

# **Mars Rover Research and Software Infrastructure Development With CLARAty**

**Richard Volpe**

JPL Space Exploration Technology Program Office  
NASA Mars Technology Program  
2009 Mars Science Laboratory Project

**Issa Nesnas**

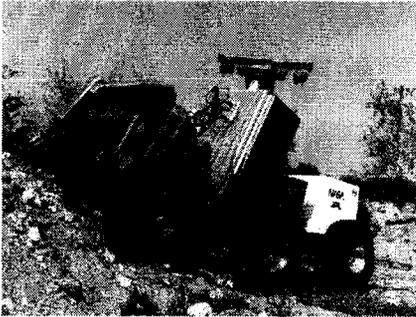
Mobility Concepts Technology Section

*Jet Propulsion Laboratory  
California Institute of Technology*

## **Outline**

- **Mars Rover Technology History Lessons**
- **CLARAty Overview**
- **Multi-Platform Support**
- **Multi-User Integration**
- **New Technology Validation and Infusion**
- **Summary**

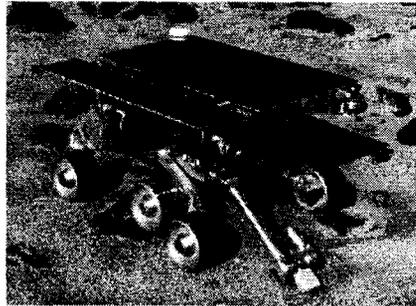
# ***JPL Rovers, Technology History***



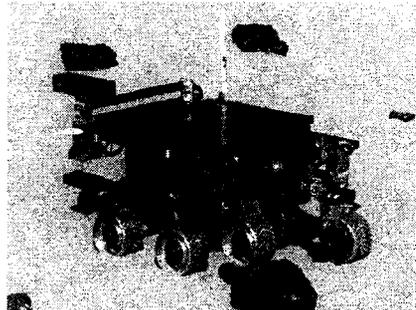
**1990, Robby:** stereo vision, 6 DOF manipulation, path planning, conditional sequence execution, Sun workstation and 68020 processors. (3m long, 1000kg mass) 100m traverse in arroyo.



**1994, Rocky 4:** bump/proximity sensing, 1 DOF rock chipper, reactive navigation algorithm, integrated spectrometer, 6811 processor. (60cm, 10kg). Tens of meters traverse in arroyo with soil return to lander with active beacon

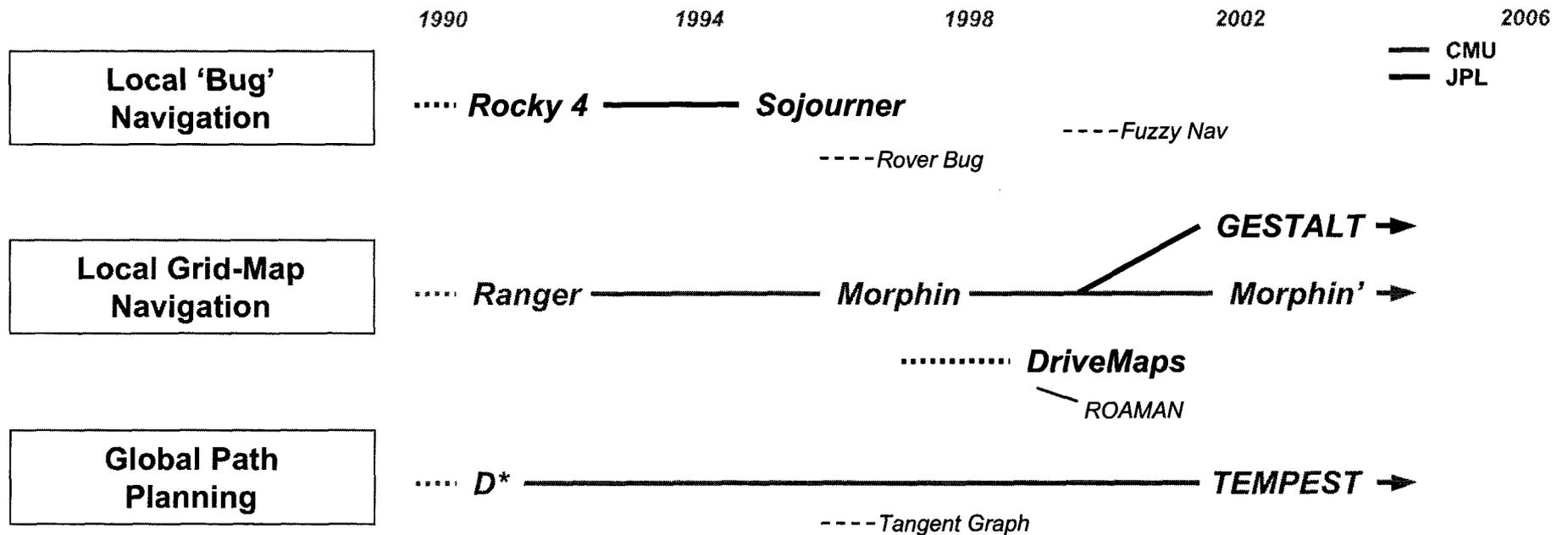


**1997, Rocky 7:** stereo vision, 4 DOF arm, 3 DOF mast, stereo vision for onboard hazard detection and path planning, sun sensing and gyro for heading determination, rock constellation based localization. 68060 processor, (60cm, 15kg). 1km integrated traverse in Mojave.



