rendering Platform to enable model-based development
- Supports ROS Workspace code generation with code preservation
- Supports parallel deployment and monitoring of ROS processes
- Real-World Application: An Autonomous Ground Support Equipment (AGSE) robot for the NASA Student Launch competition, 2014-2015.

Robot Operating System (ROS)

- Meta-operating system framework for Robotic System Development
- Open-Source Multi-Platform Support
- Industrial Robotics, UAV Swarms, Low-power Image Processing Devices, etc.
- Requirement in several DARPA Robotics Projects (DRC)
- Enables Development of Network of Interacting ROS *nodes*
- Various Interaction Patterns: Client-Server, Publish-Subscribe, and Time-triggered Operations
- ROS Applications are packaged set of ROS nodes
- ROS Master: Singleton Discovery and Communications Broker

ROSMOD Component



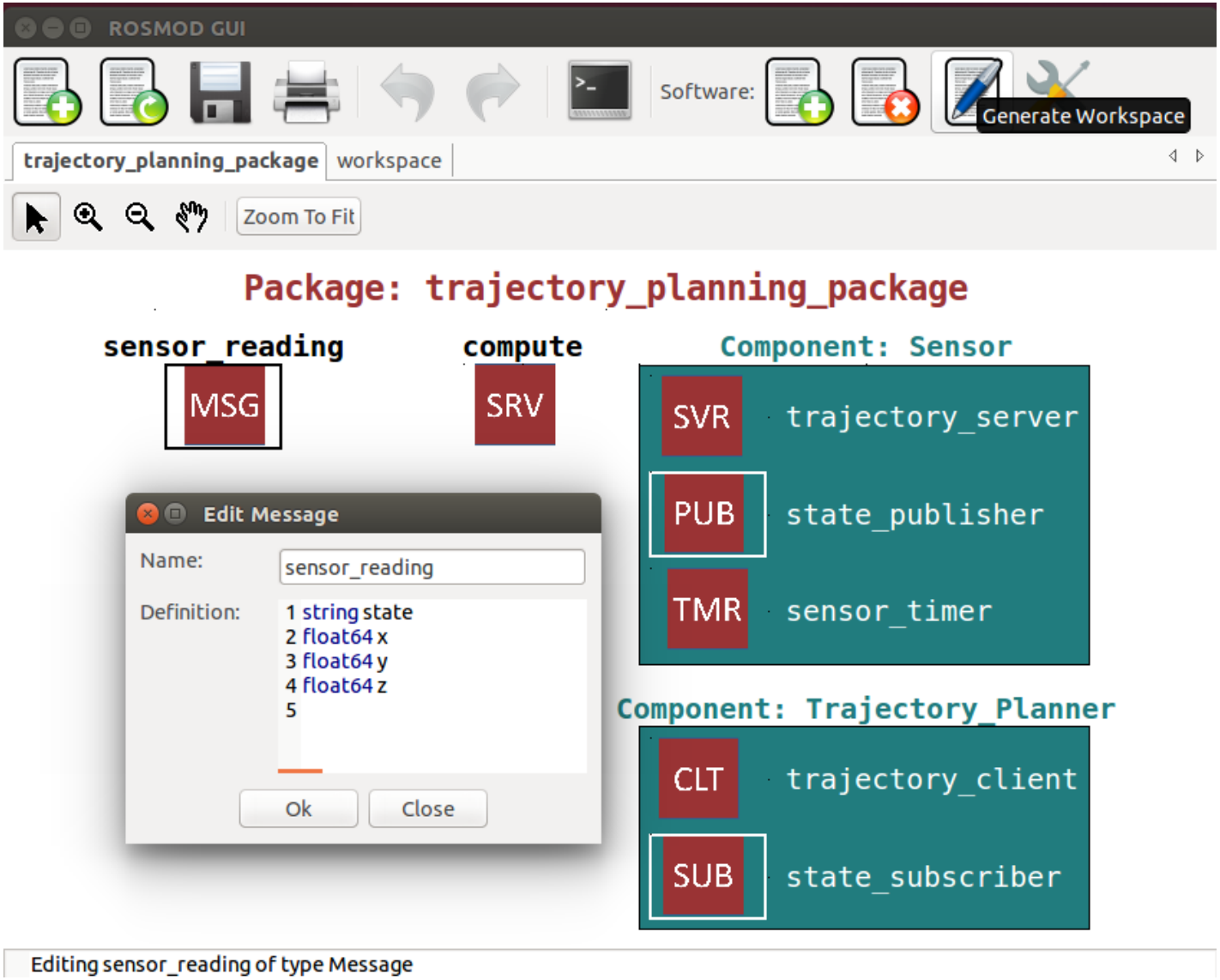
Component Model

- Each Component has a Message Queue
