ROSMOD

Pranav Srinivas Kumar, William Emfinger & Gabor Karsai

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 Corporation. 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 Corporation.

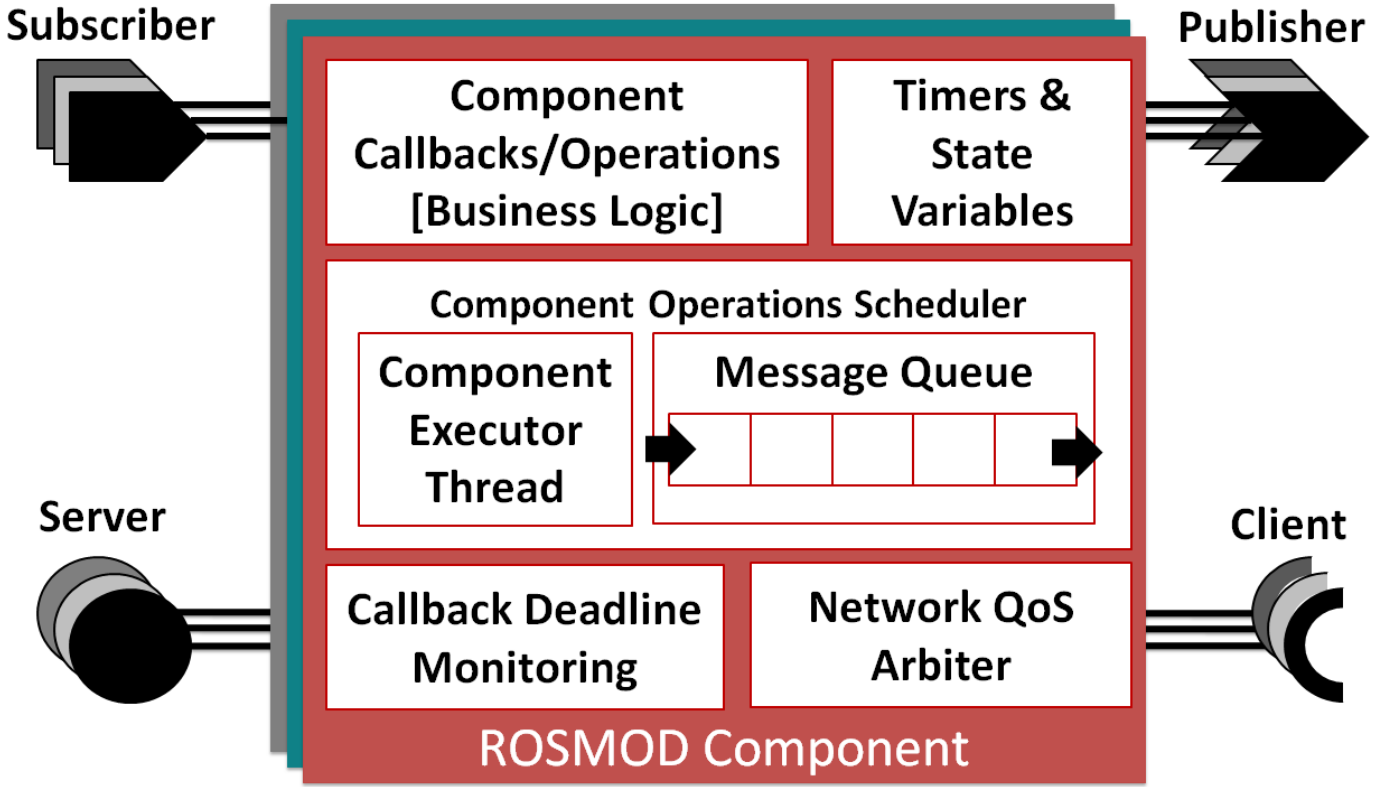
Introduction

- **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 robotic applications on embedded devices
- Provides a Graphical User Interface & Rendering Platform
- 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

