CLARAty: Towards Standardized Abstractions and Interfaces for Robotics Systems
Coupled Layer Architecture for Robotic Autonomy

Issa A.D. Nesnas

Jet Propulsion Laboratory
In Collaboration with
Ames Research Center
Carnegie Mellon University
University of Minnesota

ICRA 2005 – Principle and Practice of Software Development in Robotics
April 18-22, 2005
Motivation
Various Rovers Developed by JPL/NASA
With Different Mobility Mechanisms
Problem and Approach

• Problem:
  – Difficult to share software across systems
  – Different hardware/software infrastructure
  – No standard protocols and APIs
  – No flexible code base of robotic capabilities

• Approach
  – Unified robotic framework
  – Capture and integrate legacy algorithms
  – Enable faster technology development
  – Operate heterogeneous robots
Would like to support …

Custom Rovers

Manipulators

COTS Systems

Reconfigurable Robots
Challenges in Interoperability

- Mechanisms and Sensors
- Hardware Architecture
- Modular and Reusable Software Components
Locomotion

Rocky 7

Rocky 8
Reusable Wheeled Locomotion Algorithms

General flat terrain algorithms and specialized full DOF algorithms

(a) Skid Steering
(no steering wheels)

(b) Tricycle
(one steering wheel)

(c) Two-wheel steering

(d) Partially Steerable
(e.g. Sojourner, Rocky 7)

(e) All wheel steering
(e.g. MER, Rocky8, Fido, K9)

(f) Steerable Axle
(e.g. Hyperion)

ATRV (a)
Sojourner (d)
Rocky 7 (d)
FIDO (e)
Rocky 8 (e)
K9 (e)
Manipulators and Sensor Suites

- Custom Analog Sun Sensor
- 3 Accels z-axis gryo
- 3 DOF Mast
- 2 DOF Arm
- Camera Sun Sensor
- 4 DOF Mast
- 6 DOF IMU
- 4 DOF Arm

• Given different capabilities, how much reuse can be achieved?
Challenges in Interoperability

- Mechanisms and Sensors
- Hardware Architecture
- Modular and Reusable Software Components
Distributed Hardware Architecture

Rocky 8

Sun Sensor

Rocky Widgets
- Single-axis controllers
- Current sensing
- Digital I/O
- Analog I/O

Compact PCI
- x86 Arch
- Wireless E/net
- 1394 FireWire
- I2C Bus

RS232
I2C

K9

PC104+
- x86 Arch
- Wireless E/net
- 1394 FireWire
- RS422 serial Bus

1394 Bus

RS232
I2C

IMU

Potentiometers

Actuator/Encoders

PIC-SERVOs
- Single-axis controllers
- Current sensing

IMU
Custom Architecture/Variability

Rocky 7

Parallel Custom Interface
MUX/Handshaking

PID Controllers

Actuator/Encoders

Compact PCI
PPC 750 Arch
Framegrabbers
Digital I/O
Analog I/O
Wireless Ethernet

Video Switcher

Accels
Gyros

AIO

Potentiometers
Centralized Hardware Mapped Architecture

**Fido**

- Actuator/Encoders
- PID Control in Software
- Video Switcher
- PC104+
  - x86 Arch
  - Framegrabbers
  - Digital I/O
  - Analog I/O
  - Wireless Ethernet

- RS232 Serial
- IMU
- Potentiometers
Challenges in Interoperability

- Mechanisms and Sensors
- Hardware Architecture
- Modular and Reusable Software Components
Designated Target Tracking for Single-Cycle Instrument Placement

Integration of Complex Algorithms

- I/O, motion control
- Trajectory Generation
- Rough Terrain Locomotion
- Odometry Pose Estimation
- Stereo Processing
- Visual Odometry
- *Obstacle avoidance*
- Mast Control
- Visual Tracking
Navigation with Path Planning on Two Rovers

Complex Algorithms on different Platforms

• I/O, motion control
• Trajectory Generation
• Rough Terrain Locomotion
• Odometry Pose Estimation
• Stereo Processing
• Visual Odometry
• Navigation (Morphin)
  – Obstacle avoidance
  – Path Planning
And with a Simulated Rover

QuickTime™ and a Video decompressor are needed to see this picture.
Technical Approach
A Two-Layered Architecture