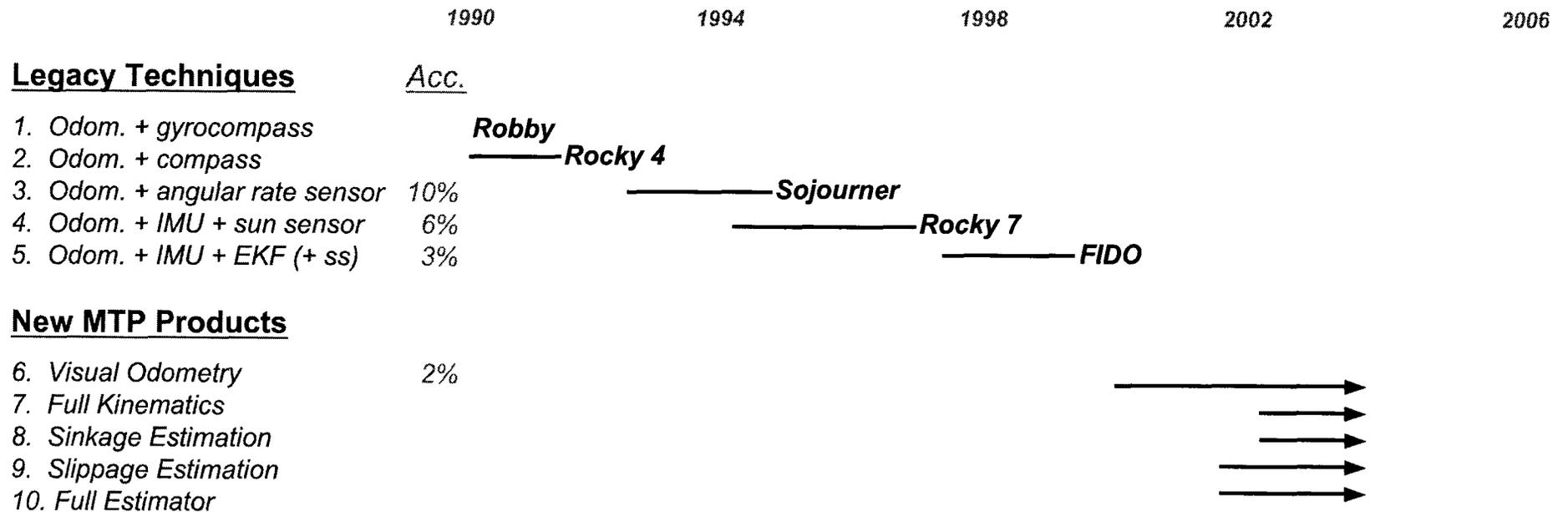
**2001, FIDO:** stereo vision, 4 DOF arm, 4 DOF mast, 3 DOF drill positioning, stereo vision for onboard terrain traversability measurement, sun sensing and IMU for position estimation, PII processor, (1m, 50kg). 20 sol desert operations from JPL including traverse and instrument positioning.

# Taxonomy of Mars Rover Navigation Development



\* Note that integration typically requires refactoring of navigation systems into parts (e.g. maps, navigation, vision, etc.)

# Improvements in Mars Rover Position Determination



# So, what are the impediments to progress?

## **Duplicative efforts prevent attainment of critical mass:**

- ***Parallel Duplication:*** Many mobile robot projects within NASA funded institutions are building systems of similar functionality without sharing the burden of software infrastructure development.
- ***Serial Duplication:*** New starts of projects often wipe the slate clean to eliminate old system problems and lack of familiarity or trust with previous product. Typically, software with legacy is due solely to a single individual or team, not the community.

## **Need to Follow software community lead:**

- ***Open source movement:*** The value of shared software is illustrated by Linux, GNU, Intel Computer Vision Library, etc.
- ***Object oriented design:*** It dominates software development, especially in industry, but is under-utilized in robotics.

## **Leveraging complimentary efforts:**

- ***Software sharing:*** Across related tasks within NASA (and DoD) is often arduous and rare.