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

ROSMOD Component



Component Model

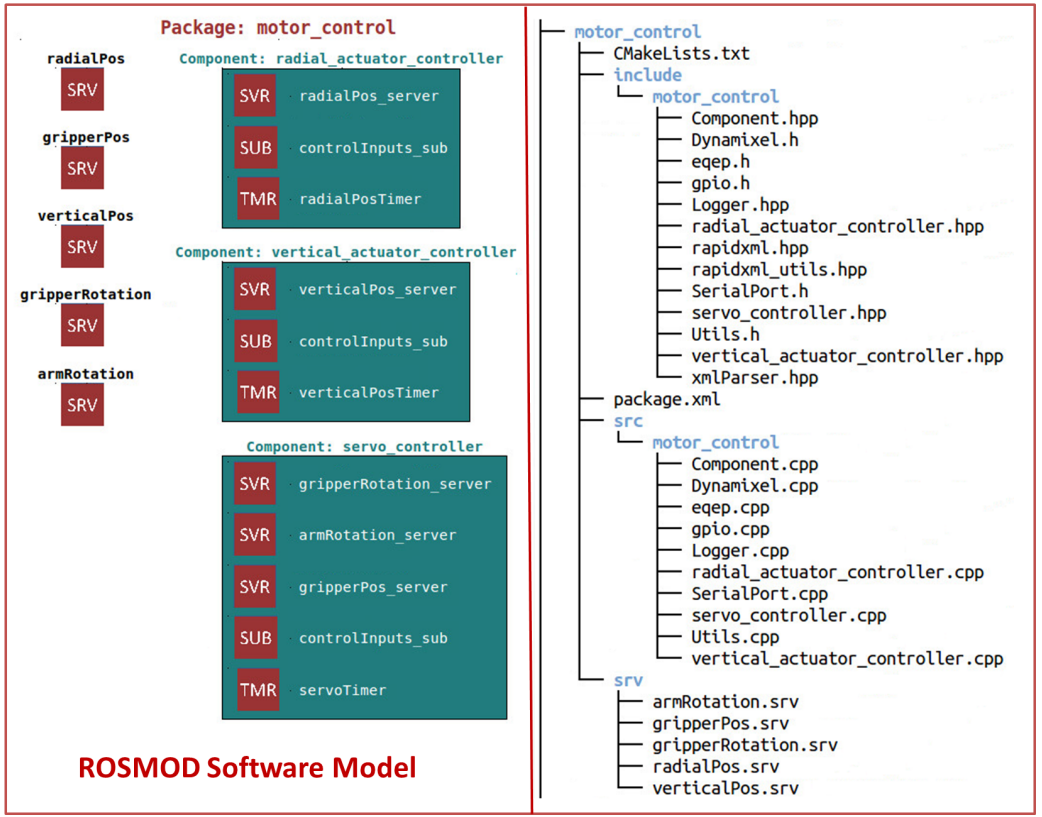
- Each Component has a Message Queue
- Each Component has a *single* executor thread
- Each Component exposes *operations* through port interfaces.
- Message Queue receives operation requests from external entities
- Component Operation Scheduler schedules a request from the queue
- Requests are processed based on a scheduling scheme e.g. PFIFO, EDF
- Operation Execution is nonpreemptive
- Single threaded operation execution helps avoid synchronization primitives and locking mechanisms

ROSMOD Projects

- Software Model
  - Represents a ROS Workspace
  - Define Messages, Services and Components
- Hardware Model
  - Define Hardware devices, IP addresses, architecture, SSH keys etc.
- Deployment Model
  - Define ROS Nodes (Processes)
  - Instantiate Components (from Software Model) in ROS Nodes
    - Each node spawns an executor thread per component instance
  - Define a ROS Node to Hardware mapping