- Each Component has a *single* executor thread
- Each Component exposes *operations* through port interfaces.
- Message Queue receives operation requests from other components
- Component Operation Scheduler schedules one request at a time from the queue
- Requests are processed based on a scheduling scheme e.g. FIFO, PFIFO, or EDF scheduling
- Operation execution is single-threaded (i.e. nonpreemptive) per component
- Single threaded operation execution helps avoid synchronization primitives and locking mechanisms in application code

ROSMOD Projects: Models

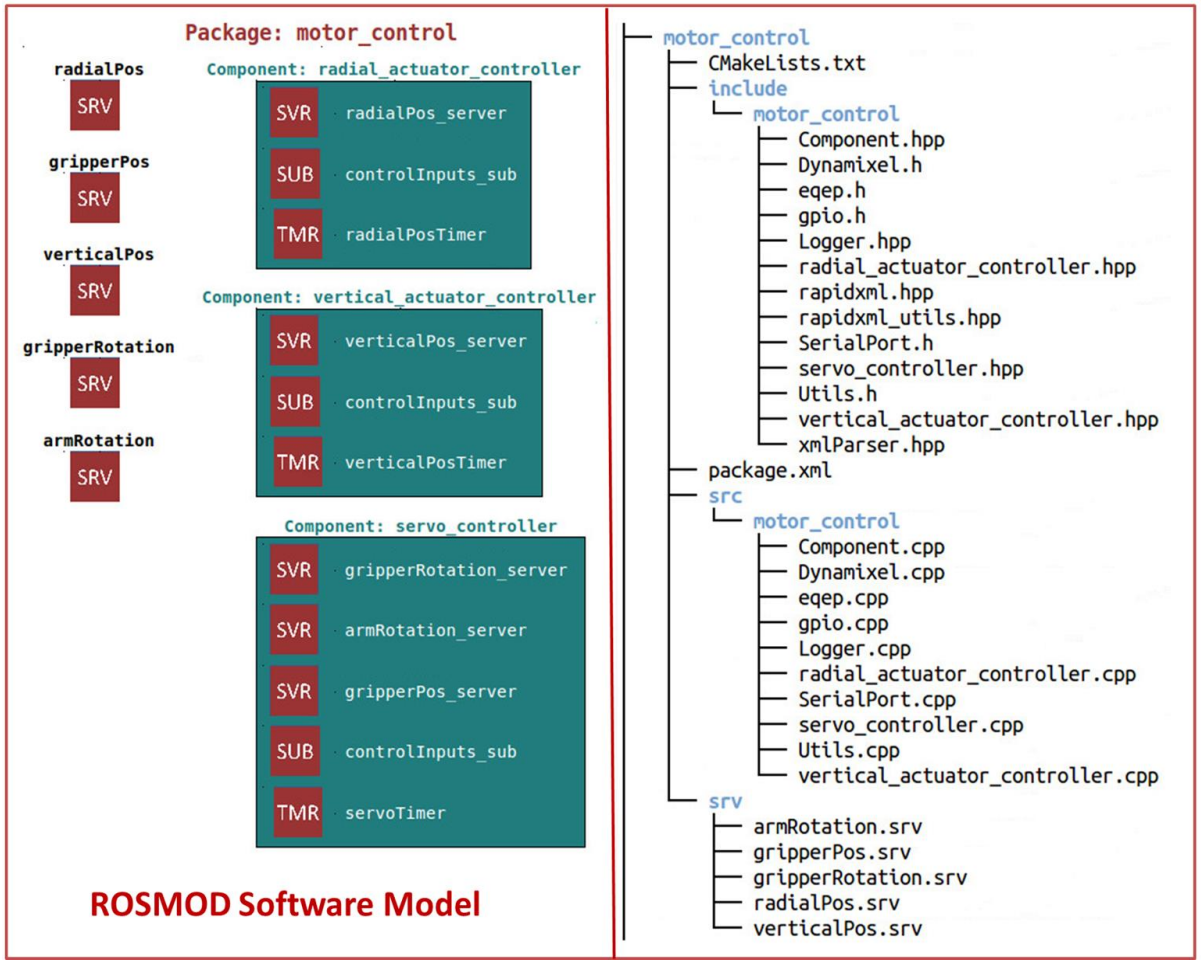
- Software Model
  - Represents a ROS Workspace
  - Defines Messages, Services and Components
- Hardware Model
  - Defines Hardware Devices - IP Address, SSH Keys, Architecture etc.
- Deployment Model
  - Define ROS Nodes (Processes)
  - Instantiate Components (from Software Model) in ROS Nodes
  - Define a ROS Node to Hardware Mapping