# **Unified Research Software Motivation**

- 1. Legacy Capture**
- 2. Tighter Coupling of Robotics & AI**
- 3. Multi Platform Support**
- 4. Multi User Integration**
- 5. Complementary Algorithm Leveraging**
- 6. Competitive Algorithm Comparison**
- 7. Technology Validation**
- 8. Mission Infusion**

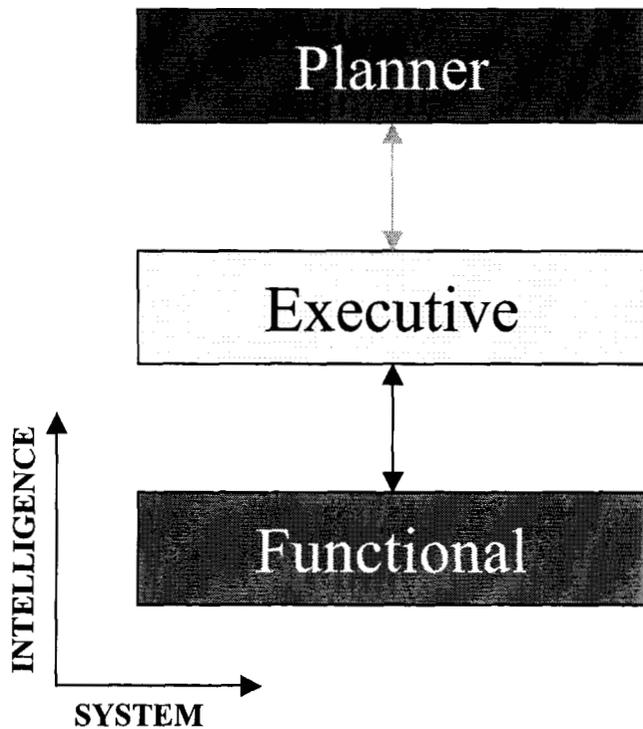
# **Enabling Architecture**

**CLARAty = Coupled Layer Architecture for Robotic Autonomy**

# A View of Architecture Hierarchy

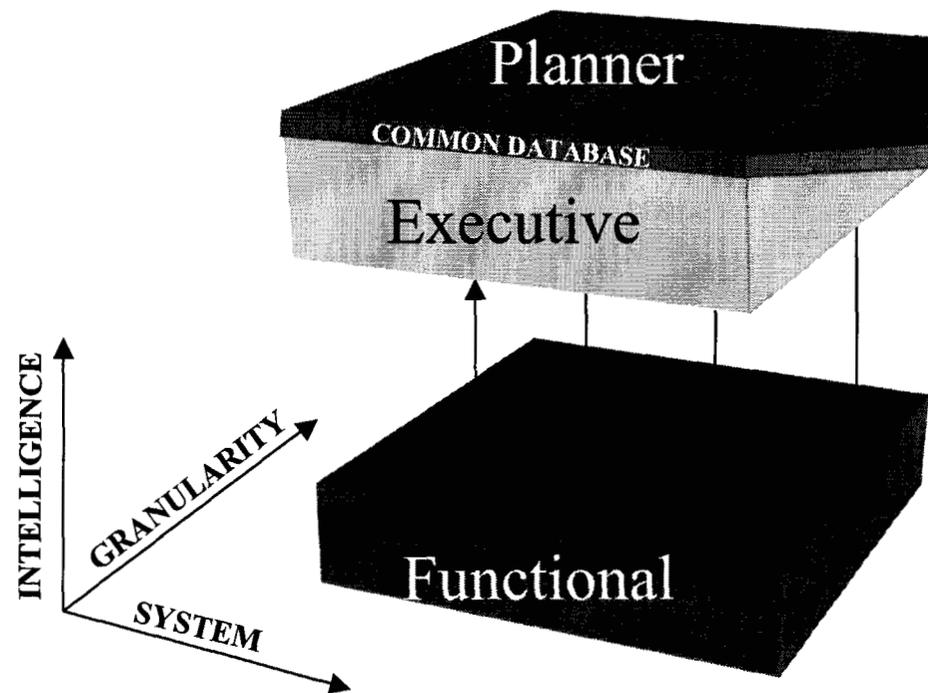
## Typical 3 Level Architecture

- Functional Level is often flat – typically a thin layer over the hardware
- Planner has no access to Functional Layer.
- Abstraction and granularity is mixed with intelligence.