CLARAty = Coupled Layer Architecture for Robotic Autonomy

**THE DECISION LAYER:**
Declarative model-based
Global planning

**INTERFACE:**
Access to various levels
Commanding and updates

**THE FUNCTIONAL LAYER:**
Object-oriented abstractions
Autonomous behavior
Basic system functionality

Adaptation to a system
The Decision Layer

- General Planners (e.g. CASPER)
- Executives (e.g. TDL)
- Activity Database
- Plans
- Rover Models
- FL Interface
The Functional Layer

- Navigation
- Path Planning
- Estimation
- Transforms
- Motion Control
- Input/Output
- Rover
- Behaviors
- Manipulation
- Vision
- Math
- Communication
- Hardware Drivers
- Simulation
- Locomotion
- Science
- Sensor

Adaptations:
- Rocky 8
- FIDO
- K9
- Rocky 7
Standardizing Base Abstractions

Abstractions

<table>
<thead>
<tr>
<th>CoordMotionSystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>_position: Vector</td>
</tr>
<tr>
<td>_velocity: double</td>
</tr>
<tr>
<td>_acceleration: double</td>
</tr>
</tbody>
</table>

| change_position(Location& loc) |
| change_velocity(double vel) |
| set_position(Location& loc) |
| set_velocity(double vel) |
| get_position() |
| get_velocity() |
| _compute_angles() |

Wheeled_Locomotor

Legged_Locomotor

Manipulator

Leg

Wheel

Steerable Wheel

Abstractions

-Wheel Locomotormotor
  -Vehicle_Interface & _interface
  -DPoint _goal
  -double _goal_heading
  -Vector<double> motor_pos
  -Vector<double> motor_delta_pos
  -Vector<double> motor_vel
  +Wheel_Locomotor(Vehicle_Interface & vi)
  +~Wheel_Locomotor()
  +bool is_driving()
  +bool is_steering()
  +bool is_moving()
  +void move(Drive_Command &dc)
  +void move(Drive_Sequence &seq)
  +void move( double length, double heading = 0)
  +void move( const DPoint& a, const DPoint& b)
  +void move( const DPoint& a, const DPoint& b, const double heading)
  +void move_A_to_B(const DPoint& a, const DPoint& b)
  +void move_A_to_B(const DPoint& a, const DPoint& b, double heading)
  +void drive_continuous(const double linear_velocity, const double angular_velocity)
  +void drive_continuous(const double linear_velocity, const double direction, const double angle)
  +void position_drive()
  +void velocity_drive()
  +Drive_Sequence & generate_drive_sequence(const DPoint& a, const DPoint& b)
  +Drive_Sequence & generate_drive_sequence(const DPoint& a, const DPoint& b, const double heading)
  +void calibrate_motors()
  +void set_threshold_steer_angle(double threshold)
  +void wait_until_driving_done(double percent=100.0)
  +void wait_until_steering_done(double percent=100.0)
  +void wait_until_done(double percent=100.0)
  +void wait_until_drive_loop_done()

APIs and Behaviors

State Machines

Runtime Models

Motor

Thread 1 (controls)
  - motor.change_position(2PI)
  - do other things

Thread 2 (monitors)
  - based on watchdog
  - position = motor.get_position()
  - if (position > x) motor.stop()

Thread 1 (controls) and Thread 2 (monitors) are used for monitoring and controlling the motor's position.
Unified Mechanism Model

Mechanism Tree

Bodies and Joints

Ground Body

Body Tree

Body0

Body1

Body2

Body3

Body4

Body5

B1

B2

B3

B4

B5

C1

C2

C3

C4

C5

Relative to body reference frame

Arm mount frame

Body reference frame

Camera mount frame

Sensor mount frame

Center of mass

Leaves of tree define finest shape

Bounding Shape Tree

Bounding Shapes

Resolution Levels

Coarse Shape

Finer Shape

Finest Shape
Connecting Bodies and Joints
Architectural Traverse Example

- **Rover**
  - K9 Rover

- **Navigator**
  - Gestalt Navigator

- **Mapper**
  - Grid Mapper

- **Terrain Sensor**
  - Stereo Engine
  - Stereo Camera
    - Camera R
    - Camera L