Graphical User Interface



Workspace Generation

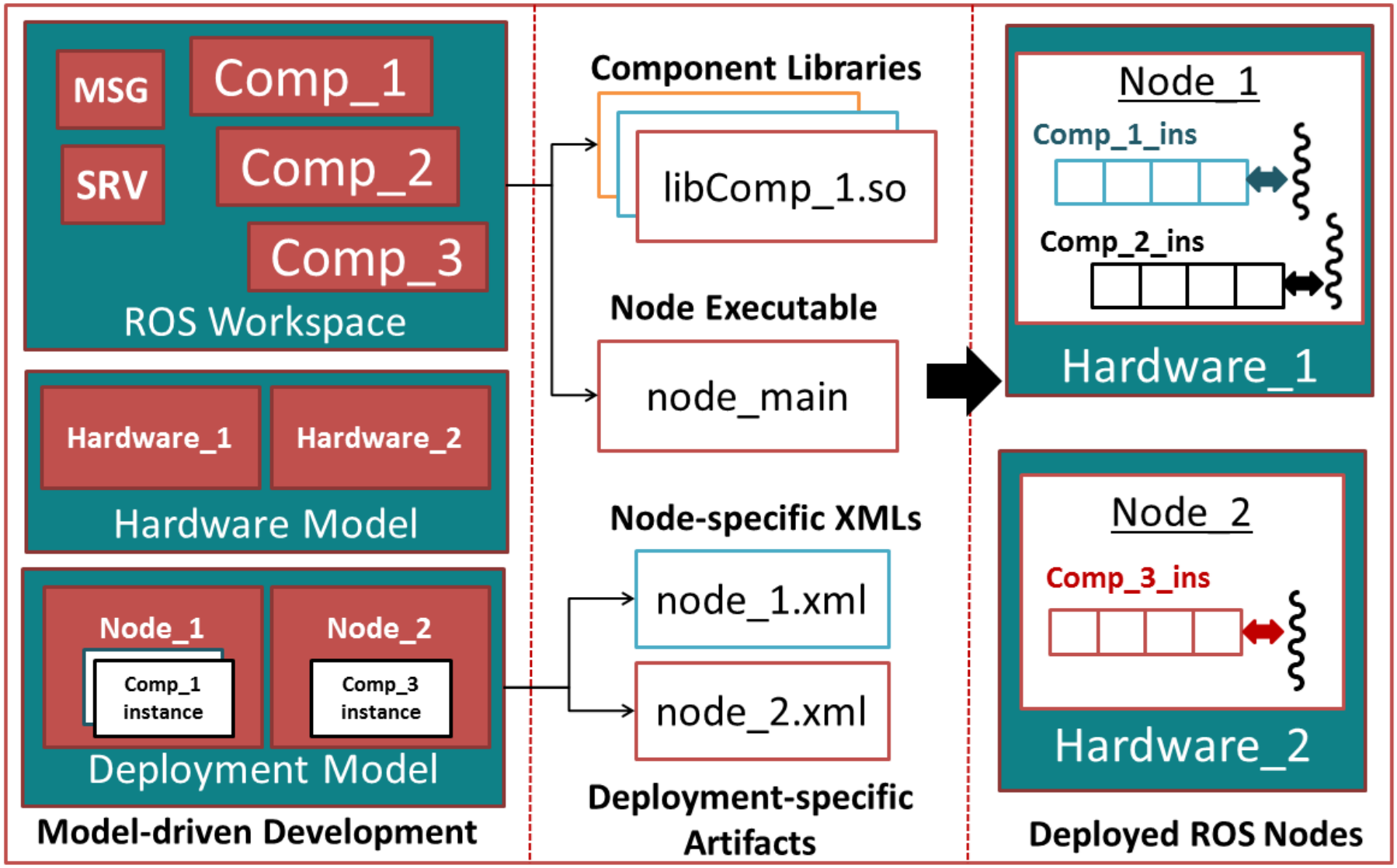
- C++ Classes for each ROSMOD Component
- Package-specific msg and srv files
- Logging and XML parsing framework
- Build system files
- Code preservation markers and Doxygen comments
- Follows ROS Package guidelines



