

CLARAty: Improving Software Reliability for Robotic Space Applications

Coupled **L**ayer **A**rchitecture for **R**obotic **A**utonomy

Issa A.D. Nesnas

Jet Propulsion Laboratory,
California Institute of Technology

In Collaboration with

Ames Research Center
Carnegie Mellon University
University of Minnesota

49th Meeting of the IFIP 10.4 Working Group on Dependable and Fault Tolerant Computing
Tucson, AZ, USA, February 15-19, 2006



Presentation Overview



- Historical Antecedents
- Problem and Objectives
- What is CLARAty?
- Interoperability Success Stories
 - Navigation
 - Single-cycle Instrument Placement
- Challenges of Software Interoperability
- Challenges in Technology Infusion
- Architecture Overview
- Challenges Ahead



Application Domains

- Navigation
- Single-cycle Instrument Placement



Complex Algorithms on different Platforms

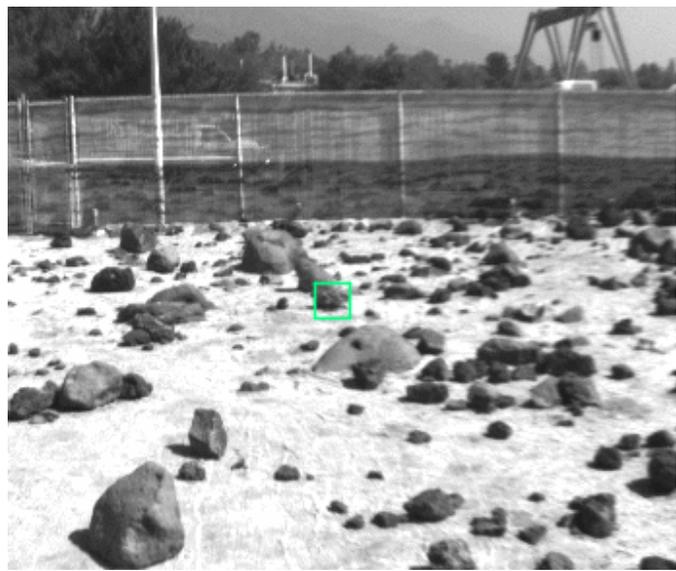
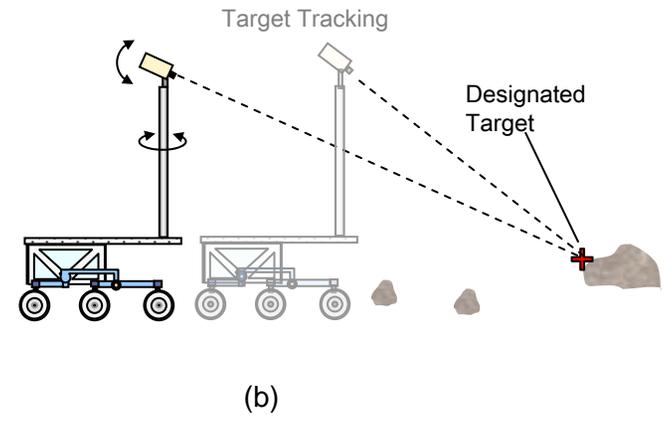
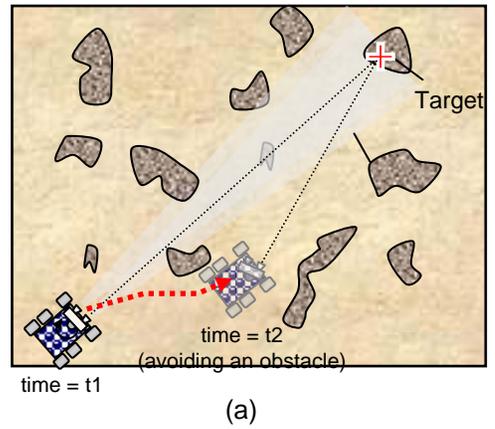
- I/O, motion control, QuickTime™ and a decompressor are needed to see this picture.
- Trajectory Generation
- Rough Terrain Locomotion
- Odometry Pose Estimation
- Stereo Processing
- Visual Odometry
- Navigation (Morphin)
 - Obstacle avoidance
 - Path Planning



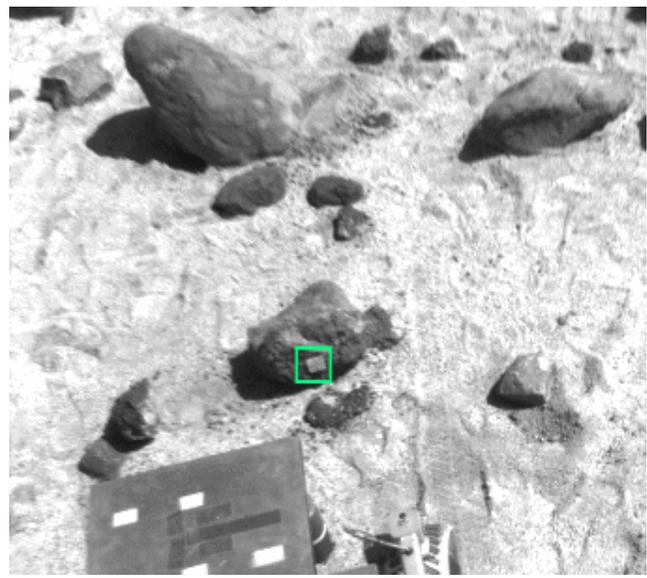
Application Domains

- Navigation
- Single-cycle Instrument Placement

Single-cycle Instrument Placement



1st Frame



Changes in FOV

37th Frame after 10 m



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

QuickTime™ and a Video decompressor are needed to see this picture.



Integrated Single-Cycle Instrument Placement



QuickTime™ and a
decompressor
are needed to see this picture.



Approaches



- There are several approaches to improving software reliability. These include:
 - Improved processes for software development
 - Static code analysis and validation tools
 - Increased software reliability through reuse
 - Formal technology validation
 - Nightly regression testing
 - Fault-tolerant software and redundant computing

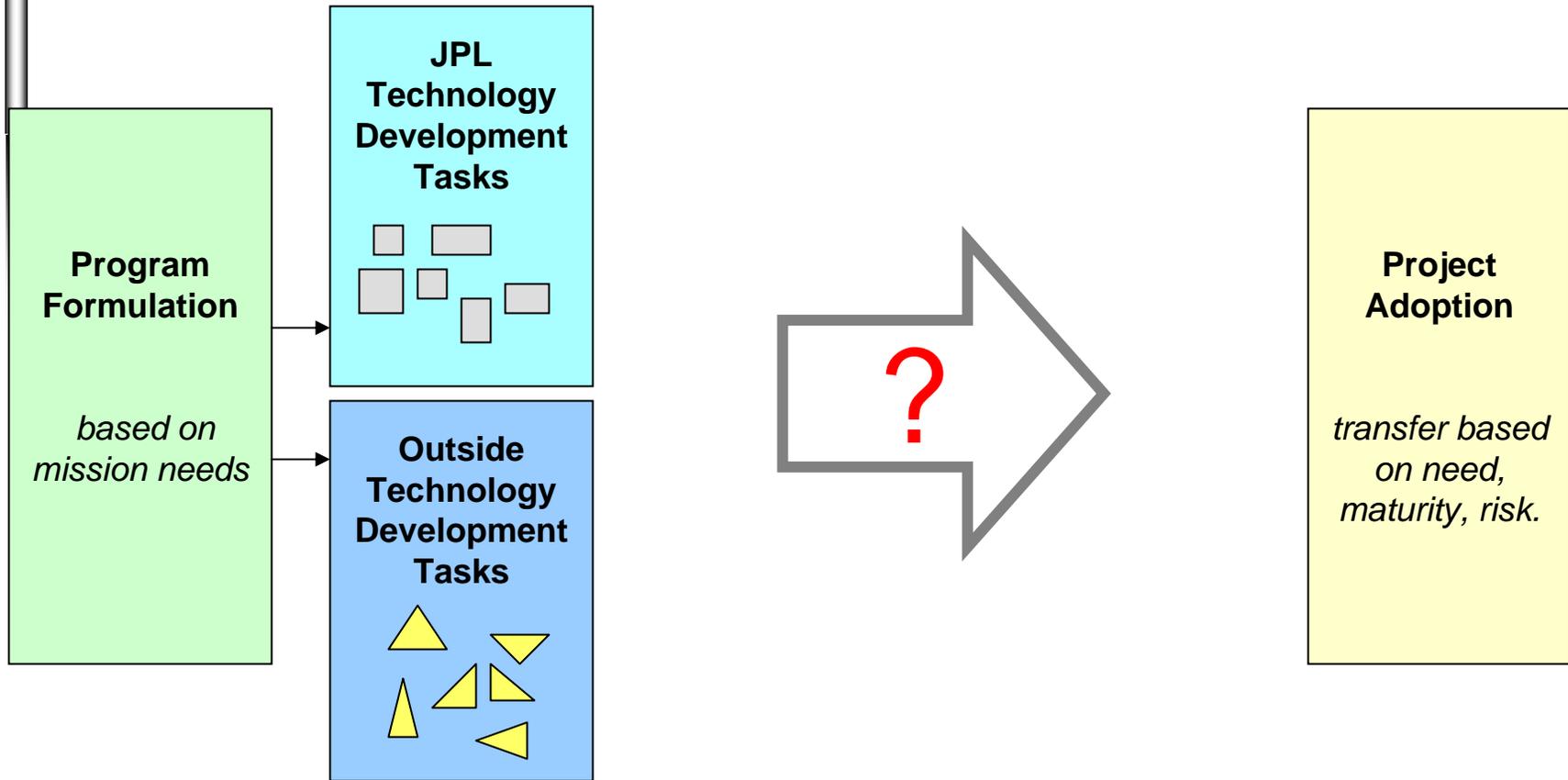


Technology Development and Infusion into Flight





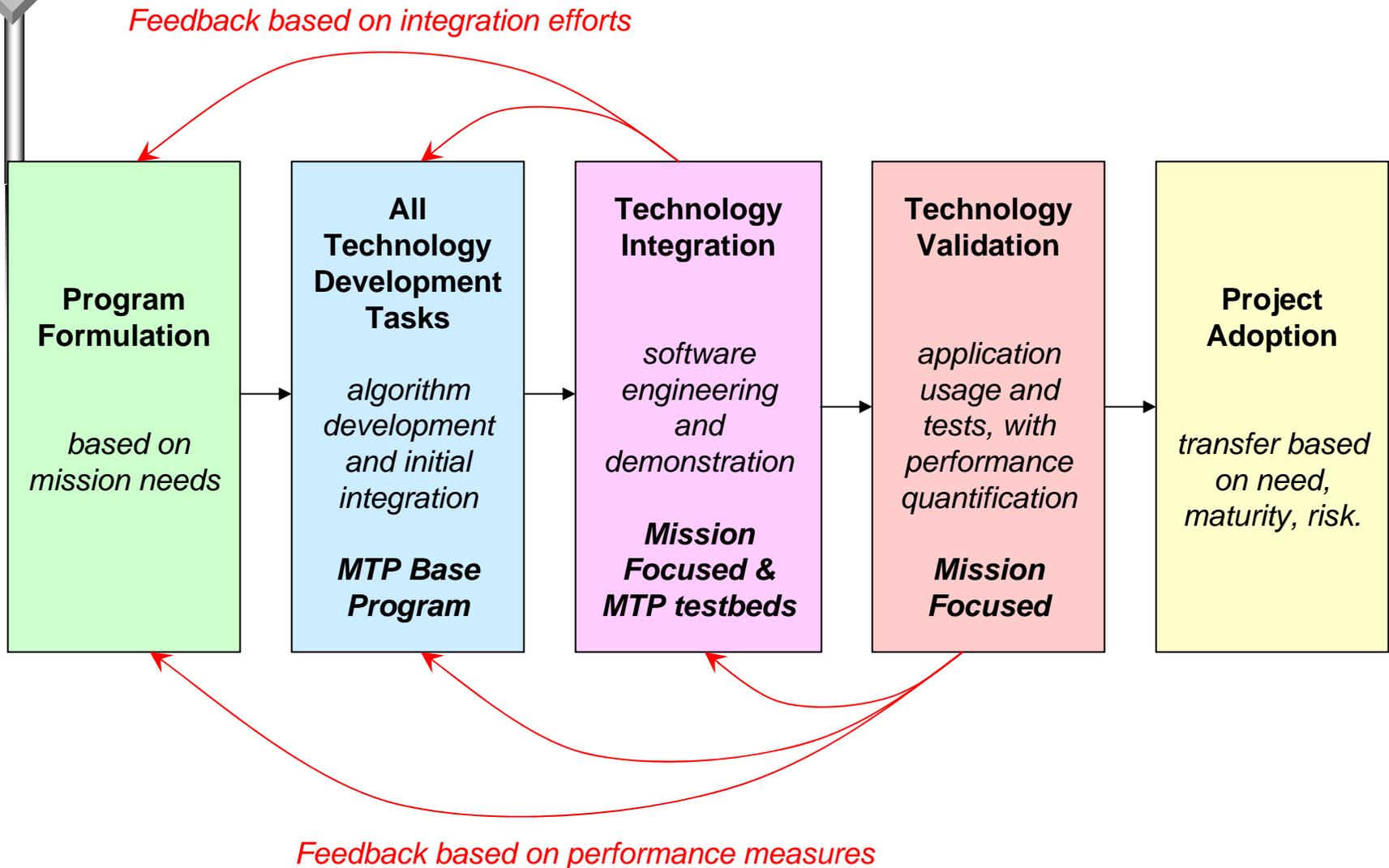
Technology Infusion Process – Historically



*Algorithm development
and initial integration*

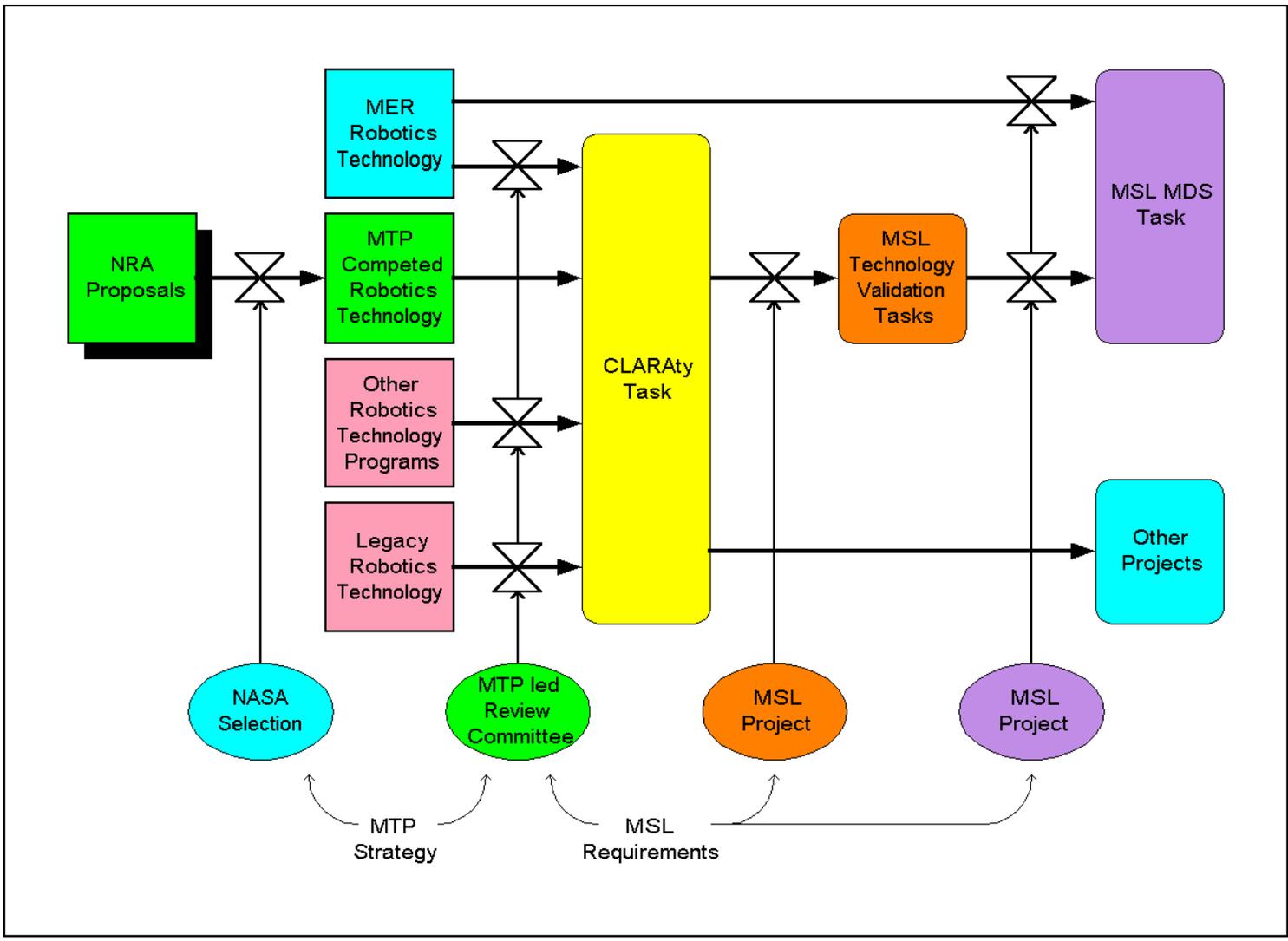


Technology Infusion Process – MTP





Technology Component Flow





Flight Software Processes and Tools

- List coding conventions, rules, and guidelines.
- Avoid technologies that have not been thoroughly tested by previous missions or by the research community.
- Hold formal design reviews
 - Review designs before and after implementation
 - Review interfaces, implementations, test plans, commands, and telemetry for each software component
- Use code buddy reviews
 - Have someone other than the developer statically review the code and look for potential problems or violations of the coding conventions.
 - Use automated tools for source code analysis to highlight suspicious code segments. For example, MER used Code Wizard and Cleanscape on early versions of the flight software and Coverity on most recent versions.
 - Review by internal and external teams. Use validation and verification group expertise.