- **Locomotor**
  - R8_Locomotor

- **Path Planner**
  - D* Path Planner

- **Global Cost Func**

- **Decision Layer**
  - Commanding and State Updates
  - Path Information

- **Asynchronous**
  - Rate Set at: 5 Hz
  - Rate Set at: 8 Hz
  - Rate Set at: 10 Hz

- **Synchronous/or Asynchronous**
  - Rate Set at: 10Hz used by other activities

- **Decision Layer**
  - Active

April 18, 2005

ICRA2005 - I.A.N. 26
Architectural Traverse Example

- Rover
  - K9 Rover
- Navigator
  - R7/Soj Navigator
- Mapper
  - Obstacle Mapper
- Terrain Sensor
  - Stereo Engine
  - Stereo Camera
    - Camera R
    - Camera L
  - Stereo Processor
    - JPL Stereo
- Locomotor
  - ROAMS_Locomotor
- Pose Estimator
  - EKF Pose Estimator

Asynchronous
- Rate Set at: 5 Hz
- Rate Set at: 8 Hz
- Rate Set at: 10Hz

Synchronous/or Asynchronous:
- Rate Set at: 10 Hz used by other activities
Some Results on Reusability
### Some Software Inter-operability Statistics

<table>
<thead>
<tr>
<th>Algorithm Description</th>
<th>1st Adaptation</th>
<th>Subsequent Adaptations (time in days)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>Time</td>
</tr>
<tr>
<td>3D Extended Kalman Filter for Rover Pose Estimation</td>
<td>Rocky 8</td>
<td>60</td>
</tr>
<tr>
<td>3D Locomotion for wheeled vehicles</td>
<td>Rocky 7</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>FIDO (JPL)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>K9 (ARC)</td>
<td>4</td>
</tr>
<tr>
<td>Morphin Navigator</td>
<td>Rocky 8</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>K9 (ARC)</td>
<td>4</td>
</tr>
<tr>
<td>Mast Control Software</td>
<td>Dexter</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>FIDO (JPL)</td>
<td>4</td>
</tr>
</tbody>
</table>
# Code Reusability for Motion Control