ROSMOD Application Development

- Prepare a ROSMOD Model of the application: interfaces, architecture, deployment
- Generate the ROS Workspace using ROSMOD
  - Generated code includes Code-Preservation Markers
- Add *business logic code* to generated skeleton callbacks/operations
- Add new components, ports, messages etc. to the model
- Re-generate ROS Workspace
  - Previously added business logic code is preserved
  - Newly added modeling elements manifest as new code segments
- No need to complete the ROSMOD model to begin implementing the application code
- On-the-fly feature additions and rapid prototyping

Software Deployment Infrastructure



Autonomous **G**round **S**upport **E**quipment (AGSE) Robot

NASA Student Launch Competition, 2014-2015

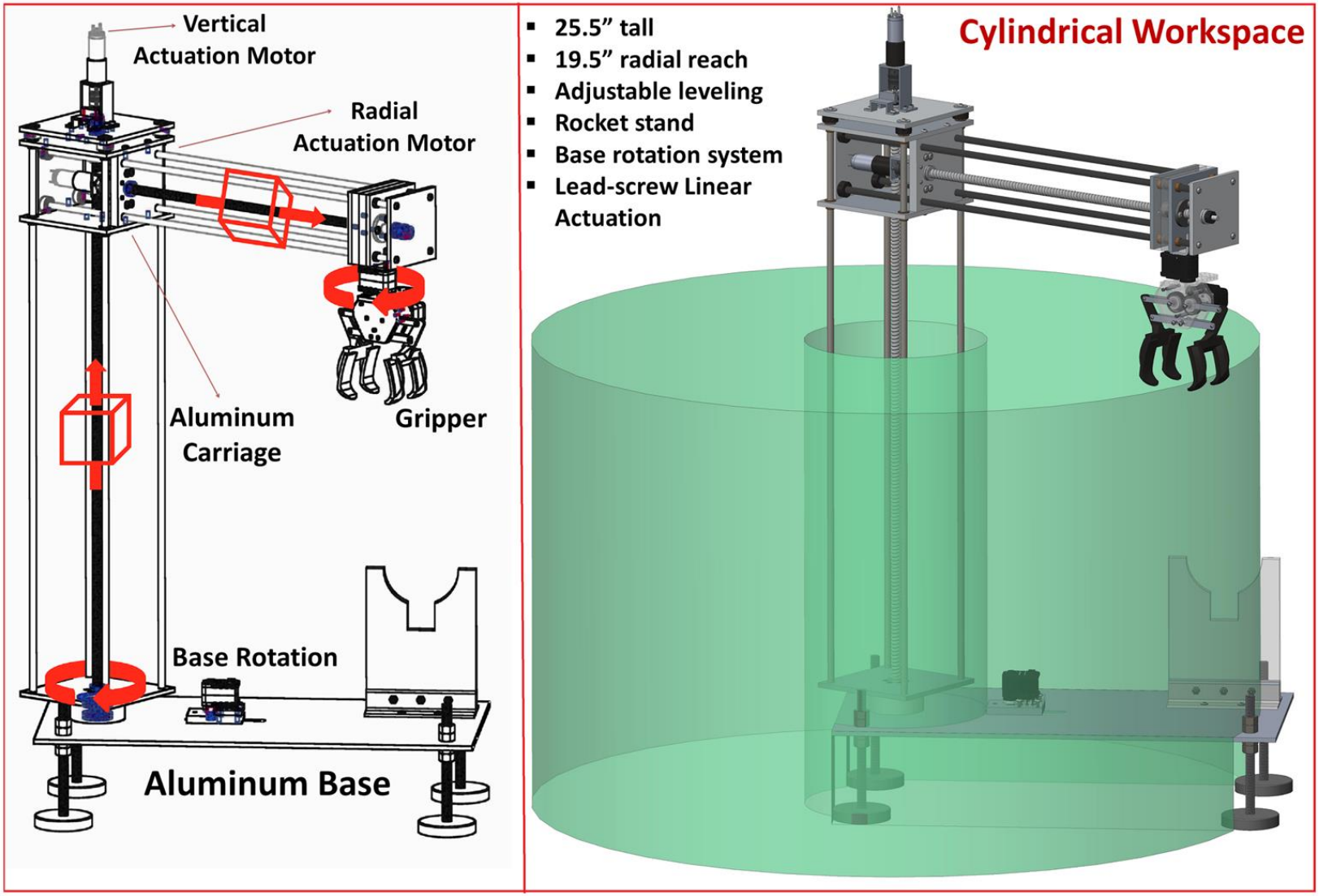
Vanderbilt Aerospace Club

Pranav Srinivas Kumar, William Emfinger, Dexter Watkins, Benjamin Gasser, Connor Caldwell, Frederick Folz, Alex Goodman, Christopher Lyne, Jacob Moore, Cameron Ridgewell, Robin Midgett and Amrutur Anilkumar