Rigorous Flight Software Testing



- **Unit Testing:**
 - Extensive testing of each module in isolation by the developer
- **Regression Testing:**
 - Integrated module testing by a dedicated test team after new modules are integrated.
- **System Testing:**
 - Project wide rehearsals of expected mission scenarios
 - Can last several days where several different activities would be tested in the manner they would be used in the mission.
 - All communication is done during communication passes.



Integrating new Capabilities in Flight Software (MER Example)



- Train operations personnel for the use of the new capabilities.
- Load new the version of software into a different EEPROM bank while maintain the current version on another bank. In the event of an unintended reboot while starting the new version, the older version gets automatically used.
- Checkout new capabilities in a gradual manner.



- Assignment of one "owner" developer per software module
- Object-oriented style design, with emphasis placed on interfaces, encapsulation, and modularity
- Objects implemented as hierarchical state machines
- Asynchronous message passing as the principle means of communication between objects
- Severe limitations on use of dynamic memory allocation to avoid heap fragmentation
- Extensive use of diagnostics embedded throughout the software, including many design-by-contract assertions
- Reference:
 - Glenn E. Reeves & Joseph F. Snyder "A Overview of the Mars Exploration Rovers' Flight Software" 2005 IEEE International Conference on Systems, Man and Cybernetics Waikoloa, Hawaii, October 10-12, 2005

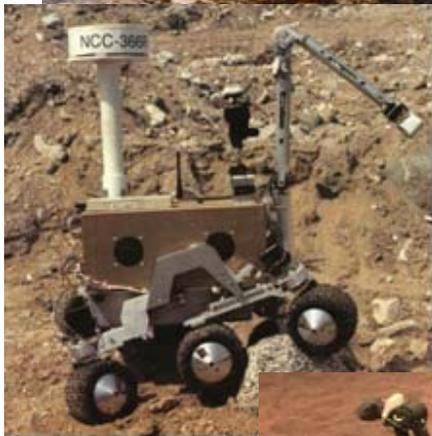
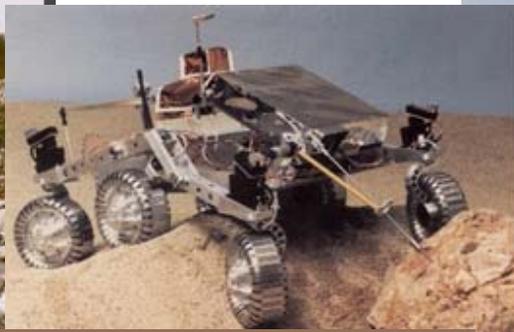


Historical Antecedents and Motivation





Some JPL Robots





Historical Antecedents



- **Late 80's - Early 90's:** parallel robotic developments
 - RSI, MOTES, Satellite Servicing, Robby, Mircover
 - No shared *hardware* or *software*
- **Mid 90's:** Mars rover research centralized with Rocky 7
 - First flight rover
- **Late 90's:** Expansion and diversification of rover work
 - No software interoperability (Rocky 7, FIDO, Athena, DARPA)
 - Autonomy demonstration of Remote Agent Experiment (ARC and JPL)
 - MDS investigates reusable software for spacecraft control.
- **'99-Early 00:** Exploration Technology Program develops concept for a unifying autonomy architecture
 - Unifying autonomy and robotic control
 - Started the CLARAty task
- **Today:**
 - Unification of several robotic developments at JPL, ARC, and CMU
 - Two flight rovers with several new robotic capabilities



Problem and Approach



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

- Objectives
 - Unify robotic infrastructure and framework
 - Capture and integrate legacy algorithms
 - Simplify integration of new technology
 - Operate heterogeneous robots



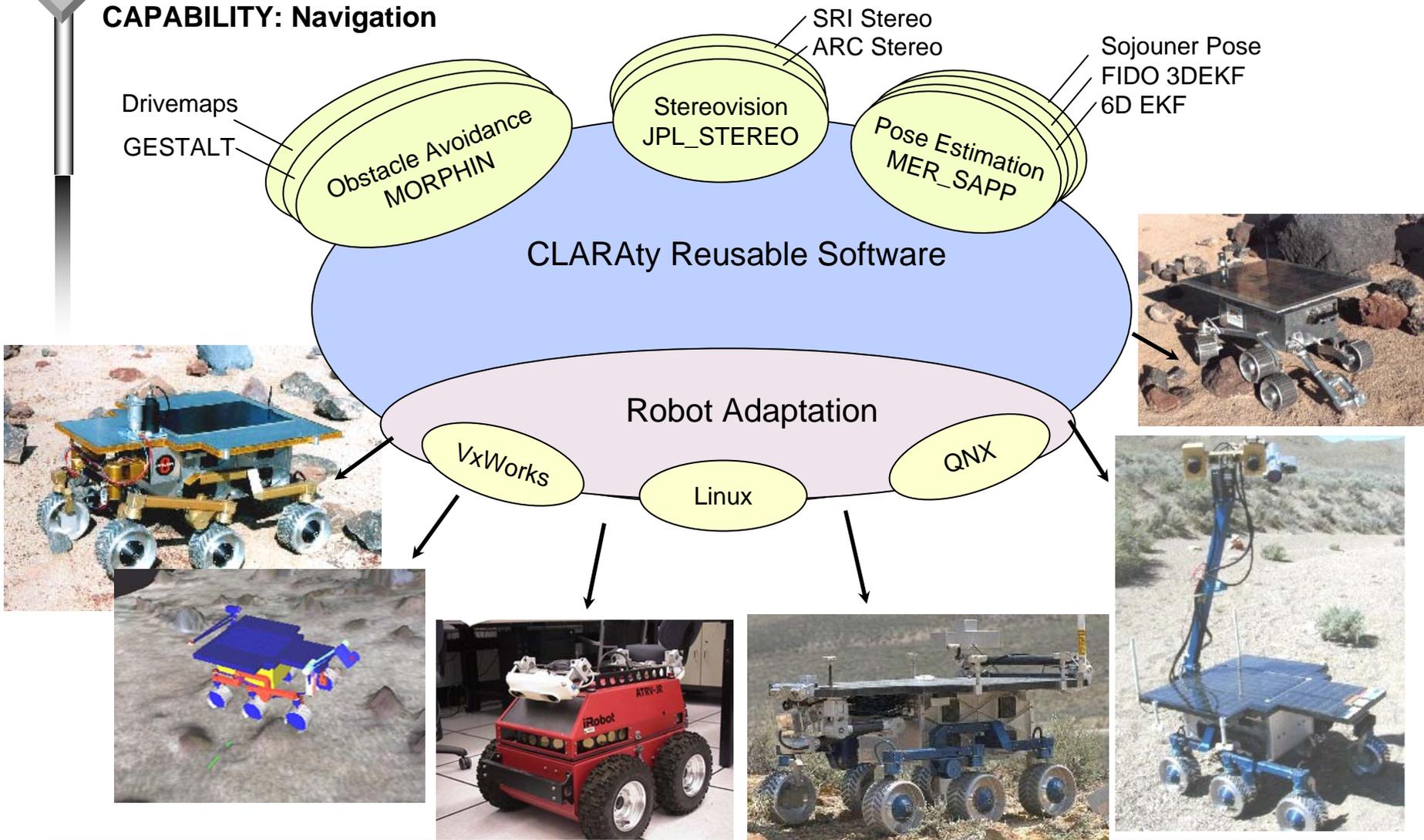
What is CLARAty?



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

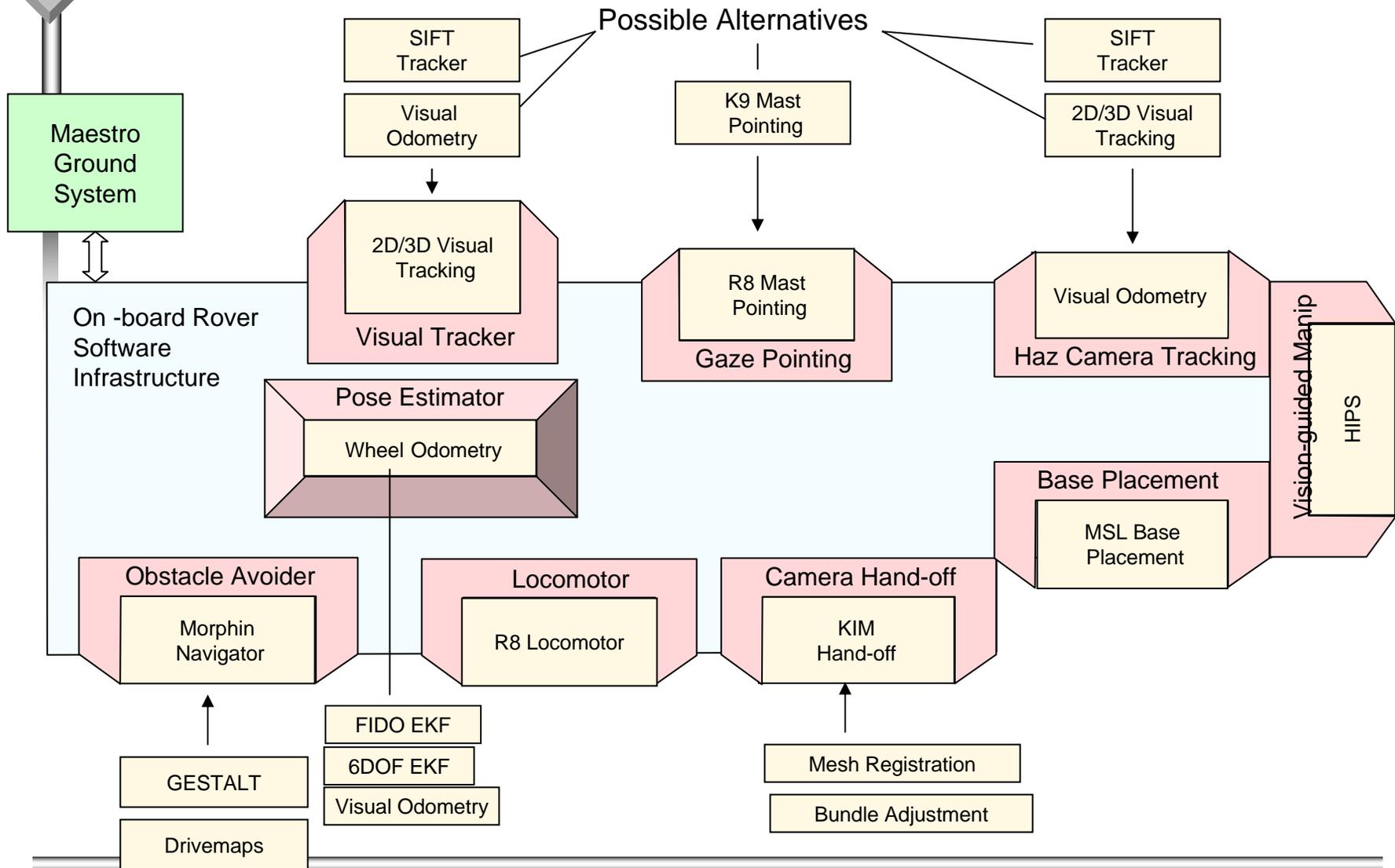
*A tool for technology development
and maturation*

CAPABILITY: Navigation





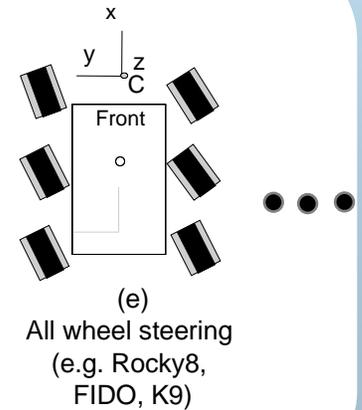
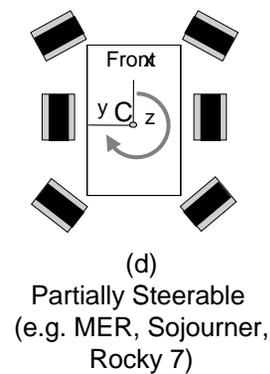
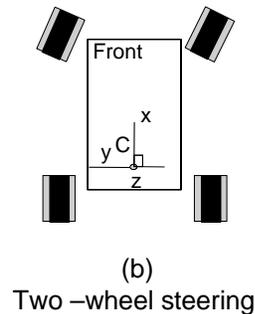
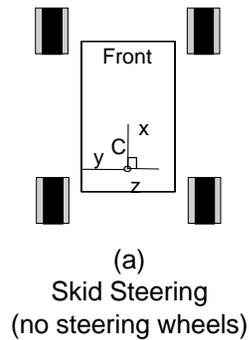
CLARAty Framework for Single-cycle Instrument Placement





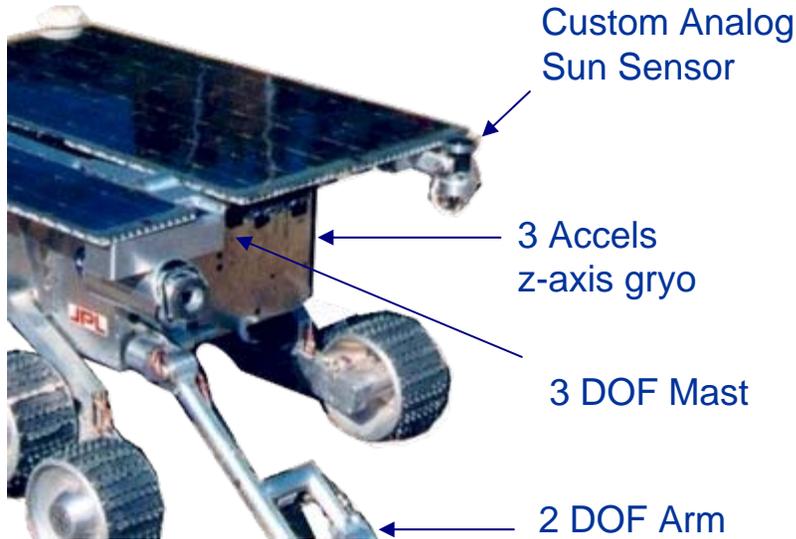
Challenges in Reuse

- Mechanisms and Sensors
- Hardware Architecture
- Software Algorithms



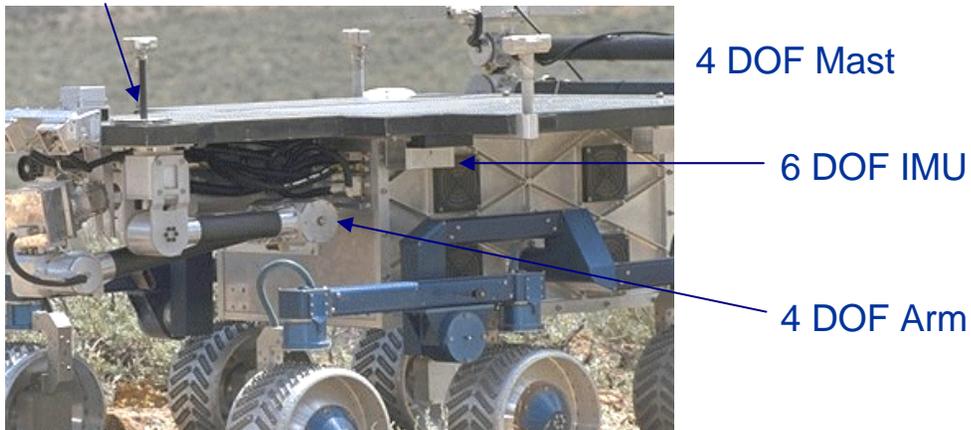


Different Sensors and Appendages



QuickTime™ and a
None decompressor
are needed to see this picture.