Workspace Generation

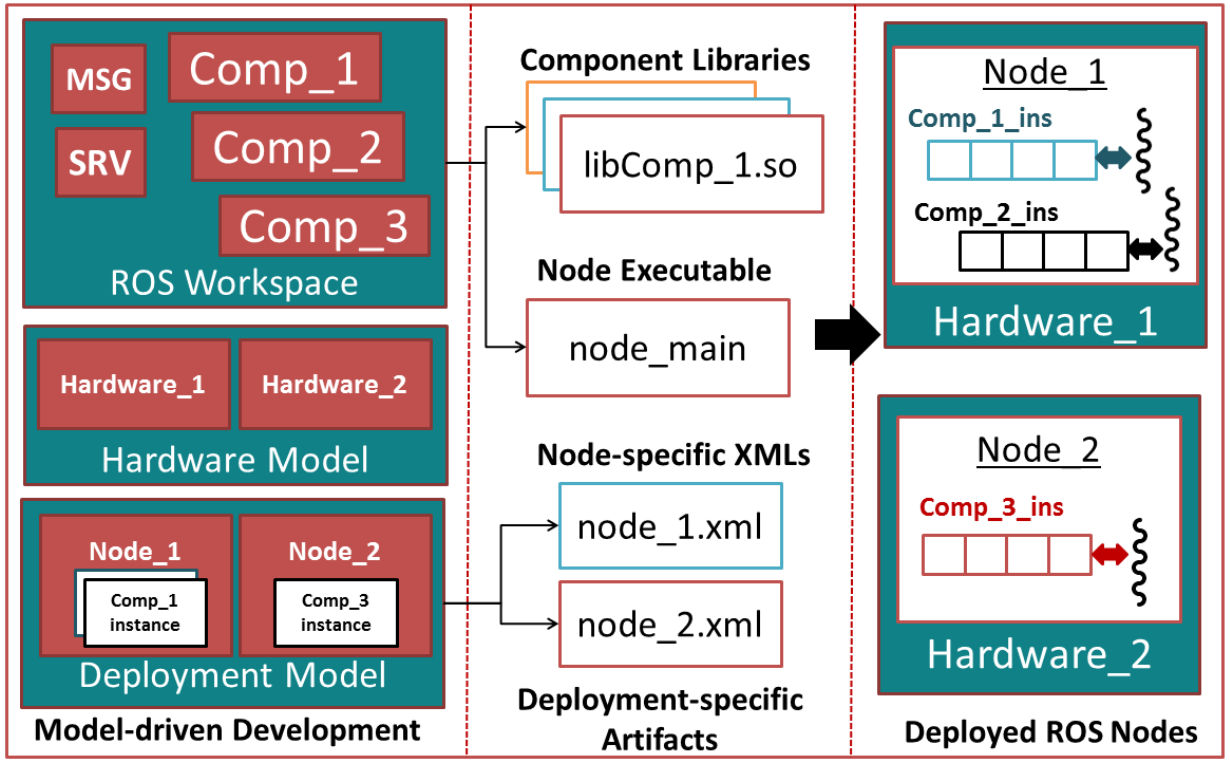
- C++ Classes for each ROSMOD component
- Package-specific Message and Service files
- Logging and XML parsing framework
- Build system files
- Code preservation markers & Doxygen comments
- Following ROS Package guidelines



ROSMOD Application Development

- Prepare a ROSMOD Model of the Application
- Generate the ROS Workspace using ROSMOD
  - Generated code includes Code-Preservation Markers
- Add *business logic code* to generated skeleton callbacks/operations
  - Server operations, Timer callback operations etc.
- Add new components, ports, messages etc. to the Model using the GUI
- 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 App!
- On-the-fly feature additions and rapid prototyping

Software Deployment Infrastructure



- Dynamically Loadable Libraries, one for each Component Definition
- Generic ROS node executable
- Configuration XML for each ROS Node
- Parallel Deployment in each Hardware
- Runtime Monitoring with PID and Logs