<table>
<thead>
<tr>
<th>Rocky 7 Modules</th>
<th>Lines of Code</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controlled Motor</td>
<td>2,652</td>
<td>Reusable</td>
</tr>
<tr>
<td>Input Output</td>
<td>2,690</td>
<td>Reusable</td>
</tr>
<tr>
<td>Bits</td>
<td>1,580</td>
<td>Reusable</td>
</tr>
<tr>
<td>Resources (Timers, etc)</td>
<td>725</td>
<td>Reusable</td>
</tr>
<tr>
<td>Rocky 7 Motor</td>
<td>927</td>
<td>Non-reusable</td>
</tr>
<tr>
<td>Rocky 7 H/W Maps</td>
<td>841</td>
<td>Non-reusable</td>
</tr>
<tr>
<td>Motor Controller LM629</td>
<td>1,143</td>
<td>Reusable</td>
</tr>
<tr>
<td>Digital I/O Board (S720)</td>
<td>576</td>
<td>Reusable</td>
</tr>
<tr>
<td>PCI Components</td>
<td>329</td>
<td>Reusable</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>11,463</strong></td>
<td><strong>85%</strong></td>
</tr>
<tr>
<td><strong>Total Reusable</strong></td>
<td><strong>9,255</strong></td>
<td><strong>85%</strong></td>
</tr>
<tr>
<td><strong>Total Reusable - Strict</strong></td>
<td><strong>7,788</strong></td>
<td><strong>67%</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FIDO Modules</th>
<th>Lines of Code</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controlled Motor</td>
<td>2,652</td>
<td>Reusable</td>
</tr>
<tr>
<td>Trajectory Generator</td>
<td>691</td>
<td>Reusable</td>
</tr>
<tr>
<td>PID Controller</td>
<td>997</td>
<td>Reusable</td>
</tr>
<tr>
<td>Input Output</td>
<td>2,690</td>
<td>Reusable</td>
</tr>
<tr>
<td>Bits</td>
<td>1,580</td>
<td>Reusable</td>
</tr>
<tr>
<td>Resources (Timers, etc)</td>
<td>725</td>
<td>Reusable</td>
</tr>
<tr>
<td>Common Definitions</td>
<td>2,380</td>
<td>Reusable</td>
</tr>
<tr>
<td>FIDO Motor</td>
<td>2,086</td>
<td>Non-reusable</td>
</tr>
<tr>
<td>FIDO H/W Maps</td>
<td>1,494</td>
<td>Non-reusable</td>
</tr>
<tr>
<td>Encoder Counter (ISA P 400)</td>
<td>463</td>
<td>Reusable - H/W</td>
</tr>
<tr>
<td>Analog Input Board (MSI P415)</td>
<td>519</td>
<td>Reusable - H/W</td>
</tr>
<tr>
<td>Analog Output Board (MSI P460)</td>
<td>462</td>
<td>Reusable - H/W</td>
</tr>
<tr>
<td>Digital I/O Board (MSI P560)</td>
<td>602</td>
<td>Reusable - HCTL</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>17,341</strong></td>
<td><strong>79%</strong></td>
</tr>
<tr>
<td><strong>Total Reusable</strong></td>
<td><strong>14,463</strong></td>
<td><strong>86%</strong></td>
</tr>
<tr>
<td><strong>Total Reusable - Strict</strong></td>
<td><strong>12,292</strong></td>
<td><strong>68%</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rocky 8 Modules</th>
<th>Lines of Code</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controlled Motor</td>
<td>2,652</td>
<td>Reusable</td>
</tr>
<tr>
<td>Trajectory Generator</td>
<td>691</td>
<td>Reusable</td>
</tr>
<tr>
<td>PID Controller</td>
<td>997</td>
<td>Reusable</td>
</tr>
<tr>
<td>Input Output</td>
<td>2,690</td>
<td>Reusable</td>
</tr>
<tr>
<td>Bits</td>
<td>1,580</td>
<td>Reusable</td>
</tr>
<tr>
<td>Resources (Timers, etc.)</td>
<td>725</td>
<td>Reusable</td>
</tr>
<tr>
<td>Rocky 8 Motor</td>
<td>1,180</td>
<td>Non-reusable</td>
</tr>
<tr>
<td>Rocky 8 H/W Maps</td>
<td>626</td>
<td>Non-reusable</td>
</tr>
<tr>
<td>Widget Board Software</td>
<td>2,126</td>
<td>Reusable - widget</td>
</tr>
<tr>
<td>Motor Controller HCTL</td>
<td>900</td>
<td>Reusable - HCTL</td>
</tr>
<tr>
<td>I2C Master</td>
<td>1,165</td>
<td>Reusable - I2C</td>
</tr>
<tr>
<td>I2C Master Tracii</td>
<td>1,223</td>
<td>Reusable - Tracii</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>16,314</strong></td>
<td><strong>89%</strong></td>
</tr>
<tr>
<td><strong>Total Reusable</strong></td>
<td><strong>14,365</strong></td>
<td><strong>87%</strong></td>
</tr>
<tr>
<td><strong>Total Reusable - Strict</strong></td>
<td><strong>11,464</strong></td>
<td><strong>67%</strong></td>
</tr>
</tbody>
</table>
## Code Resuability for Locomotion Example

<table>
<thead>
<tr>
<th>Module</th>
<th>Lines of Code</th>
<th>Status</th>
<th>Depends On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wheel Locomotor</td>
<td>1445</td>
<td>Reusable</td>
<td>Motion Sequence, 1D Solver, Homogeneous Transforms</td>
</tr>
<tr>
<td>Motion Sequence</td>
<td>540</td>
<td>Reusable</td>
<td>Vector</td>
</tr>
<tr>
<td>Matrix, Vector, Array</td>
<td>1083</td>
<td>Reusable</td>
<td>-</td>
</tr>
<tr>
<td>1D Solver</td>
<td>356</td>
<td>Reusable</td>
<td>-</td>
</tr>
<tr>
<td>Location, Homogeneous Transforms</td>
<td>341</td>
<td>Reusable</td>
<td>Rotation Matrix, Point 2D</td>
</tr>
<tr>
<td>Rotation Matrices</td>
<td>435</td>
<td>Reusable</td>
<td>-</td>
</tr>
<tr>
<td>Point 2D</td>
<td>131</td>
<td>Reusable</td>
<td>-</td>
</tr>
<tr>
<td>Controlled Motor</td>
<td>2080</td>
<td>Reusable</td>
<td></td>
</tr>
<tr>
<td>Rocky 8 Locomotor</td>
<td>250</td>
<td>Non-reusable</td>
<td>Rocky 8 Motor</td>
</tr>
<tr>
<td>Rocky 8 Motor</td>
<td>334</td>
<td>Non-reusable</td>
<td>Widget Motor, etc...</td>
</tr>
<tr>
<td>Total</td>
<td>6995</td>
<td></td>
<td>584 (non-reusable)</td>
</tr>
<tr>
<td>Total Reusable</td>
<td>~92%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