Camera Sun Sensor





Challenges in Reuse

- Mechanisms and Sensors
- Hardware Architecture
- Software Algorithms

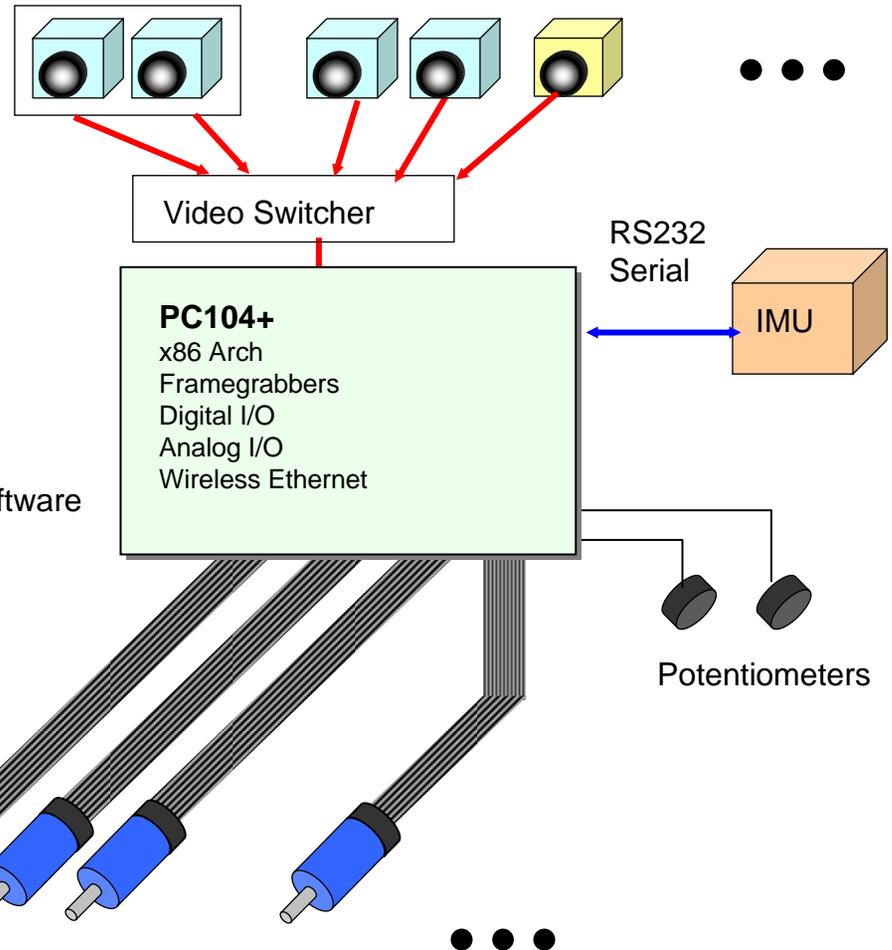


Centralized Hardware Architecture



FIDO

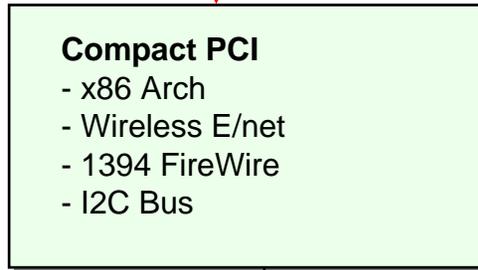
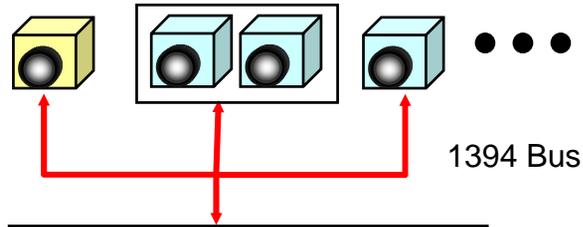
PID Control in Software



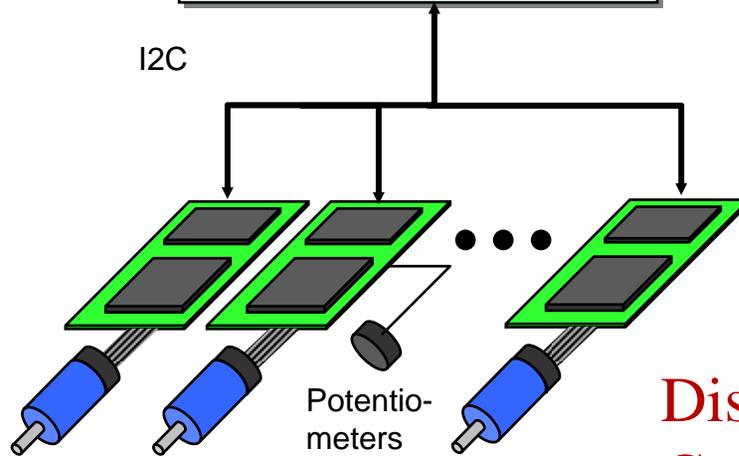


Rocky 8

Sun Sensor



I2C



Actuator/Encoders

Rocky Widgets

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

Distributed Motion Control and Vision



Challenges in Interoperability

- Mechanisms and Sensors
- Hardware Architecture
- Software Algorithms



The *new* algorithms to be integrated may:

- Have architectural mismatches with the framework
- Include multiple orthogonal functionalities
- Make implicit assumptions about the platform
- Duplicate functionality in the framework
- Use incompatible data structures
- Are complex and hard to tune
- Require highly specialized domain expertise
- Are poorly implemented

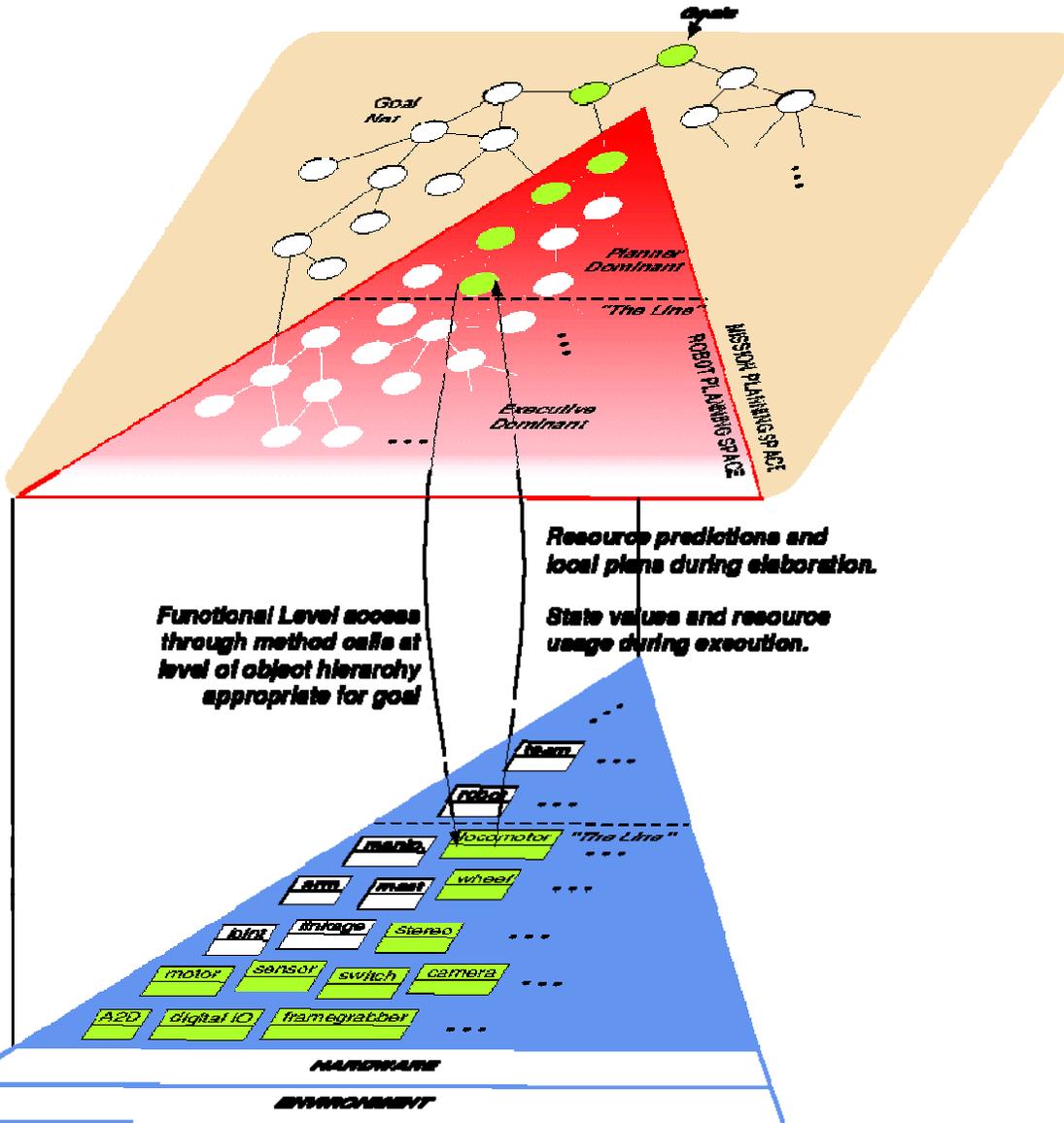


Architecture and Process



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

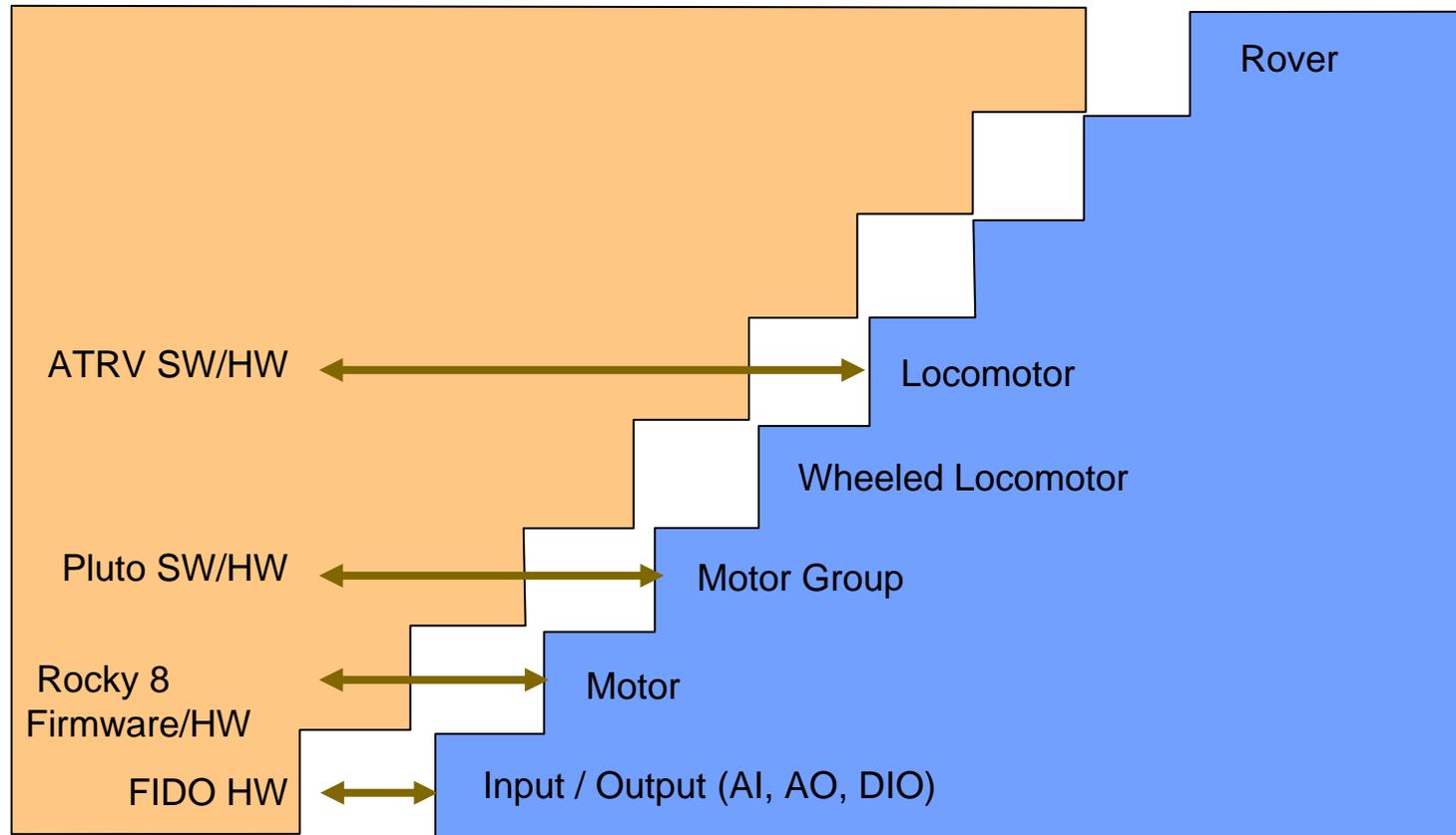


Multi-level Abstraction Model



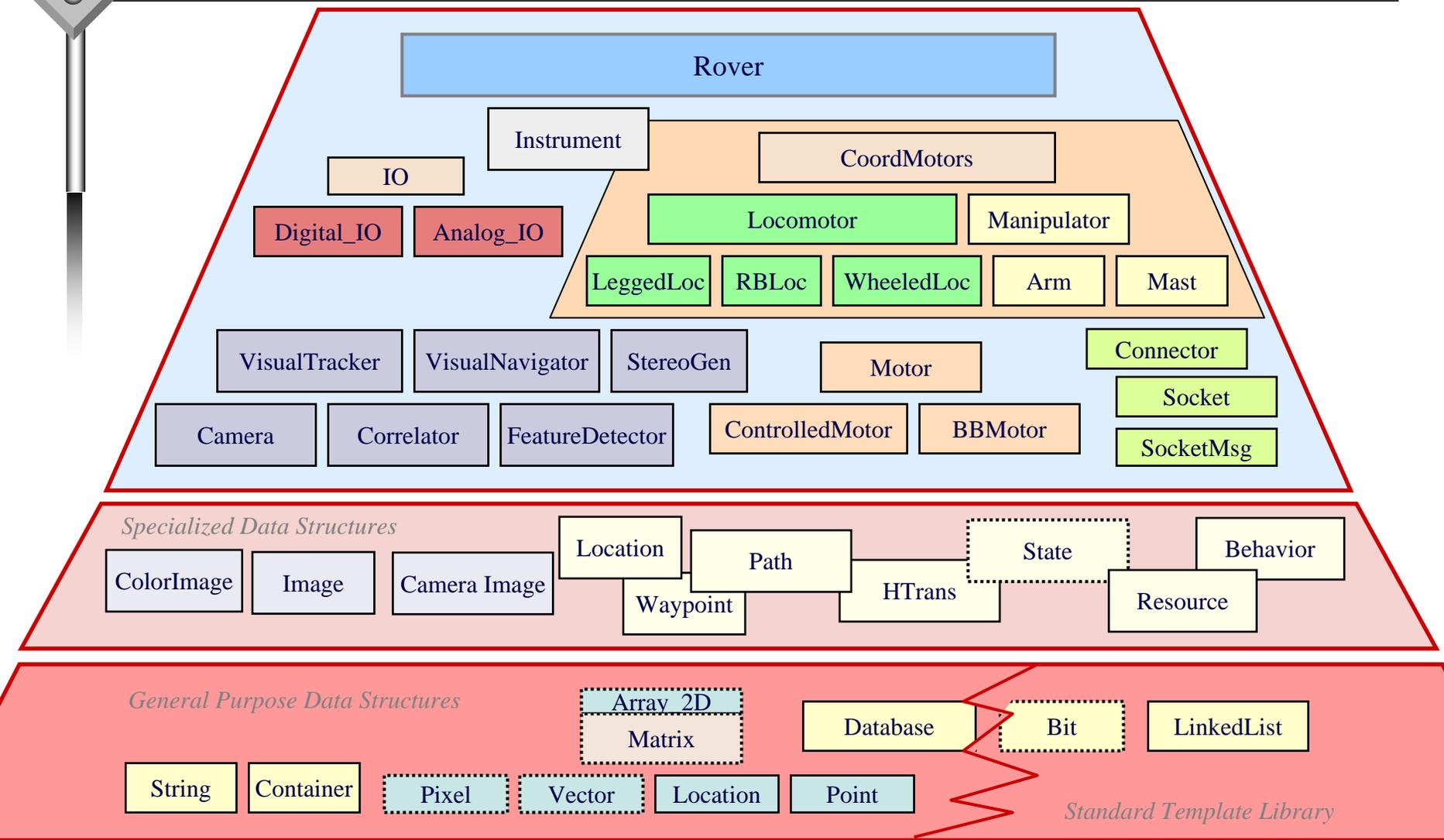
Use abstractions
Interface at different levels