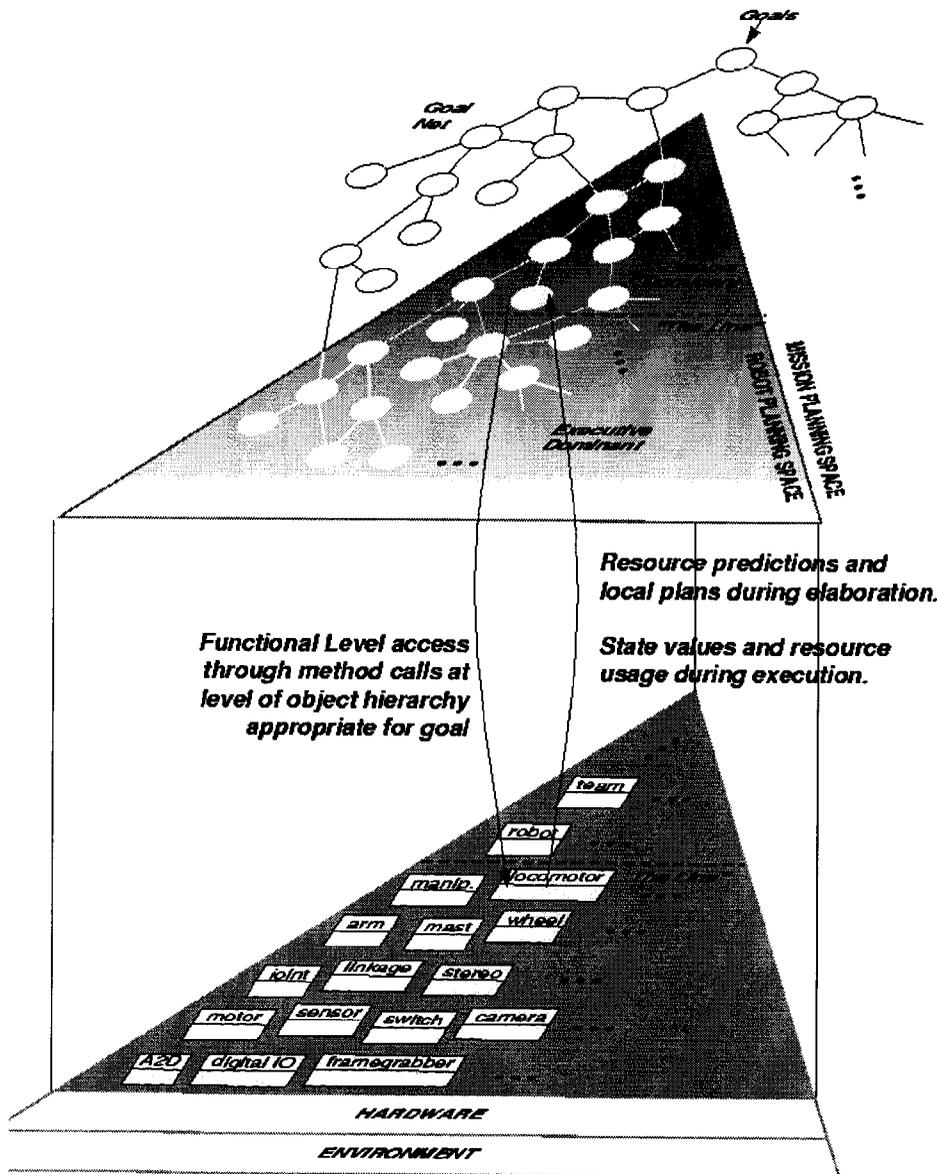
## CLARAty 2 Layer Architecture

- Functional Layer contains object-oriented abstraction of hardware at all levels of system granularity.
- Planner and Exec are similar, dominating at different levels of granularity, sharing a common database.
- Planner does not have direct access to the Functional Layer for execution, but executive may be minimized.