• Use abstraction to master complexity
• Encapsulate and abstract hardware variations
• Provide multi-level access through Decision Layer for fault diagnosis and recovery
• Use domain expertise to guide design
• Make all assumptions explicit
• Stabilize external interfaces rapidly
• Document processes and products well
• Avoid over-generalization - define scope
• Encapsulate system specific runtime models
• Do not comprise performance - least common denominator solutions are unacceptable in hw/sw interactions
• Standardize Hardware
Acknowledgements

CLARAty Team (multi-center)

Jet Propulsion Laboratory
- ROAMS/Darts Team
- CLEaR Team
- Instrument Simulation Team
- Machine Vision Team
- FIDO Team

Ames Research Center
- K9 Team

Carnegie Mellon University
Thank you for your Attention
Examples of CLARAty Reusability

![Diagram showing examples of CLARAty reusability, with controlled and non-reusable components.

Non-Resuable Layer:
- Fido_Motor
- R8_Motor
- Sim_Motor
- R7_Motor

Reusable Components:
- ControlledMotor
- Joint
- Linear_Axis
- Fido_Motor
- R8_Motor
- Sim_Motor
- R7_Motor

HW reusable Components:
- Counter
- PID Controller
- DIO
- Analog_out
- Analog_in
- HCTL_Chip
- Trajectory
- Trajectory_Generator

Non-reusable Components:
- Resuable
- HW reusable
- Non-reusable
- Mz<Type>
- MSI P460
- MSI P430
- MSI P415
- MSI P430
- LM629_Chip
- R7_MC_Board

April 18, 2005
Adapting to a Rover

![Diagram of the rover simulation system]

- **Decision Layer**: Rocky 8 Models/Heuristics
- **Connector**: Multi-level access Connector
- **Generic Functional Layer**: Rocky 8 Specialized Classes & Objects
- **Simulation**
- **Hardware Drivers**
Supported Platforms

Rocky 8
VxWorks x86
JPL

K9
Linux x86
Ames
JPL

Rocky 7
VxWorks ppc
JPL

FIDO
VxWorks x86
JPL

ATRV
Linux x86
CMU

ROAMS
Solaris Linux
JPL
CLARAty Team

NASA Ames Research Center
- Maria Bualat
- Sal Desiano
- Clay Kunz (Data Structure Lead)
- Eric Park
- Randy Sargent
- Anne Wright (Cog-E & Core lead)

• Carnegie Mellon University
  - David Apelfaum
  - Reid Simmons (Navigation lead)
  - Chris Urmson
  - David Wettergreen

• University of Minnesota
  - Stergios Roumeliotis
  - Yukikazu Hidaka

• Jet Propulsion Laboratory
  - Max Bajracharya (34) (Cog-E & Vision lead)
  - Edward Barlow (34)
  - Antonio Diaz Calderon (34)
  - Caroline Chouinard (36)
  - Gene Chalfant (34)
  - Tara Estlin (36) (Deputy Manager & Decision Layer lead)
  - Erann Gat (36)
  - Dan Gaines (36) (Estimation Lead)
  - Mehran Gangianpour (34)
  - Won Soo Kim (34) (Motion lead)
  - Michael Mossey (31)
  - Issa A.D. Nesnas (34) (Task Manager)
  - Richard Petras (34) (Adaptation lead)
  - Marsette Vona (34)
  - Barry Werger (34)

• OphirTech
  - Hari Das Nayar
Summary

• CLARAty provides a repository of reusable software components at various abstraction levels
• It attempts at capturing well-known robot technologies in a basic framework for researchers
• It publishes the behavior and interfaces of its components
• It allows researchers to integrate novel technologies at different levels of the architecture
• It is a collaborative effort within the robotics community
• It will run on multiple heterogeneous robots
Component Analysis