Example of
Levels of Abstraction
for locomotors



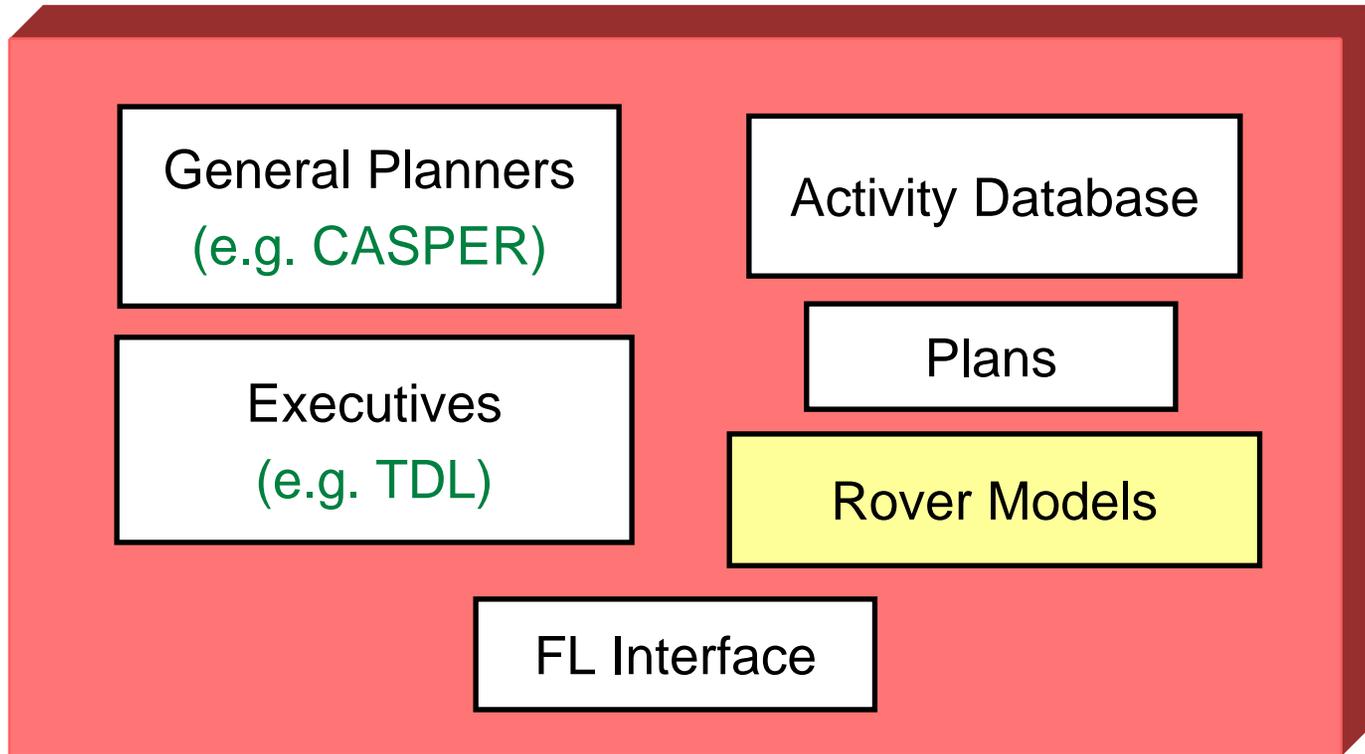


Functional Layer Components



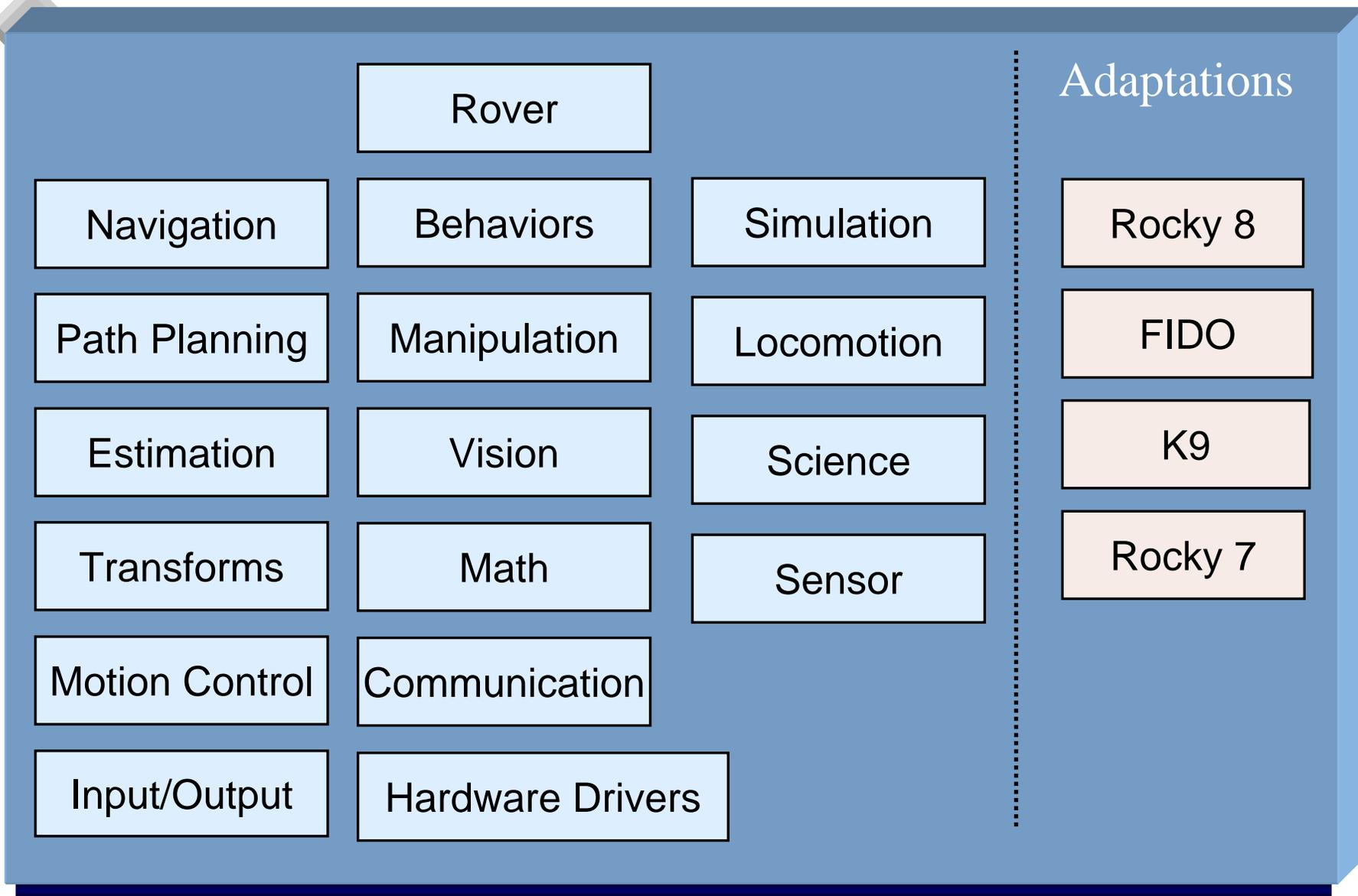


The Decision Layer





The Functional Layer

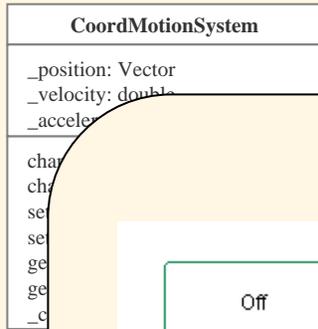




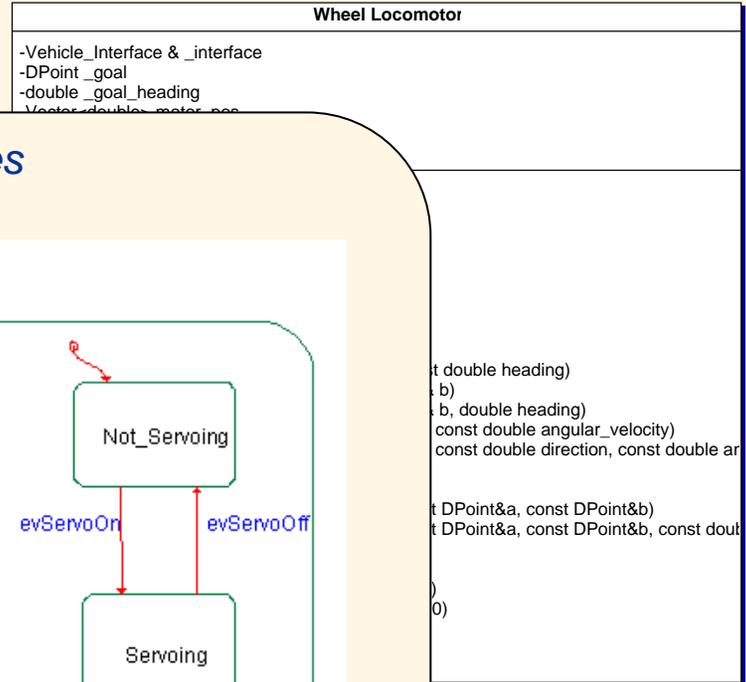
Standardizing Base Abstractions



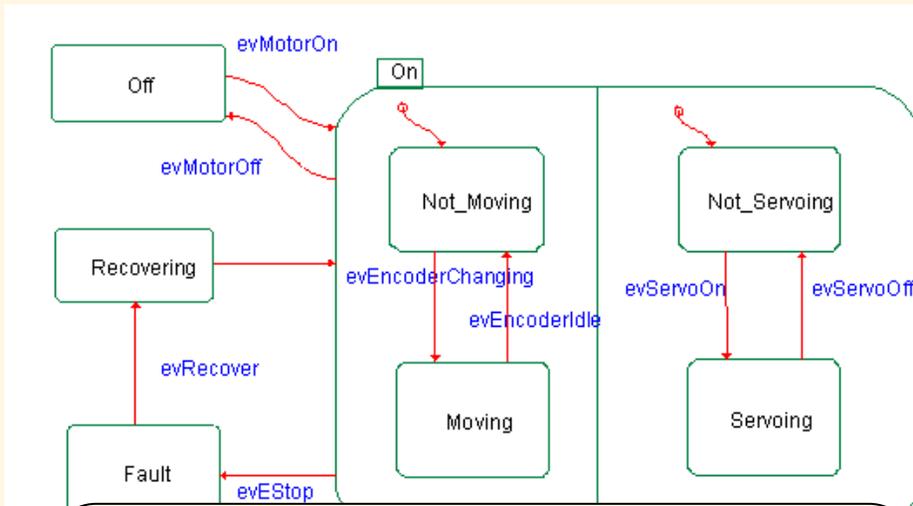
Abstractions



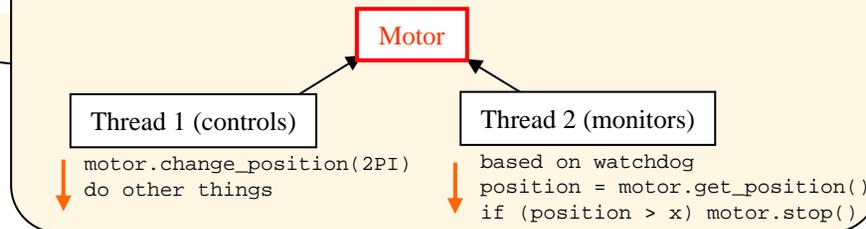
APIs and Behaviors



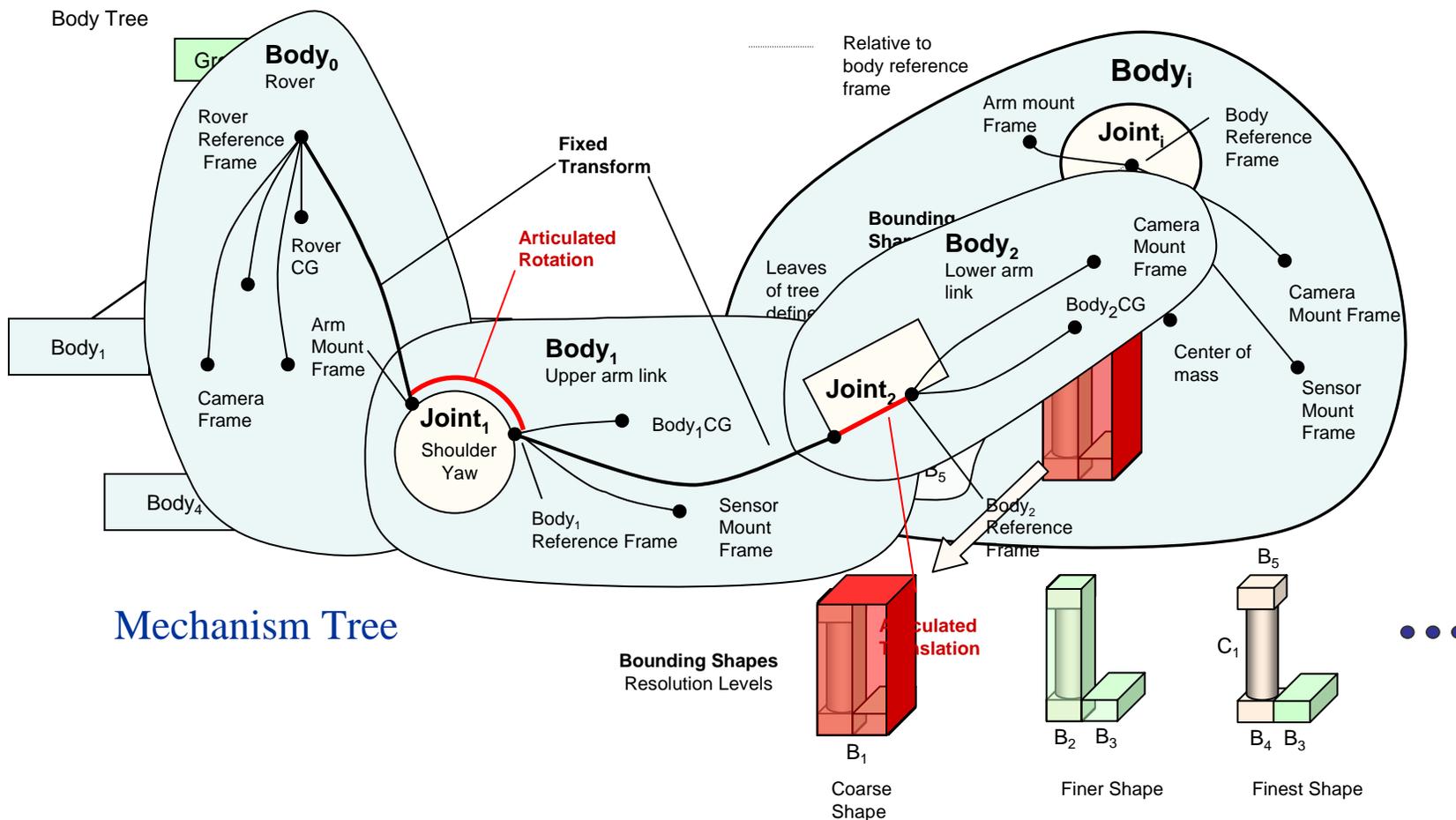
State Machines



Runtime Models

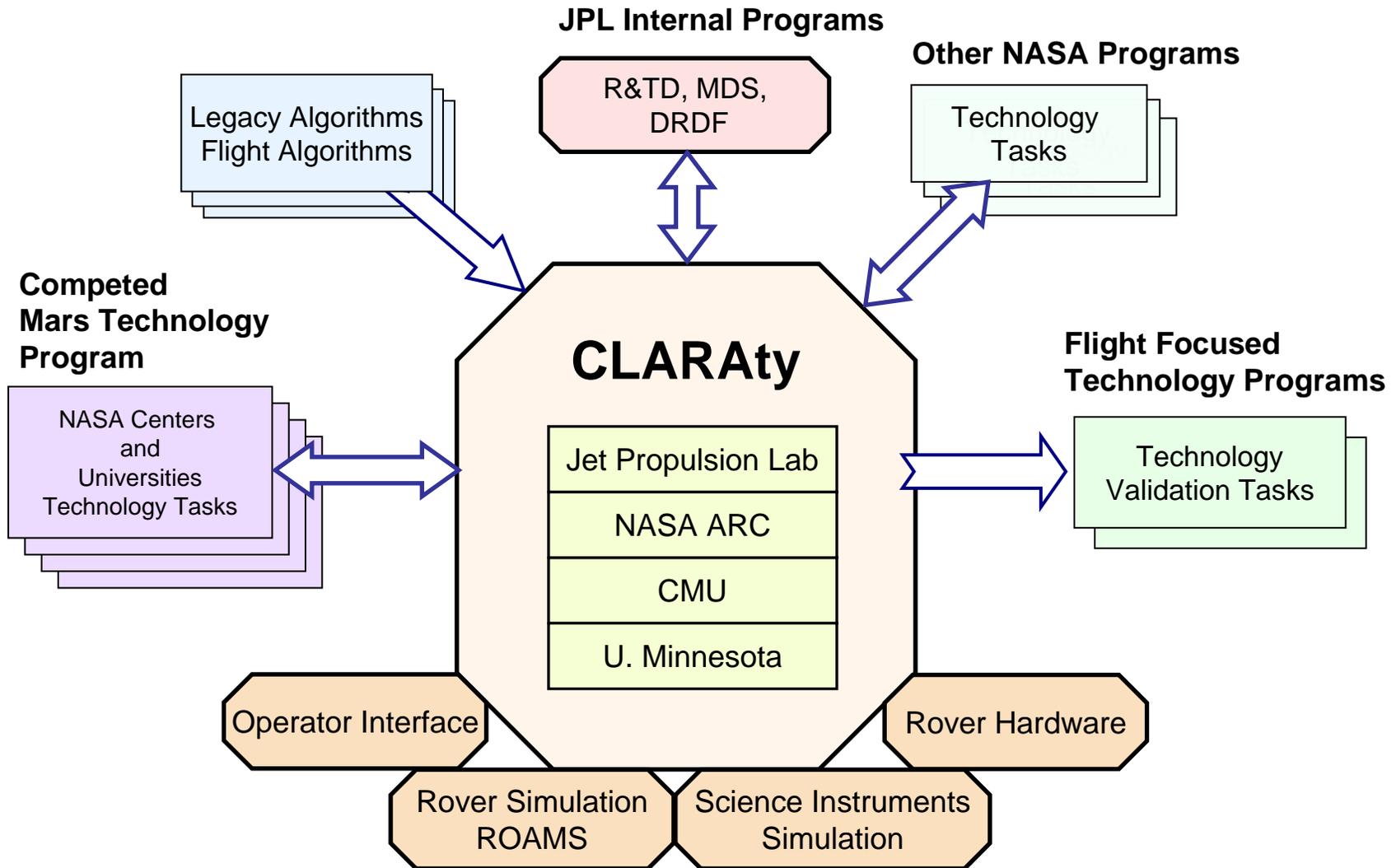


Bodies and Joints





Collaborations & Interactions



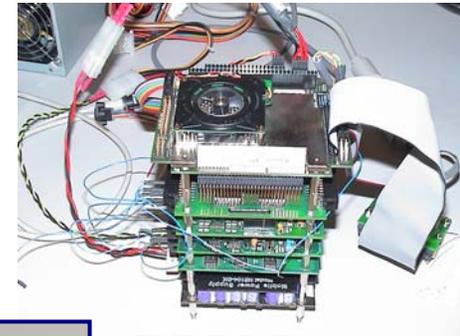


Unit and Regression Testing





CLARAty Test Bed for Regression Testing



FIDO2 Stack



ATRV Jr.



Dexter Manipulator
Bench top



Rocky 8 PPC Bench top



Challenges Ahead



- Mature framework and robotic capabilities
- Investigate relevance to flight and define migration path
- Develop regression tests for long-term maintainability of robotic capabilities - *very hard and open research topic*
- Maintain current capabilities on existing platforms
- Develop new capabilities (e.g. continuous motion)
- Integrate new technologies from competed programs
- Develop a releasable version
- Develop formal documentation and tutorials
- Identify and deploy on low-cost rover platform
- Open source



Summary



- Developed a unified and reusable software framework
- Deployed at multiple institutions
- Deployed on multiple heterogeneous robots
- Integrated multiple technologies from different institutions
- Delivered algorithms for formal validation
- Enabled new technology developments on multiple platforms
- Integrated flight algorithms for detailed performance characterization and operation on research rovers.
- Taking a technology from inception, to development in CLARAty, to validation, and now to integration into flight



Current CLARAty Core Team



- **NASA Ames Research Center**

- Clay Kunz
- Eric Park
- Susan Lee

- **Carnegie Mellon University**