# CLARAty Two-Layered Architecture

CLARAty = Coupled Layer Architecture for Robotic Autonomy



## THE DECISION LAYER:

Declarative model-based  
Mission and system constraints  
Global planning

## INTERFACE:

Access to various levels  
Commanding and updates

## THE FUNCTIONAL LAYER:

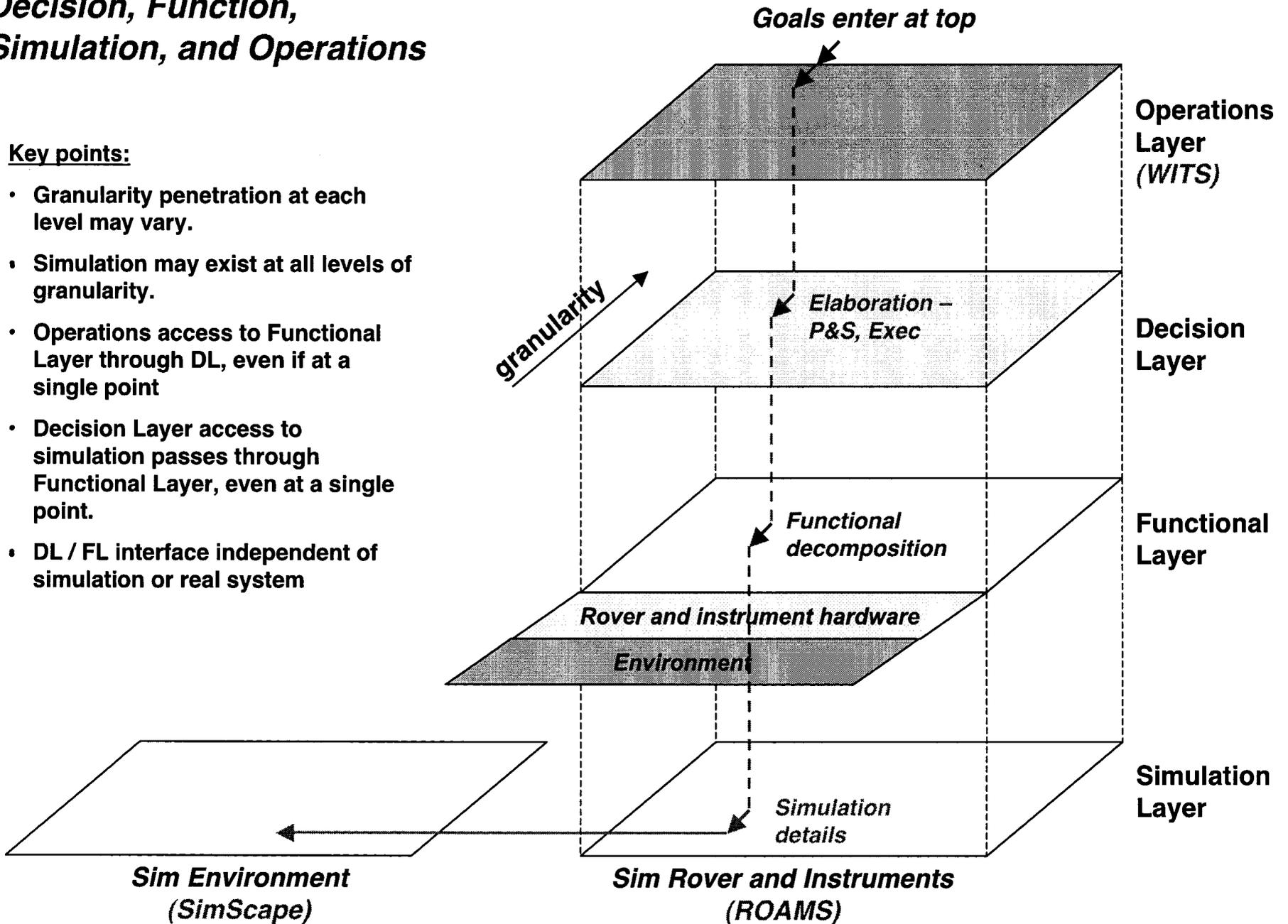
Object-oriented abstractions  
Autonomous behavior  
Basic system functionality

Adaptation to a system

# Granularity Matching between Layers of Decision, Function, Simulation, and Operations

## Key points:

- Granularity penetration at each level may vary.
- Simulation may exist at all levels of granularity.
- Operations access to Functional Layer through DL, even if at a single point
- Decision Layer access to simulation passes through Functional Layer, even at a single point.
- DL / FL interface independent of simulation or real system



Plans Results Targets Sequence Execution Group

File Edit Action Analysis Expansion

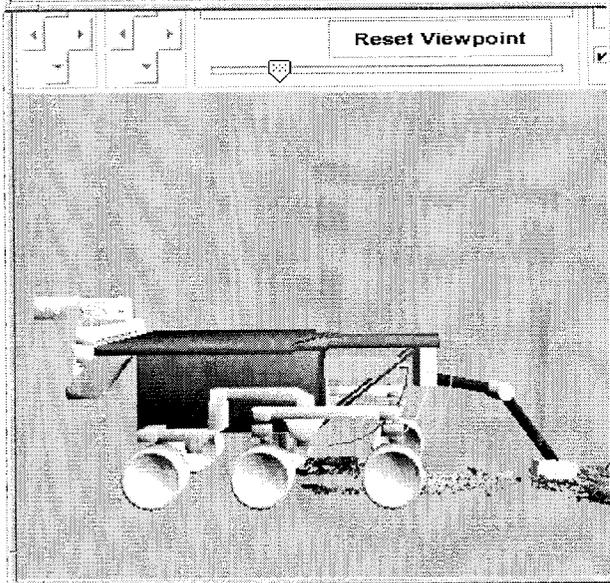
cmd\_armPosXYZ 89 539,7 732,15.410,-0.2180,-0.2284,0 9488  
 Start time= 1999-123T12:34:56  
 Duration (st,mc,rq,sq)= 0T0:0:0, 0T0:0:0, 0T0:0:0, 0T0:0:0  
 Energy (st,mc,rq,sq)= 0, 0, 0, 0  
 Total Data Volume (st,mc,rq,sq)= 461000, 461000, 461000, 2610999

arm\_cmds

- cmd\_armPosXYZ
- M\_cmd\_armPosXYZ
- cmd\_armPosXYZSlew
- cmd\_armPosXYZZbrush
- cmd\_armCoreView
- cmd\_CMI\_image
- cmd\_MosSpec
- cmd\_armMoveRel
- cmd\_armMoveAbs
- cmd\_armDeploy
- cmd\_armStow