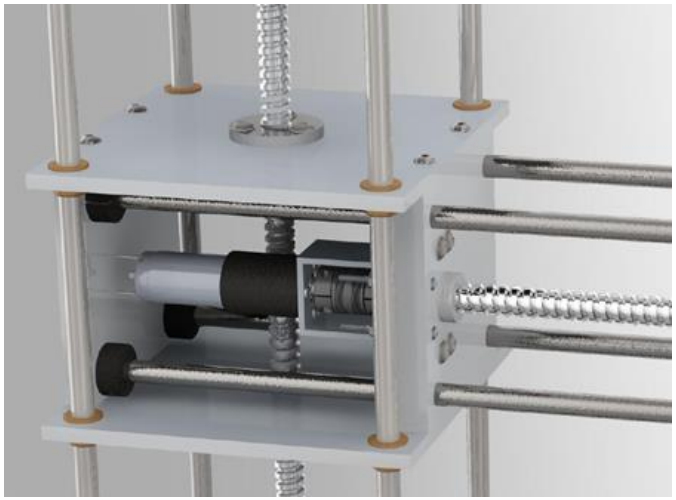
NASA Student Launch Competition

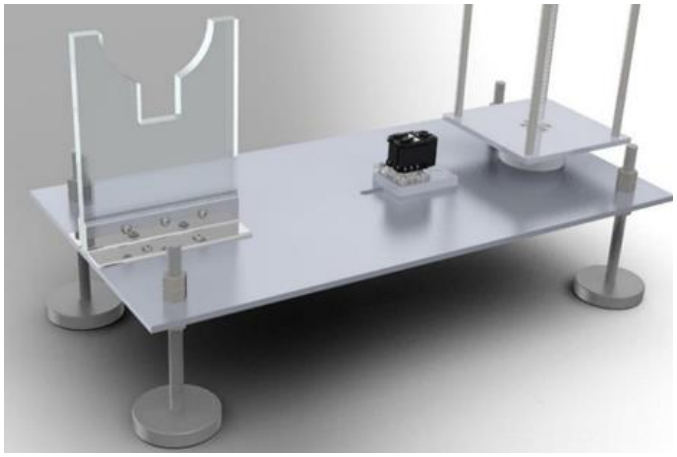
- Research-based Competition: Stimulate Rapid, Low-cost Development of Rocket Propulsion & Space Exploration Systems
- 8 Month Cycle: Design, Fabrication, and Testing of Flight Vehicles, Payloads and Ground Support Equipment
- 2014-2015 Competition: Simulate a **Mars Ascent Vehicle** (MAV)
  - Perform Sample Recovery from the Martian Surface
  - Design & Deploy an AGSE Robot
  - Autonomously retrieve a sample off the ground
  - Store sample in the payload bay of the rocket
  - Launch MAV rocket to an altitude of 3000 ft. and recover sample