- David Apelfaum
- Nick Melchior
- Reid Simmons

- **University of Minnesota**

- Stergios Roumeliotis

- **Jet Propulsion Laboratory**

- Antonio Diaz Calderon
- Tara Estlin
- John Guineau
- Won Soo Kim
- Richard Madison
- Michael McHenry
- Mihail Pivtoraiko
- Issa A.D. Nesnas
- Babak Sapir
- I-hsiang Shu

- **OphirTech**

- Hari Das Nayar

Full Credits for all Developers and Contributors at:
<http://keuka.jpl.nasa.gov/main/project/team/index.shtml>

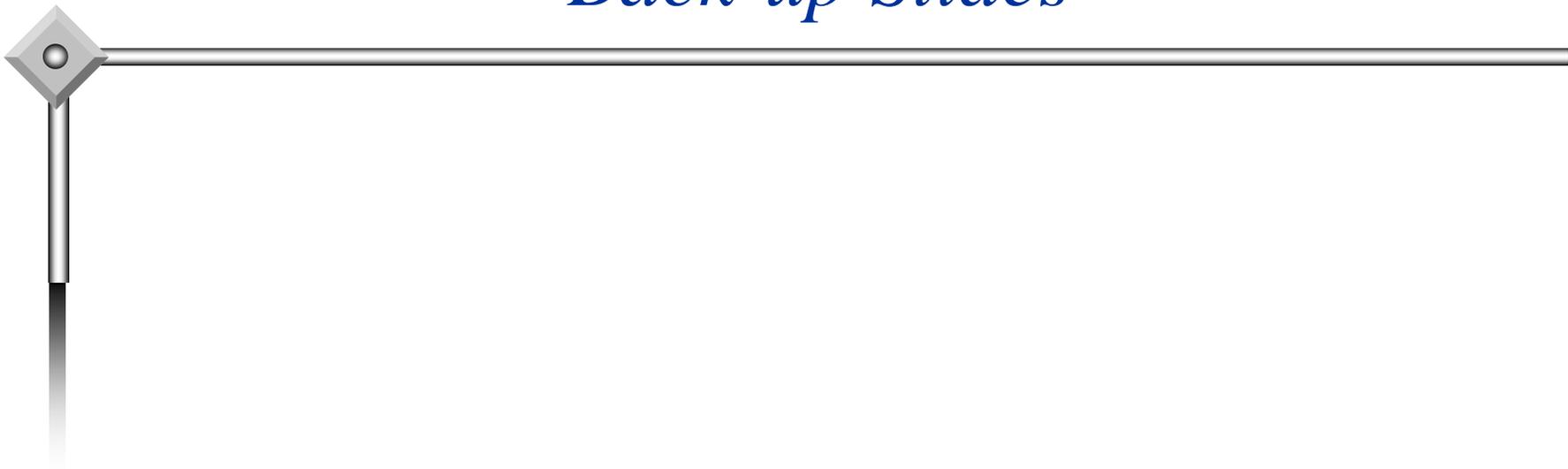


Questions?





Back-up Slides





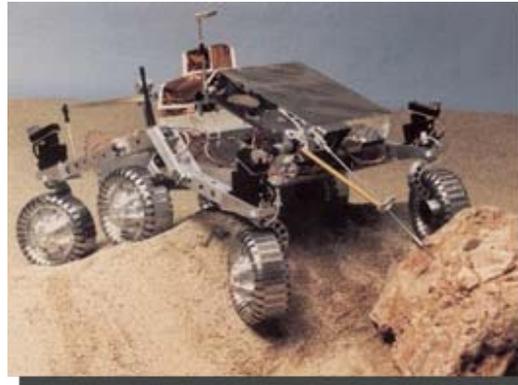
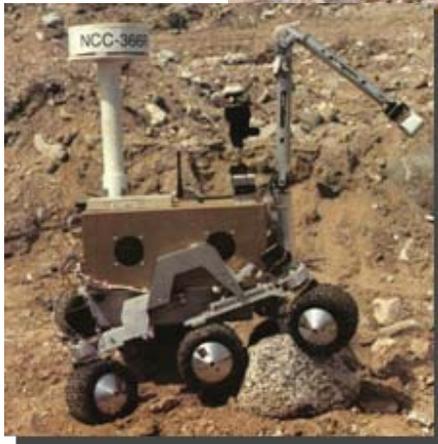
NASA/JPL Develops Various Rovers



Large



Medium



Small



For research & flight



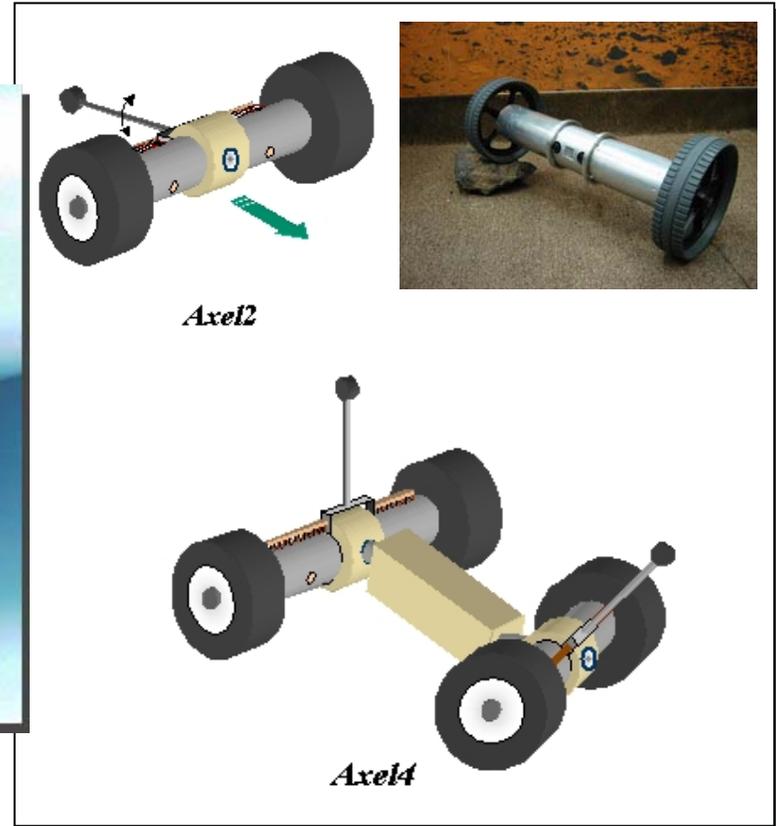
Would like to support ...



Custom Rovers



Manipulators



Reconfigurable Robots



COTS Systems

with different sensors

From wheeled Rocker-bogies with different steering

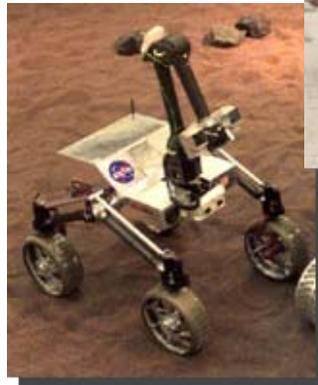
To wheels on articulated links

To inflatable wheels

From three wheelers

To four, six and even eight

From wheeled to legged





For Example: Wheeled Locomotion



Rocky 7

QuickTime™ and a None decompressor are needed to see this picture.

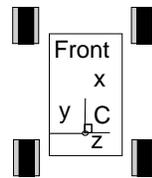


Rocky 8

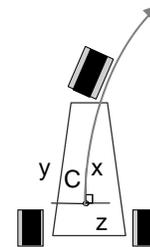
QuickTime™ and a Video decompressor are needed to see this picture.

QuickTime™ and a None decompressor are needed to see this picture.

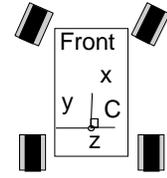
General flat terrain algorithms and specialized full DOF algorithms



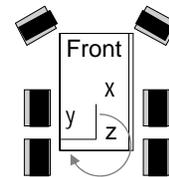
(a) Skid Steering
(no steering wheels)



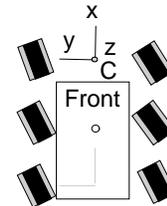
(b) Tricycle
(one steering wheel)



(c) Two-wheel steering



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

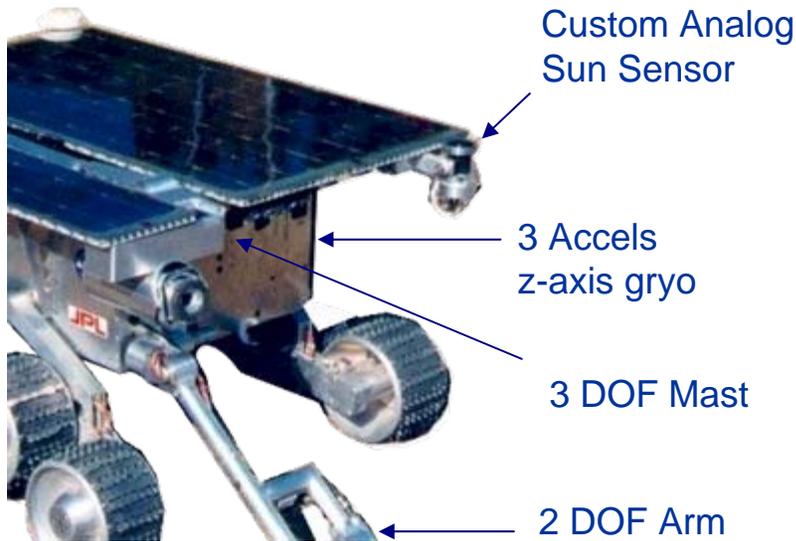


(e) All wheel steering
(e.g. MER, Rocky8, Fido, K9)



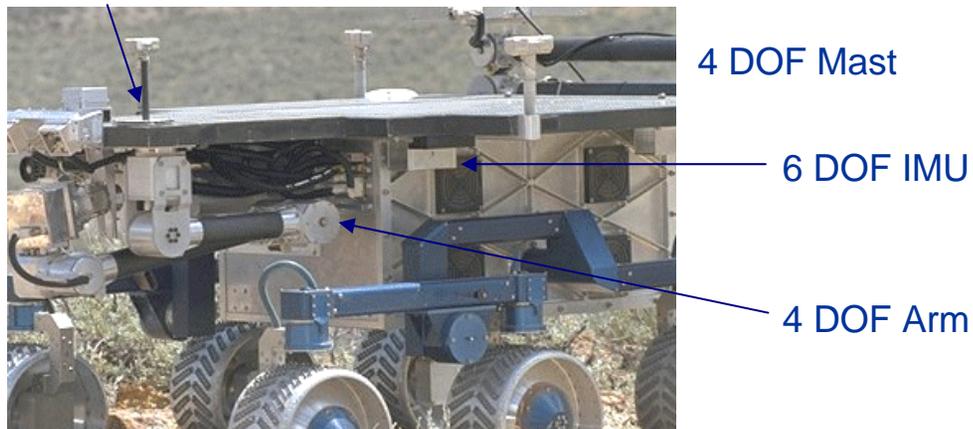
(f) Steerable Axle
(e.g. Hyperion)





QuickTime™ and a
None decompressor
are needed to see this picture.