Generic Physical Comp

Public

Object Services

Queries

Creates

Links to

Members

Objects

Sub-object

HW Object

State Handler

Private

Internal Implementation

State 1

Local Estimation

Estimator

State Machines

- optional link
Some CLARAty Statistics
- ~170 Modules (reusable entity)
- ~31 Packages (module grps)
- ~3 rovers
- ~250,000 lines of C++ code
- ~Java/scripts/ and models
Example: Generic Controlled Motor

- Define generic capabilities independent of hardware
- Provide implementation for generic interfaces to the best capabilities of hardware
- Provide software simulation where hardware support is lacking
- Adapt functionality and interface to particular hardware by specialization inheritance
- Motor Example: public interface command groups:
  - Initialization and Setup
  - Motion and Trajectory
  - Queries
  - Monitors & Diagnostics
Example: collaborative development for locomotor

**Version 1.0**
- Designed for Rocky 7
- Used Motor class
- Separated wheel control from locomotion
- Built-in pose estimation

**Version 2.0**
- Generalized design for wheeled locomotors
- Full and partially steerable vehicle
- Used generic motor classes
- Implements fixed axle model
- Developed continuous driving
- Adapted to Rocky 8, Rocky 7, and Sim

**Version 3.0**
- Separated model from control
- Add separate locomotor state
- Add concept of wheel and steerable wheel, Drive Cmd, Drive Sequence
- Adapt to ATRV, Sim, Rocky 7, Rocky 8

**Version 4.0**
- Use device and telemetry infrastructure
- Add adaptation to K9

**Future**
- ARC - 2003

**Redesign/mature**
- JPL - 1998
- CMU - 2002
- JPL - 2001

**Add**
R8 Specific Rover Implementation

Non reusable Code

Reusable Code

- CoordMotionSystem
  - Implements general fwd & inv. kinematics & joint ctrl
- Locomotor
  - Attaches proper motors
  - Restricts Steering to 2 wheels
- Manipulator
- Wheeled Locomotor
- Arm
- Mast

- Motor
- ControlledMotor
  - R8 Motor
- BBMotor
- Trajectory
  - R8_Locomotor
- Trajectory_Generator
- Timers

- IO
- Digital_IO
- Analog_IO

- Widget Board
  - Widget DIO
  - Widget AIO
  - Widget Motor

- HCTL 1100 Chip

R8

R8_Locomotor

R8_Arm

R8_Mast
Capabilities of Wheel Locomotor

• Type of maneuvers:
  – Straight line motions (fwd / bkwd)
  – Crab maneuvers
  – Arc maneuvers
  – Arc crab maneuvers
  – Rotate-in-place maneuvers (arc turn r=0)

• Driving Operation
  – Non-blocking drive commands
  – Multi-threaded access to the Wheel_Locomotor class – e.g. one task can use Wheel_Locomotor for driving while the other for position queries
  – Querying capabilities during all modes of operation. Examples include position updates and state queries
  – Built-in rudimentary pose estimation that assumes vehicle follows commanded motion
R7 Specific Rover Implementation

- Non reusable Code
- Reusable Code

**CoordMotionSystem**

- Locomotor
- Manipulator
- LeggedLoc
- WheeledLoc
- RBLoc

**Motor**

- ControlledMotor
- BBMotor

**IO**

- Digital_IO
- Analog_IO

- VPAR10Board

**Device Drivers**

- LM629Motor
- LM629Chip

**R7 Specific Rover Implementation**

- Implements general fwd & inv. kinematics & joint ctrl

- **R7 Locomotor**
  - Attaches proper motors
  - Restricts Steering to 2 wheels

- **R7 Arm**
  - Split inv. Kinematics (overrides default)
  - Attaches proper motors
  - Attaches proper cameras for mast
  - Adds filter wheel

- **R7 Mast**
Why is robotic software “hard”?

• Software:
  – Software is large and complex
  – Has lots of diverse functionality
  – Integrates many disciplines
  – Requires real-time runtime performance
  – Talks to hardware

• Hardware:
  – Physical and mechanics are different
  – Electrical hardware architecture changes
  – Hardware component capabilities vary
What is CLARAty?

CLARAty is a unified and reusable software that provides robotic functionality and simplifies the integration of new technologies on robotic platforms.

A research tool for technology development and maturation.