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

NASA Student Launch Competition, 2014-2015

Vanderbilt Aerospace Club

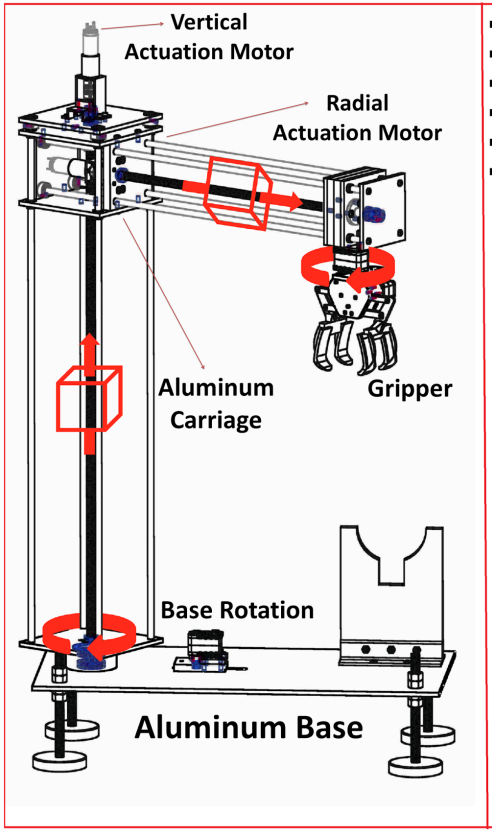
Pranav Srinivas Kumar, William Emfinger, Dexter Watkins, Benjamin Gasser, Connor Caldwell, Frederick Folz, Alex Goodman, Christopher Lyne, Jacob Moore, Cameron Ridgewell, Robin Midgett and Amrutur Anilkumar

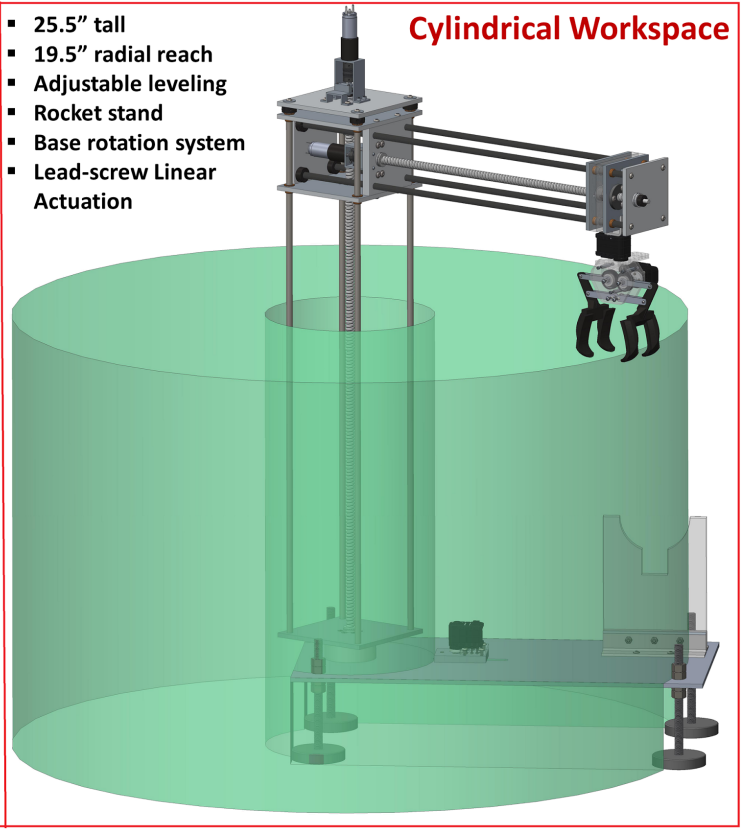
NASA Student Launch Competition

- Research-based Competition: Stimulate Rapid, Low-cost Development of Rocket Propulsion & Space Exploration Systems
- 8 Months 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 rocket to an altitude of 3000 ft. and recover sample