- Macro(20, cmd\_CMI\_image)
  - Step(1, 250, cmd\_CMI\_image)
- Request(17, 11)
  - Macro(21, SIM\_set\_viewpointAndZoom)
    - Step(1, 260, SIM\_SET\_VIEWPOINT)
    - Step(2, 270, SIM\_SET\_VIEWZOOM)
  - Macro(22, cmd\_armDeploy)
    - Step(1, 280, cmd\_armDeploy)
  - Macro(23, M\_cmd\_armPosXYZ)
    - Step(1, 290, cmd\_armPosXYZ)
  - Macro(24, cmd\_FrontHazcam)
    - Step(1, 300, cmd\_FrontHazcam)
  - Macro(25, SIM\_snapshot\_stereo)
    - Step(1, 310, SIM\_SET\_VIEWCONE\_STEREO)
    - Step(2, 320, SIM\_snapshot\_stereo)
  - Macro(26, cmd\_armStow)
    - Step(1, 330, cmd\_armStow)

```
<< washU->sowg: Notice my marker on the large boulder low on the cliff; we might want to investigate that in a later sol
<< denmark->sowg: Take partial multi-tier panorama of pile
>> COT->sowg: OK, first version sequence ready for review
```



File Image Resize Action Headings MinMax

(x,y,heading) wrt site 4 = (0, 0, 348.); wrt lander = (3.096, 2.479, 348.)  
 Panorama= /sites/site-4/panoramaView/navcamPanorama-1  
 Camera= left; Color= bw; Resolution= lo; Timetag= none

A 2D panoramic view of a rocky terrain. The image is divided into a grid of 10x10 cells. Various markers and labels are visible, including 'washU', 'pile', 'denmark', 'ips1', 'ips2', 'ips3', 'cornell', 'la1', 'core', 'W3', and 'W2'. A coordinate box shows [ 9.925, -10.961, 2.299 ].

File Image Window Action

(x,y,heading) wrt site 6 = (0, 0, 12.); wrt lander = (3.744, 2.194, 12.)  
 Image= /sites/site-6/position-1/frontHazcam/view-1  
 Camera= left; Color= bw; Resolution= hi; Timetag= none