AGSE Overview

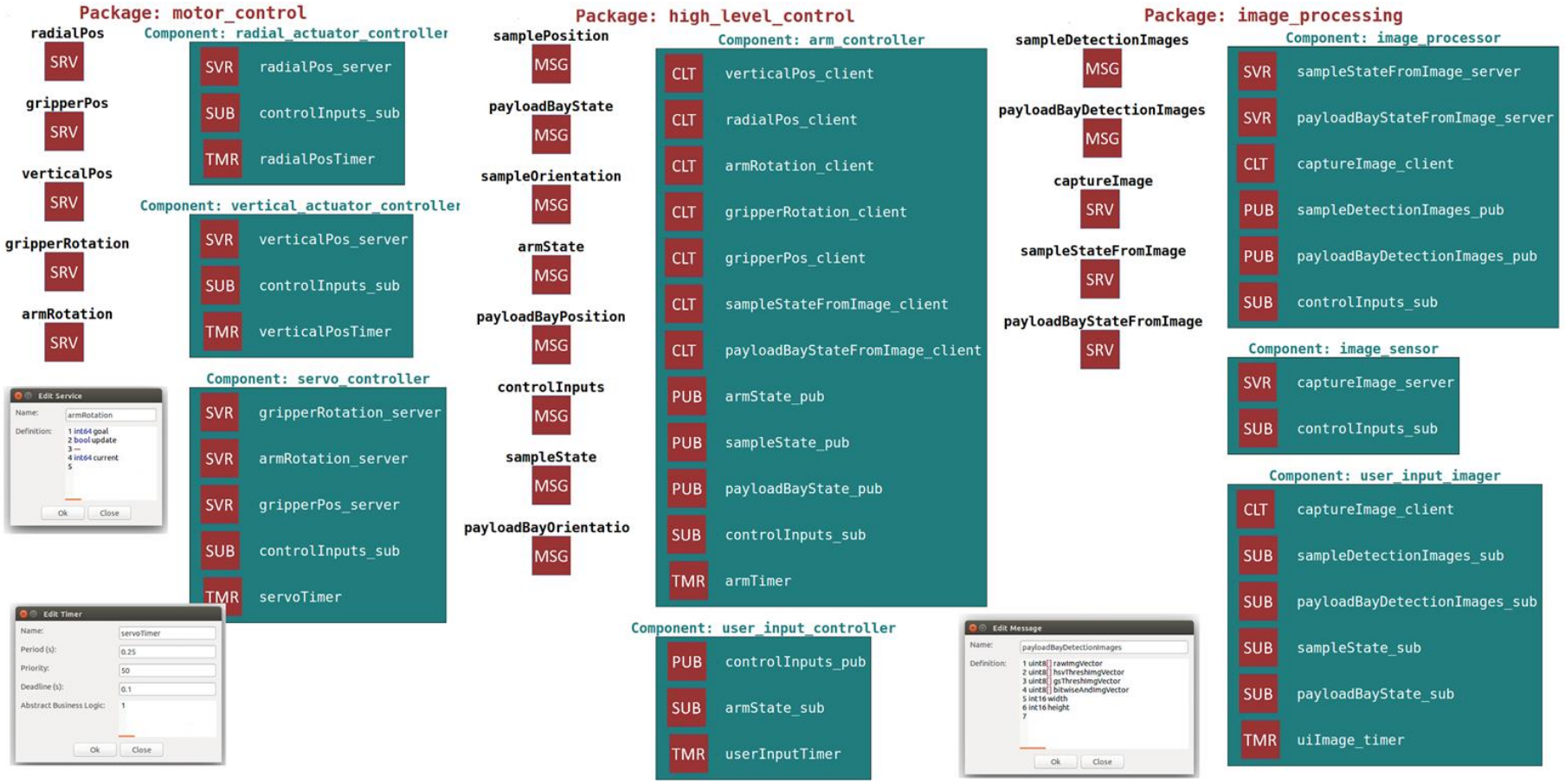


Mechanical Construction





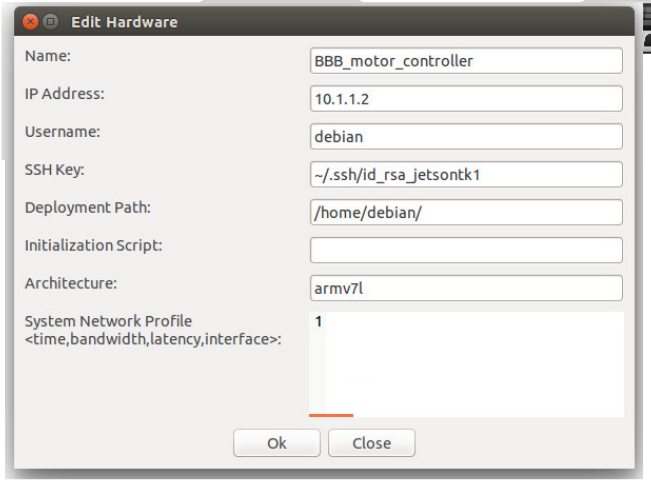
ROSMOD AGSE Software Model



AGSE Software Deployment

Hardware Model: agse

Device: BBB_motor_controller **Device: BBB_user_input** **Device: Jetson_TK1**



Deployment Model: agse

Node: arm
- arm_controller_i

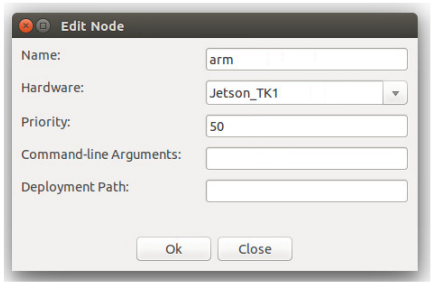
Node: detector
- image_processor_i

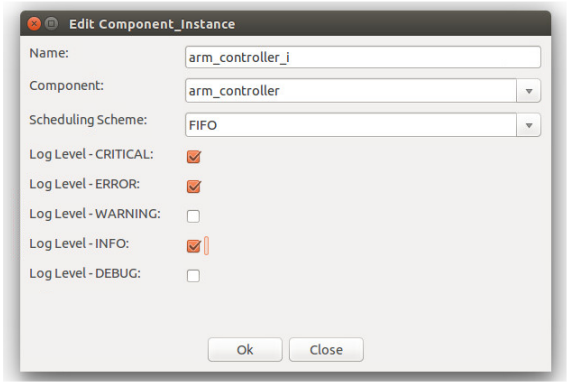
Node: positioning
- vertical_controller_i
- radial_controller_i