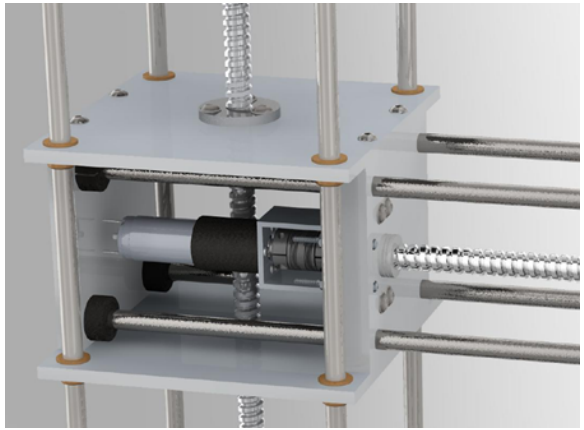
AGSE Overview

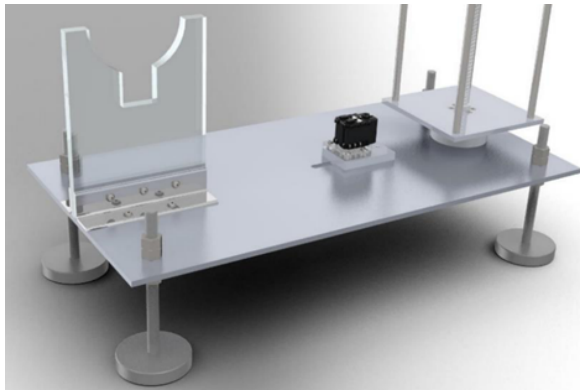
- Sample Retrieval
- Design Goals
  - Reliability
  - Simplicity
  - Robustness
- Design Considerations
  - Minimum degrees of freedom
  - Image Processing
  - DRE system
  - MDD workflow