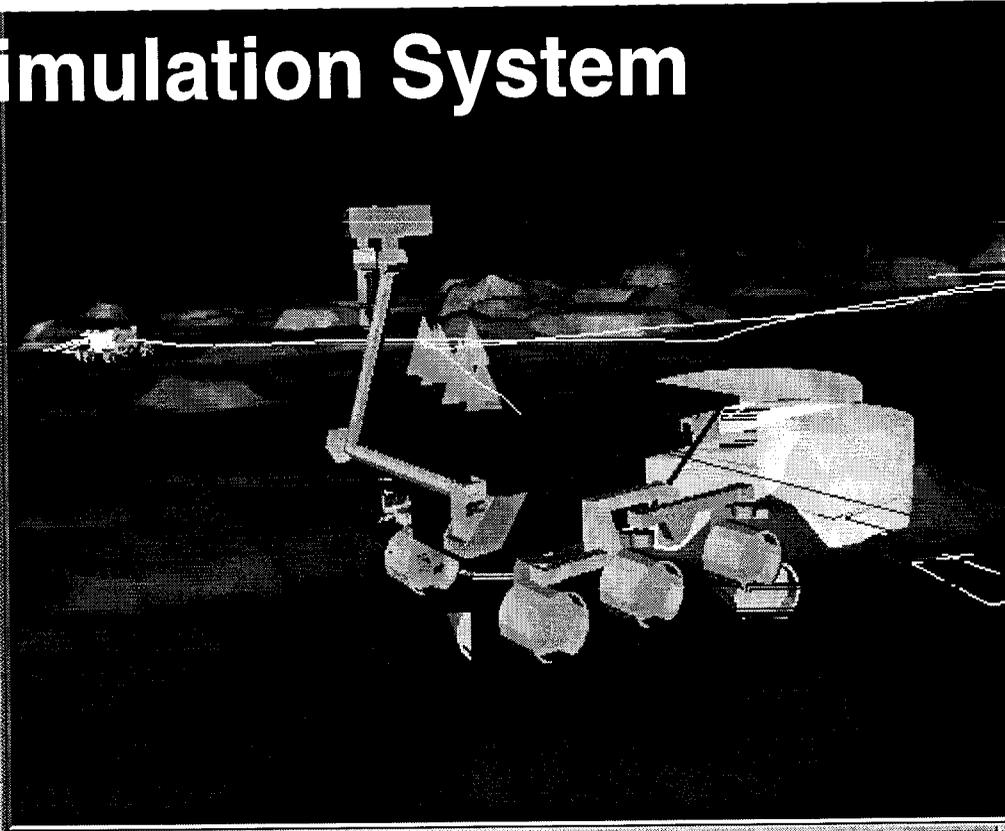
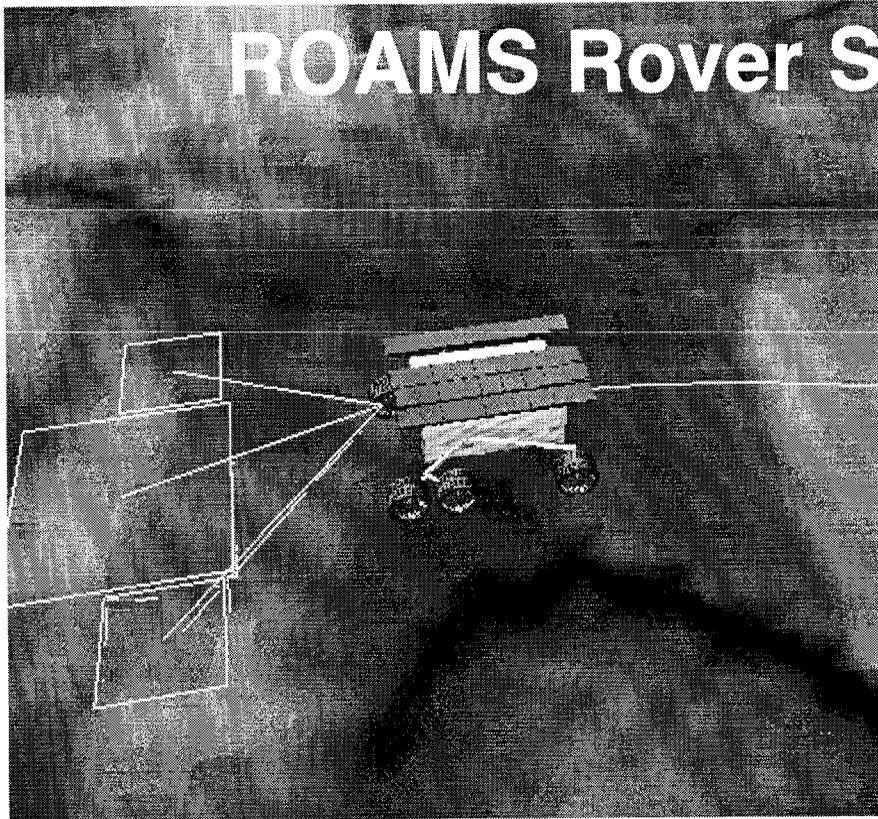
A 2D panoramic view of a rocky terrain. The image is divided into a grid of 10x10 cells. Various markers and labels are visible, including 'denmark', 'pile', 'ips3', 'cornell', 'la1', 'W3', and 'W2'. A coordinate box shows [ 0.486, -2.736, 0.323 ].

File Show Overlays Images

(x,y,heading) wrt site 6 = (0, 0, 12.); wrt lander = (3.744, 2.194, 12.)

A 2D panoramic view of a rocky terrain, similar to the previous image but with a different grid overlay. The image is divided into a grid of 10x10 cells. Various markers and labels are visible, including 'denmark', 'pile', 'ips3', 'cornell', 'la1', 'core', 'rover', 'W3', and 'W2'. A coordinate box shows [ 0.486, -2.736, 0.323 ].

# ROAMS Rover Simulation System



## Simulation Parameters

Start Clock	<input type="checkbox"/> Clock on <input checked="" type="checkbox"/> Clock off		
Translation(R7)	.125	Rotation(R7)	.3
Translation(FIDO)	.18	Rotation(FIDO)	.4
Hazard Height(R7)			
Hazard Height(FIDO)			