Node: imager
- image_sensor_i

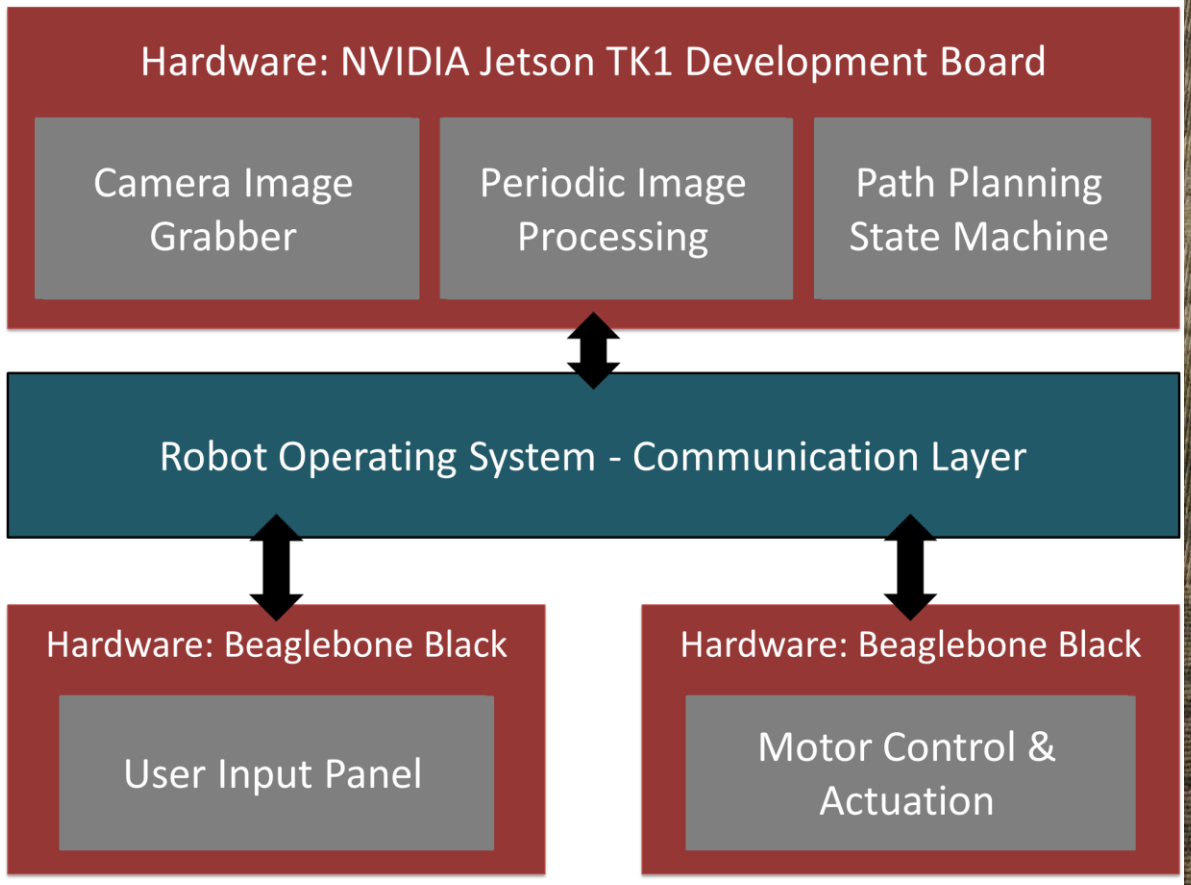
Node: servos
- servo_controller_i

Node: user_input
- user_input_controller_i
- user_input_imager_i





ROSMOD AGSE Deployment

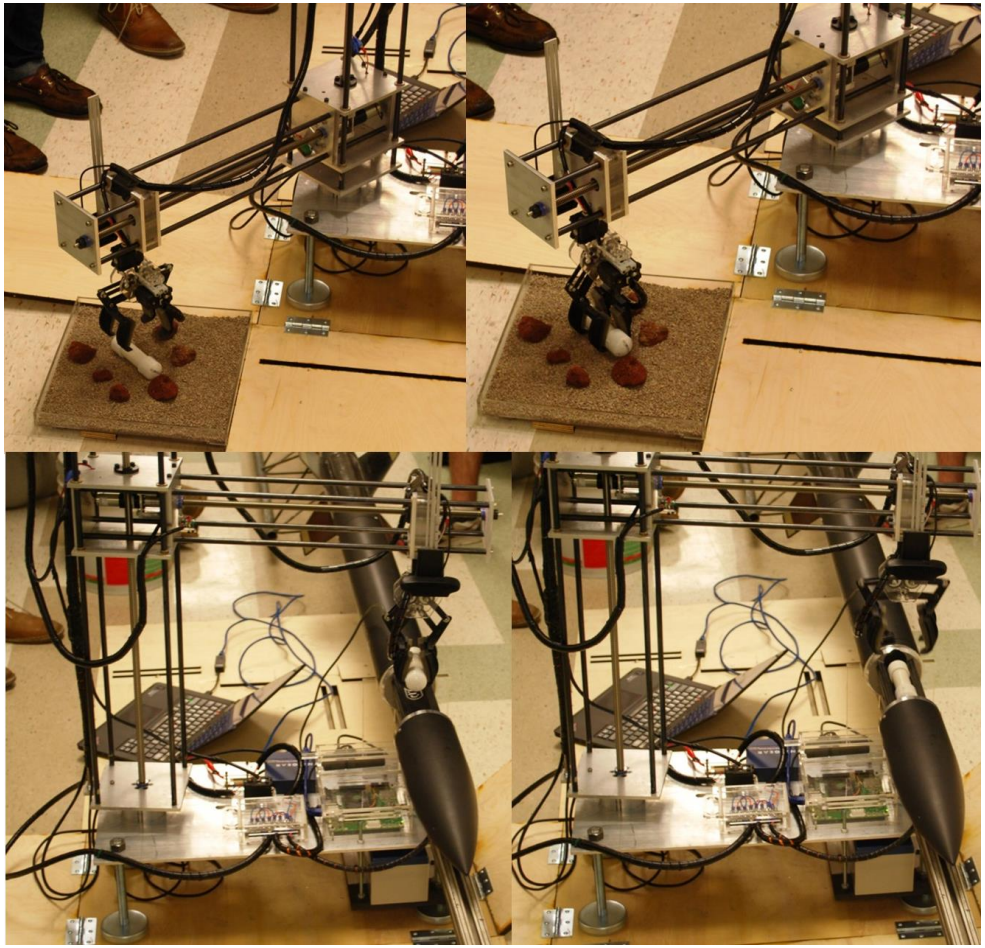


Software Rapid Prototyping

- Iterative Design-Generate-Deploy-Test Cycle
- ROSMOD generated nearly 60% of the AGSE software (6000+ lines)
  - Takes a few seconds
  - Preserves already written code
  - Developers fill in the missing pieces
  - Quick and easy integration with external libraries e.g. OpenCV
- Overall software frequently redesigned and tweaked
- Large portion of code development in under 3 weeks
  - Difficult without ROSMOD, especially with our small team

Competition Highlights

- Long Night before Competition
  - Dynamixel AX12A Servo Failure
  - Replace with spare MX28T
  - Difficult communication protocol & mounting footprint
  - Mount new servo!
  - Fix servo_controller package!
- Sample Recovery in under 4.5 mins.
- We won ☺
- Earned overall *Autonomous Ground Support Equipment Award*



Links

ROSMOD GitHub Organization
https://github.com/rosmod

AGSE Software GitHub Repository
https://github.com/finger563/agse2015

Vanderbilt Aerospace Club
http://www.vanderbilt.edu/USLI/2015/