Camera Sun Sensor



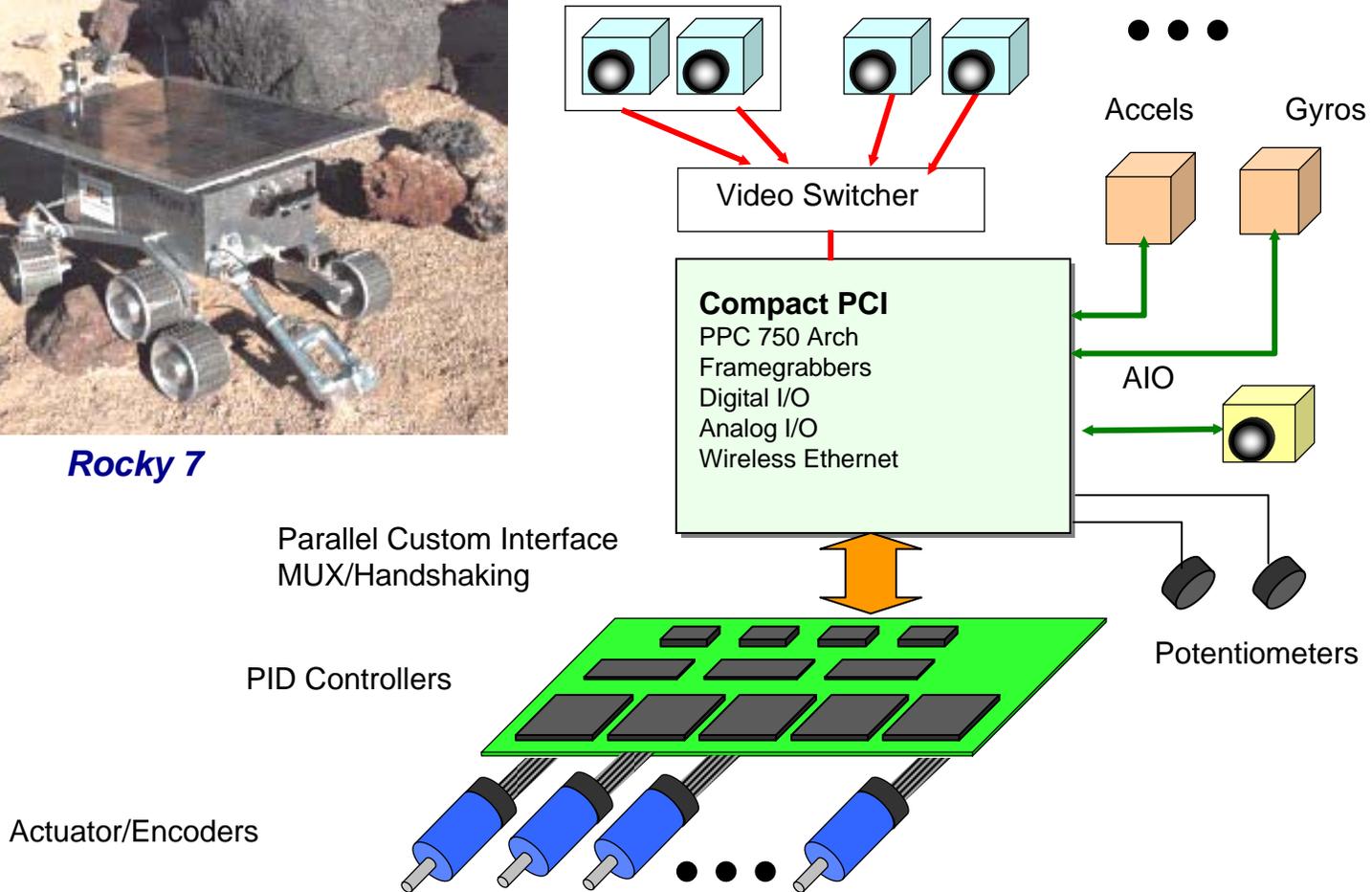
- Given different capabilities, how much reuse can be achieved?



Semi-centralized Hardware Architecture



Rocky 7





One Approach



Use the best attributes from each system
and build a common platform

Unfortunately this is not always possible

OR

Develop a model to deal with the variability



One Approach



- Develop
 - Common data structures
 - Physical & Functional Abstractions
 - E.g. motor, camera, locomotor. Stereo processor, visual tracker
 - Unified models for the mechanism
- Putting it together
 - Start with top level goals
 - Elaborate to fine sub-goals
 - Choose the appropriate level to stop elaboration
 - Interface with abstractions
 - Abstractions translate goals to action
 - Specialize abstractions to talk to hardware
 - Hardware controls the systems and provide feedback

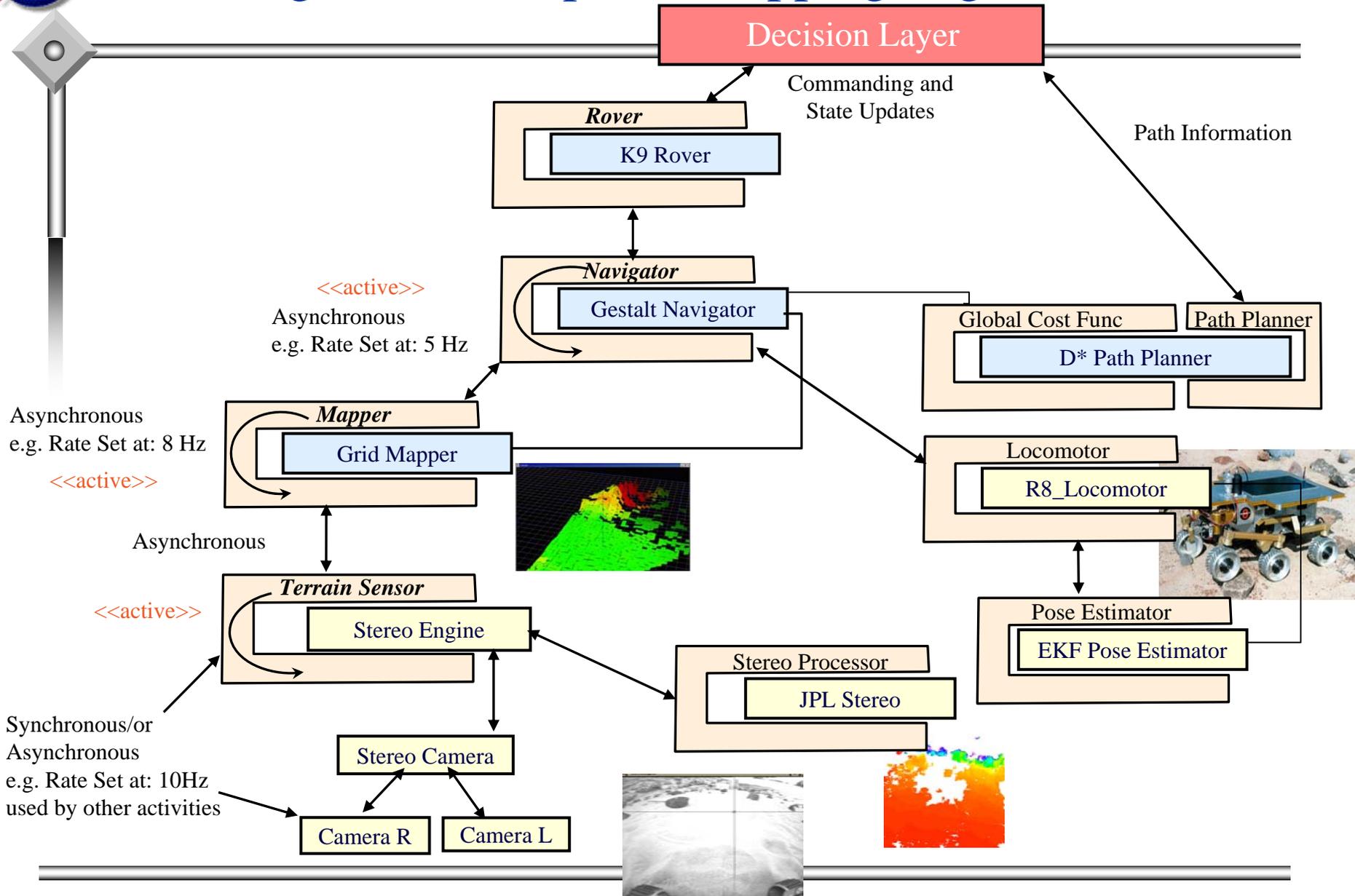


Putting it All Together





Navigation Example - Swapping Algorithms





Complex Algorithms on different Platforms

- I/O, motion control, QuickTime™ and a decompressor are needed to see this picture.
- Trajectory Generation
- Rough Terrain Locomotion
- Odometry Pose Estimation
- Stereo Processing
- Visual Odometry
- Navigation (Morphin)
 - Obstacle avoidance
 - Path Planning



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

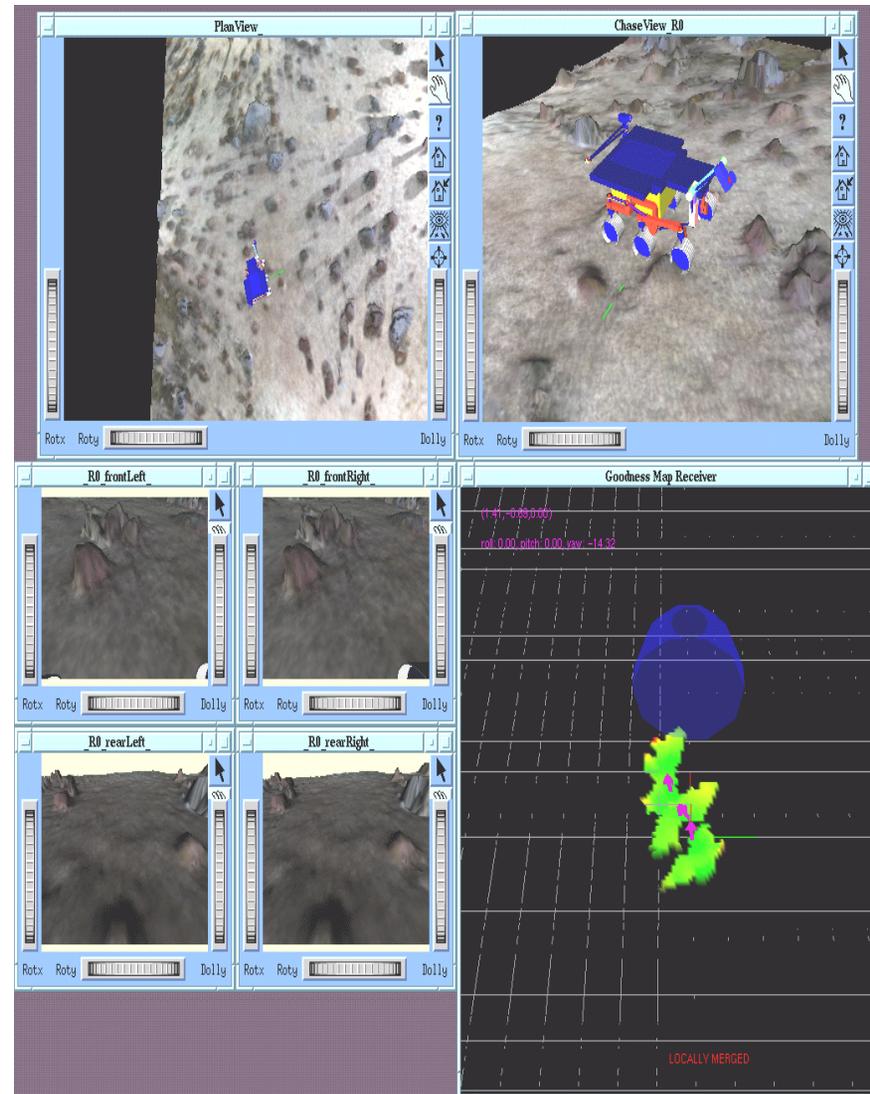
QuickTime™ and a
Video decompressor
are needed to see this picture.



And with a Simulated Rover



QuickTime™ and a
Video decompressor
are needed to see this picture.





Acknowledgements

CLARAty Team (multi-center)



Jet Propulsion Laboratory

- ROAMS/Darts Team
- CLEaR Team
- Instrument Simulation Team
- Machine Vision Group
- Robotic Systems Group



Ames Research Center

- K9 Team

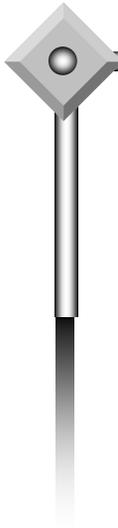


Carnegie Mellon University

University of Minnesota



Thank you for your Attention

A decorative corner element on the left side of the slide, consisting of a vertical line with a diamond-shaped cap at the top and a horizontal line extending to the right.

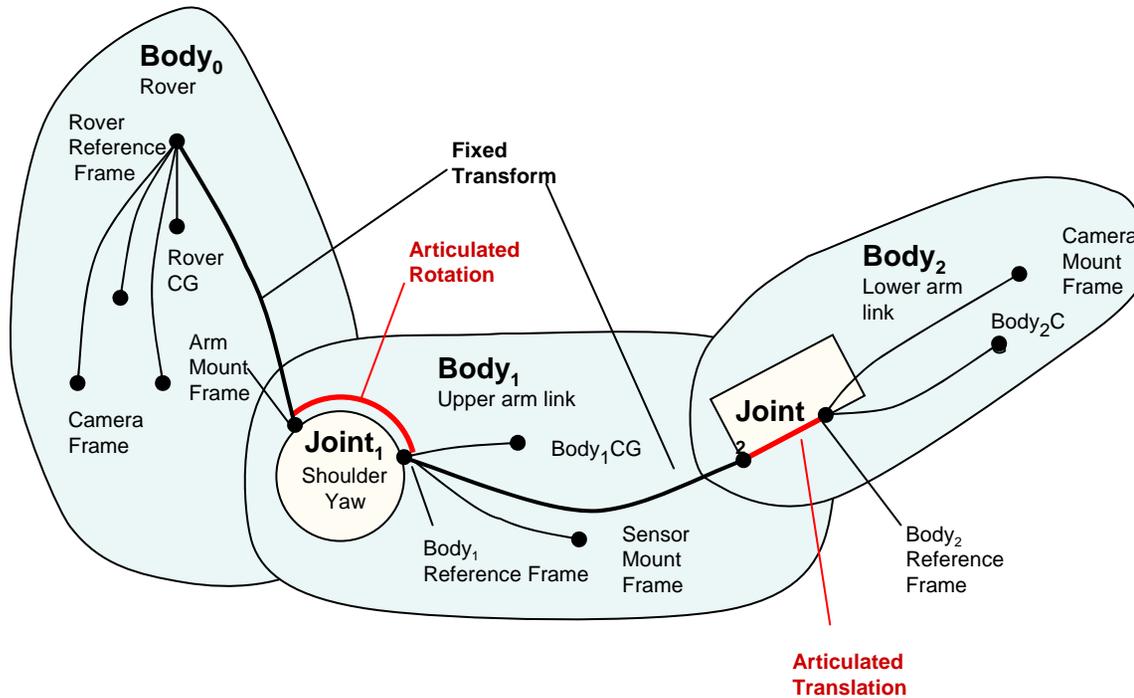


Backup Slides





Connecting Bodies and Joints





Some Results on Reusability





Some Software Inter-operability Statistics



Algorithm Description	1st Adaptation		Subsequent Adaptations (time in days)		
	On	Time	On	Adapt.Time	Tuning Time
3D Extended Kalman Filter for Rover Pose Estimation	Rocky 8	60	FIDO (JPL)	1	3
3D Locomotion for wheeled vehicles	Rocky 7	30	Rocky 8 (JPL)	10	12
			FIDO (JPL)	1	1
			K9 (ARC)	4	1
Morphin Navigator	Rocky 8	75	FIDO (JPL)	1	1
			K9 (ARC)	4	14
			ROAMS (JPL)	1	-
Mast Control Software	Dexter	21	Rocky 8 (JPL)	7	7
			FIDO (JPL)	4	1



Code Reusability for Motion Control



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

Rocky 8 Modules	Lines of Code	Status
Controlled Motor	2,652	Reusable
Trajectory Generator	691	Reusable
Input Output	2,690	Reusable
Bits	1,580	Reusable
Resources (Timers, etc.)	725	Reusable
Bits	756	Reusable
Rocky 8 Motor	1,180	Non-reusable
Rocky 8 H/W Maps	626	Non-reusable
Widget Board Software	2,126	Reusable - widget
Motor Controller HCTL	900	Reusable - HCTL
I2C Master	1,165	Reusable - I2C
I2C Master Tracii	1,223	Reusable - Tracii
Total	16,314	
Total Reusable	89%	
Total Reusable - Strict	56%	