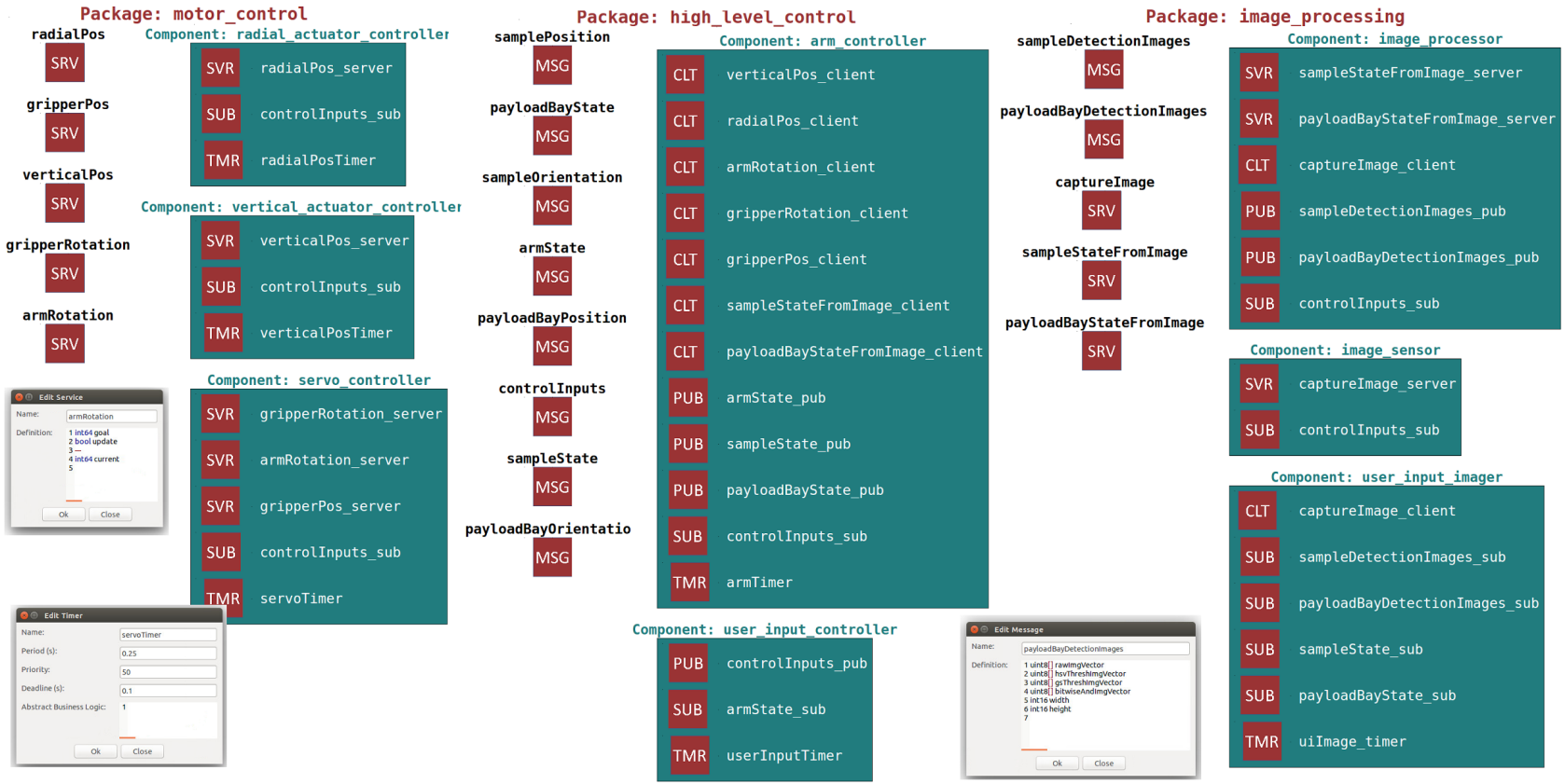


Mechanical Construction





ROSMOD AGSE Software Model

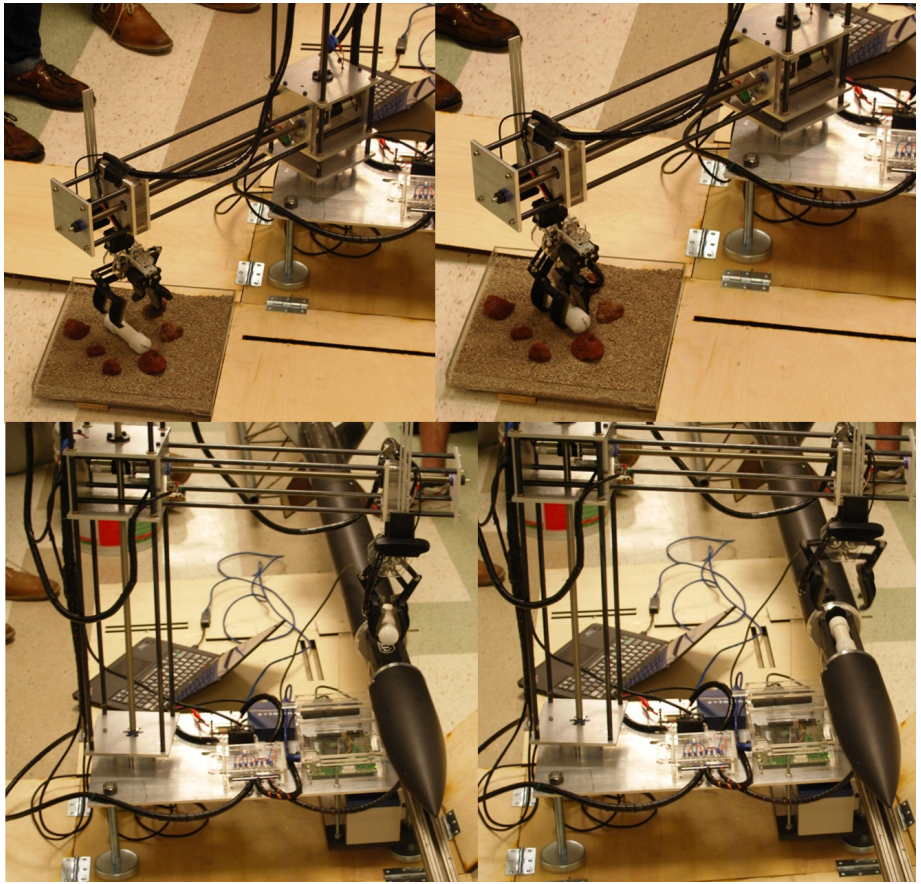


Software Rapid Prototyping

- Iterative Design-Generate-Deploy-Test cycle
- ROSMOD generated nearly 60% of the AGSE software (6,000+ lines)
  - Takes but 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
  - Improbable without ROSMOD

Competition Highlights

- Long Night before Competition:
  - Dynamixel AX12A Servo Failure
  - Replace with spare MX28T
  - Different communication protocol & mounting footprint
  - Mount new servo!
  - Fix *servo_controller* package!
- Sample Recovery in under 4.5 mins
- Earned overall *Autonomous Ground Support Equipment Award*
- Won the whole competition!



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/