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



Code Resuability for Locomotion Example



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



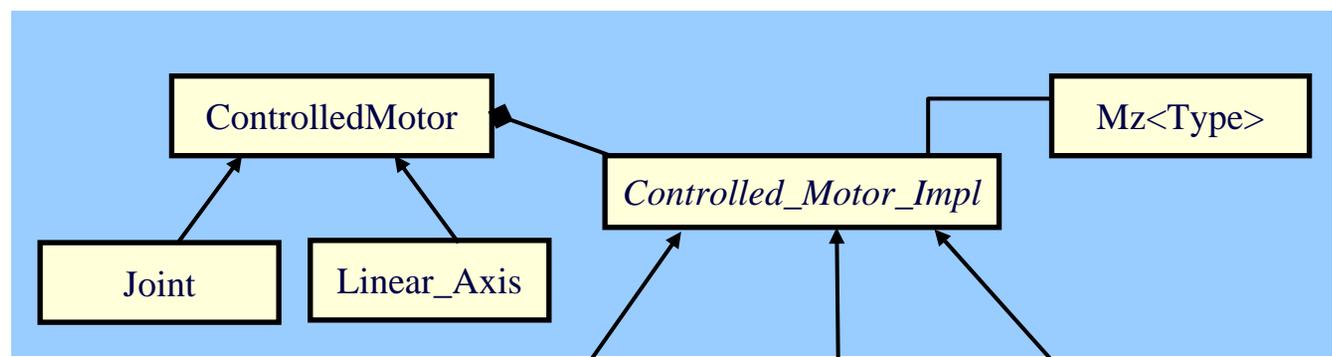
Conclusions



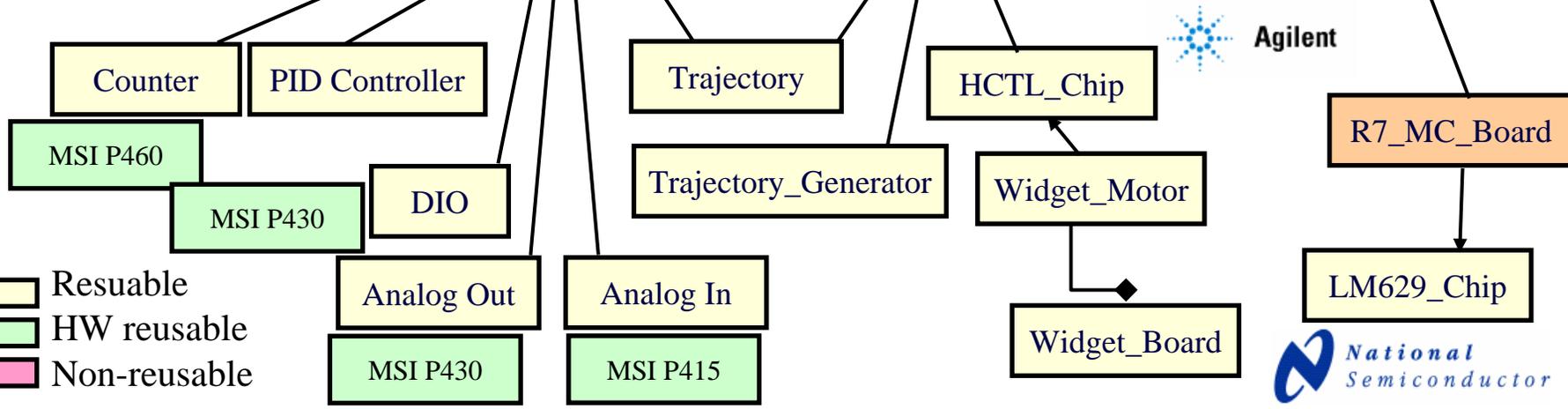
- Use abstraction to master complexity
- Encapsulate and abstract hardware variations
- Provide multi-level access through Decision Layer for fault diagnosis and recovery
- Use domain expertise to guide design
- Make all assumptions explicit
- Stabilize external interfaces rapidly
- Document processes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



Examples of CLARAty Reusability



Non-Reusable Layer

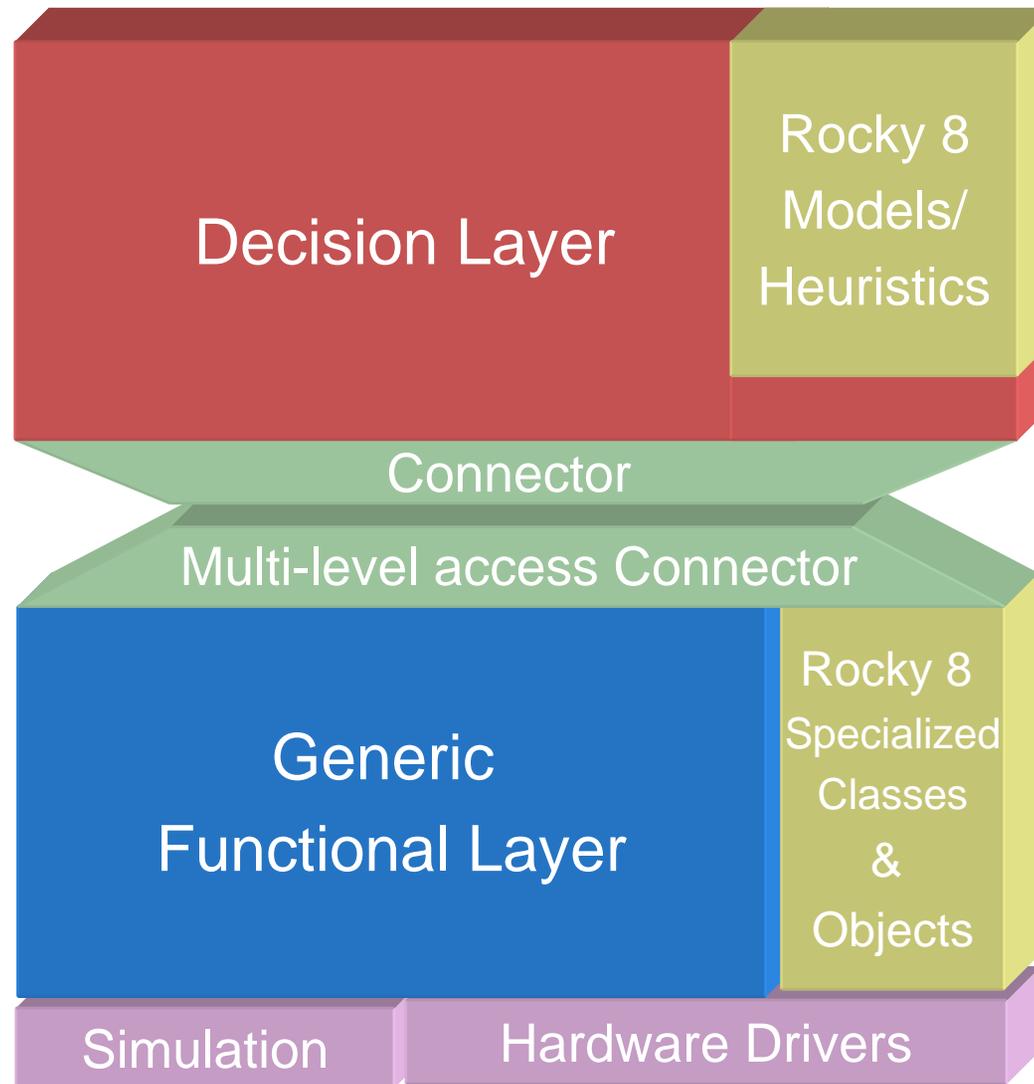


- Reusable
- HW reusable
- Non-reusable





Adapting to a Rover





Supported Platforms



Rocky 8

VxWorks x86

JPL



K9

Linux

x86

Ames



Rocky 7

VxWorks ppc

JPL



FIDO

VxWorks x86

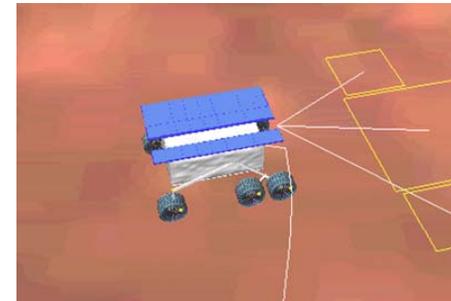
JPL



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

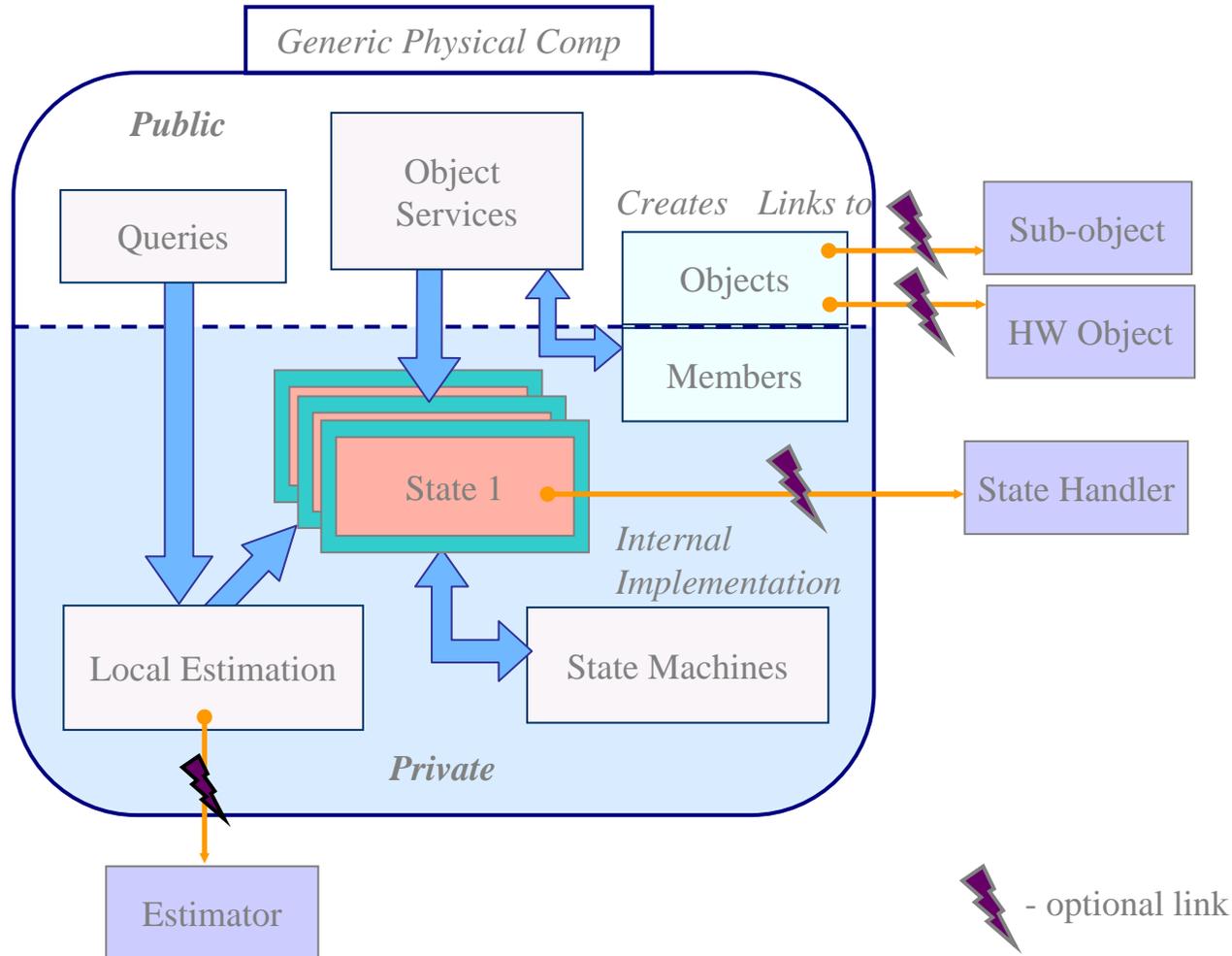


Summary



- 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





Collaborations



MTP

MIT <i>Williams</i>
CMU <i>Glymour/ Ramsey</i>

MDS <i>Lai</i>

CICT

ALERT <i>Matthies</i>
CLEaR <i>Estlin/Fisher</i>
Instr. Placement <i>Pedersen</i>
Craters <i>Cheng</i>
OASIS <i>Castano, Judd</i>

R&TD

Adv. Avionics <i>Bolotin</i>

New NRA
Competed Tasks

RMSA Competed Tasks

U. Washington <i>Olson</i>
CMU <i>Stentz</i>
U. Michigan <i>Borenstein</i>
JPL <i>Nenas</i>
JPL <i>Matthies</i>
ARC <i>Roush</i>
ARC <i>Dearden</i>
MIT <i>Dubowsky</i>
Ohio State U. <i>Li</i>

CLARAty
Jet Propulsion Lab
NASA ARC
CMU
U. Minnesota

U. Survey <i>Tunstel</i>

WITS <i>Backes</i>

Manipulation <i>Backes</i>

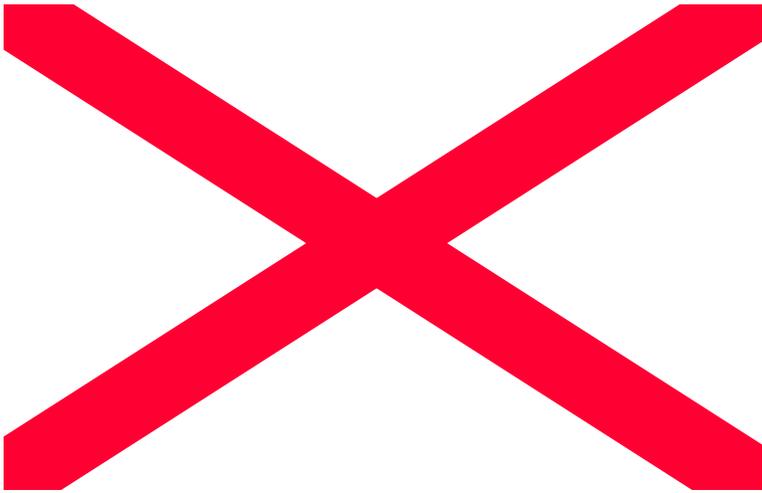
Science Sim <i>M. Lee</i>

ROAMS <i>Jain</i>

Rover Infrastructure <i>Baumgartner</i>
--

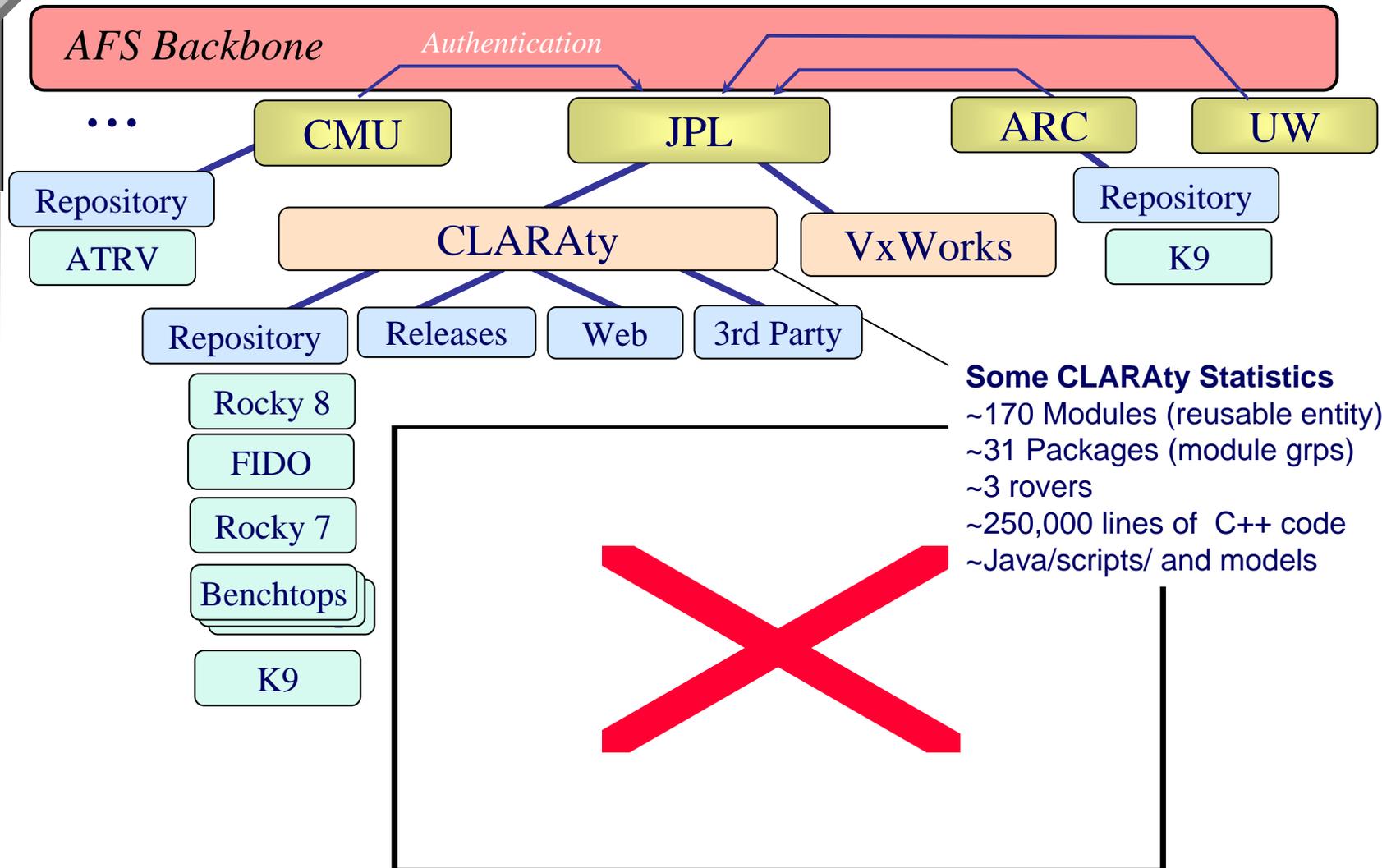
Validation Tasks

Inst. Placement <i>Kim</i>
Long Traverse <i>Huntsberger</i>
Auto. Science <i>Gat</i>

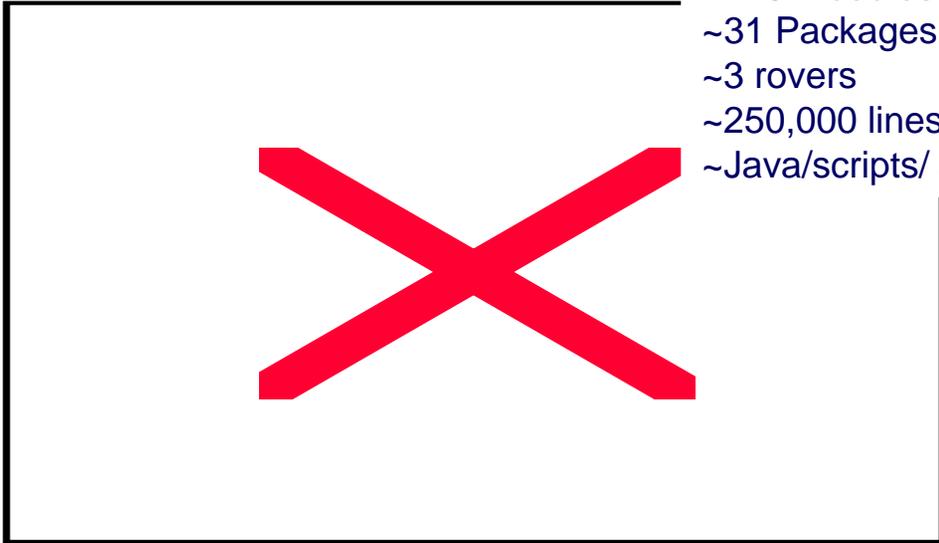




Software Development Process

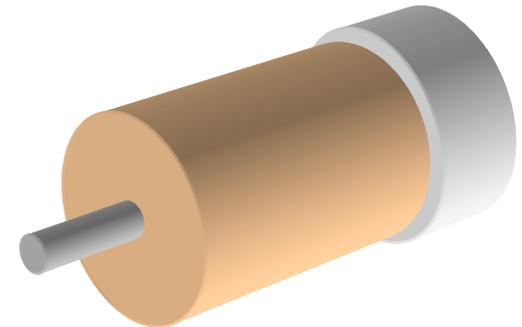
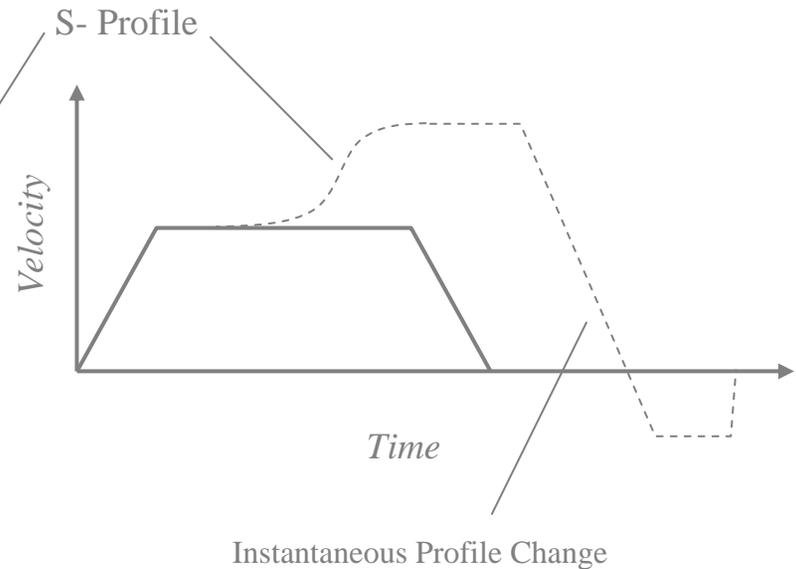


Some CLARAty Statistics
 ~170 Modules (reusable entity)
 ~31 Packages (module grps)
 ~3 rovers
 ~250,000 lines of C++ code
 ~Java/scripts/ and models



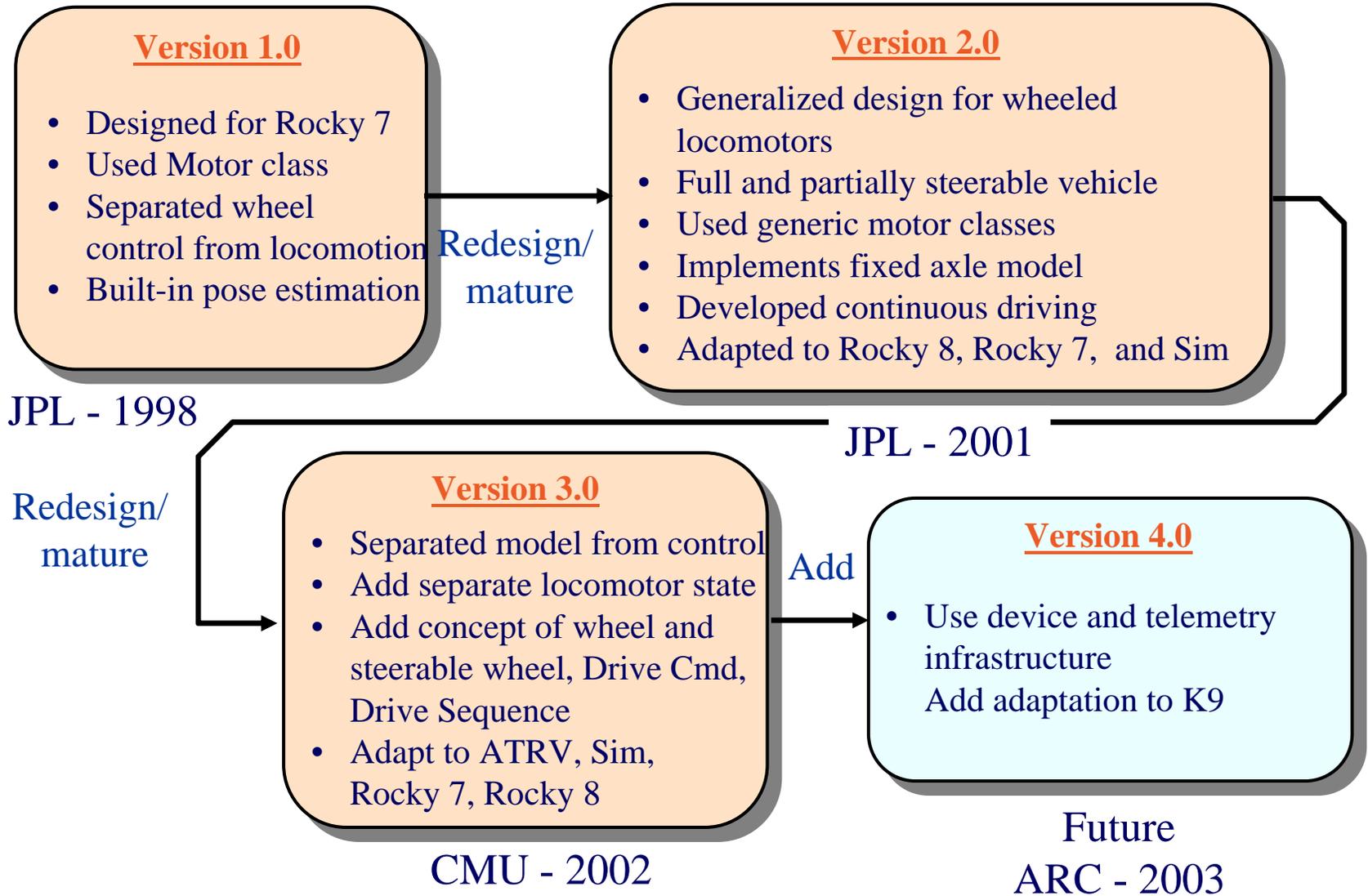
Example: Generic Controlled Motor

- 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





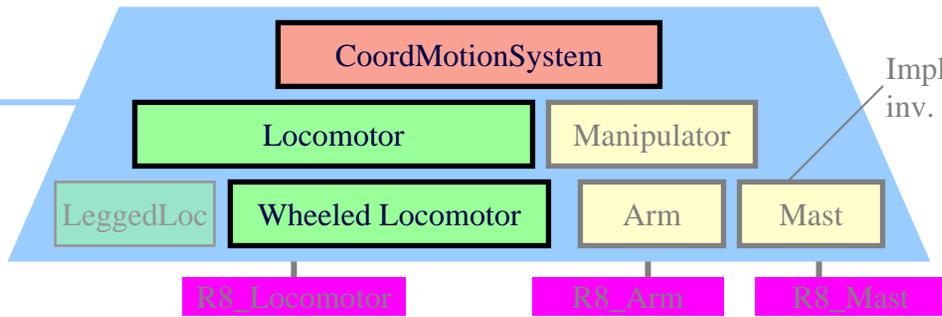
Example: collaborative development for locomotor



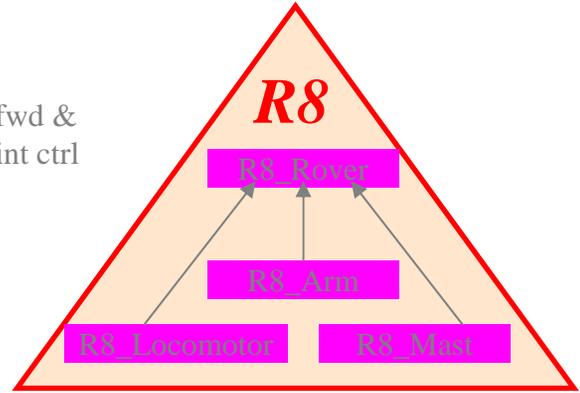


R8 Specific Rover Implementation

Non reusable Code Reusable Code

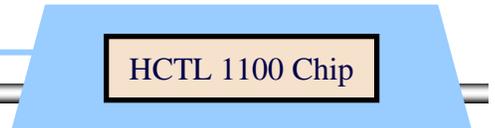
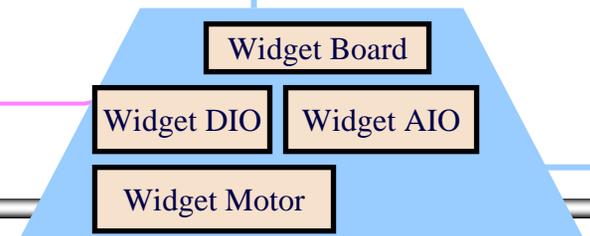
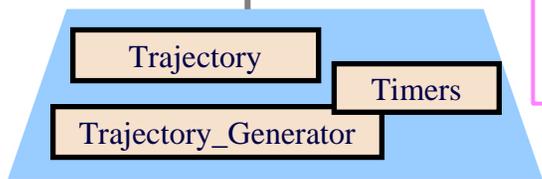
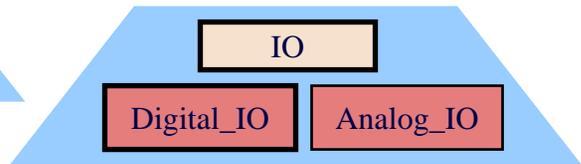
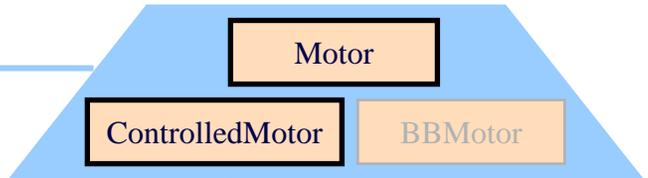


Implements general fwd & inv. kinematics & joint ctrl



- Attaches proper motors
- Restricts Steering to 2 wheels

- Specialized inv. Kinematics (overrides default)
- Attaches proper motors
- Attaches proper cameras for mast
- Adds filter wheel





Capabilities of Wheel Locomotor

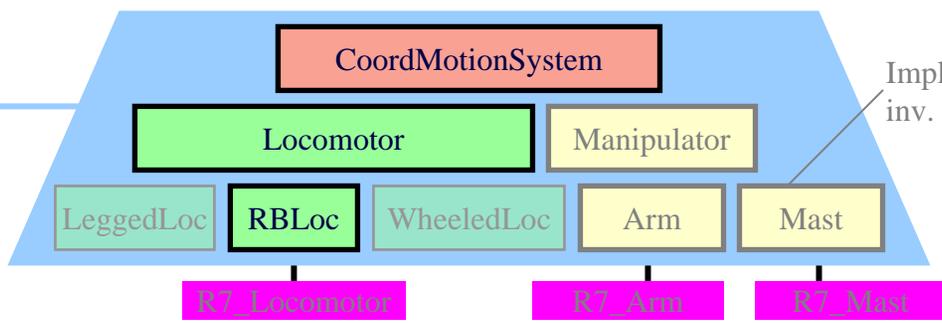


- Type of maneuvers:
 - Straight line motions (fwd / bkwd)
 - Crab maneuvers
 - Arc maneuvers
 - Arc crab maneuvers
 - Rotate-in-place maneuvers (arc turn $r=0$)
- Driving Operation
 - Non-blocking drive commands
 - Multi-threaded access to the Wheel_Locomotor class – e.g. one task can use Wheel_Locomotor 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

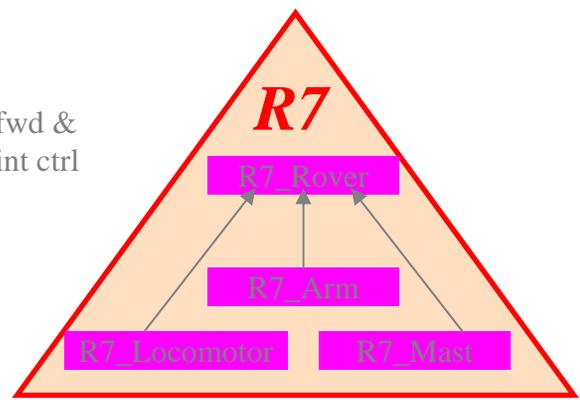


R7 Specific Rover Implementation

Non reusable Code Reusable Code

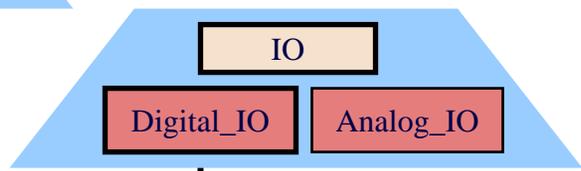
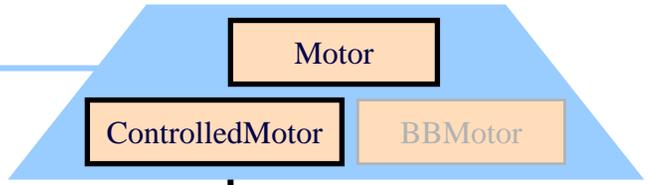


Implements general fwd & inv. kinematics & joint ctrl



- Attaches proper motors
- Restricts Steering to 2 wheels

- Specialized inv. Kinematics (overrides default)
- Attaches proper motors
- Attaches proper cameras for mast
- Adds filter wheel



Device Drivers





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