## Current Rover Position

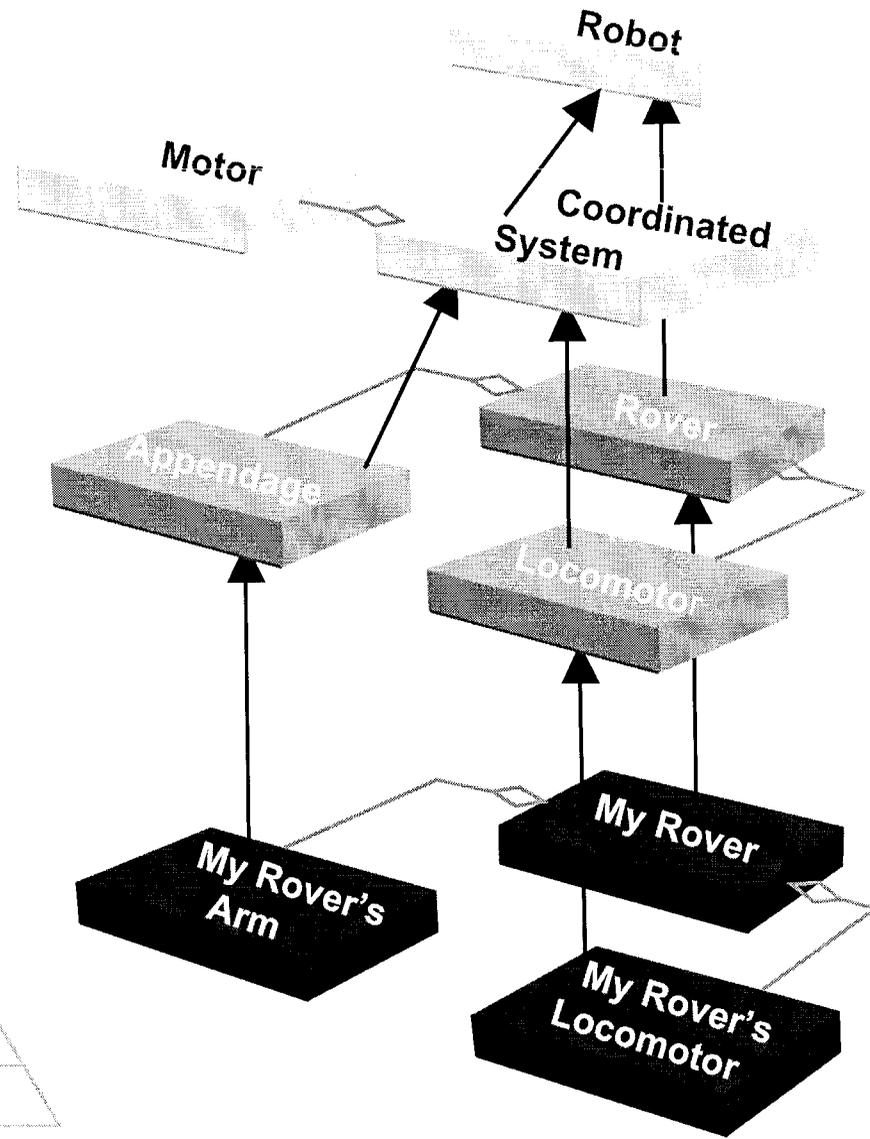
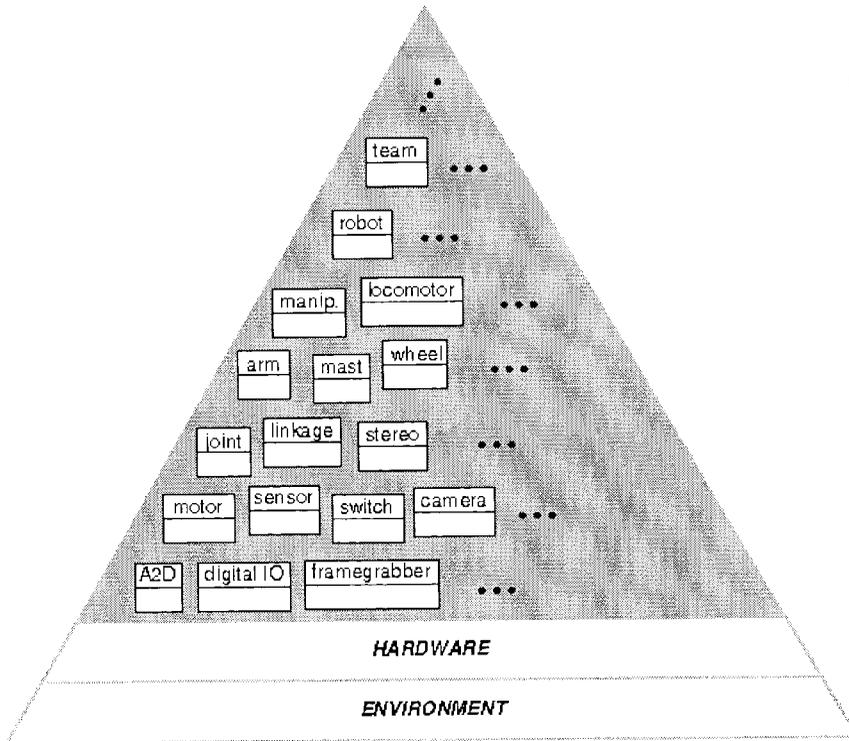
Place R7 (x,y,h)			
Current R7 x,y,h	-8.848275868	-4.91029427	0.124999999
Place FIDO (x,y,h)			
Current FIDO x,y,h	-1.149764815	-6.926942473	1.400000000

## Set Goal Position

R7 (x,y,h)	-12	-7	0
FIDO (x,y,h)	-3	-5.4	3.14



**CLARAty**  
**Functional Layer:**  
**Object-Oriented**  
**Hierarchy**

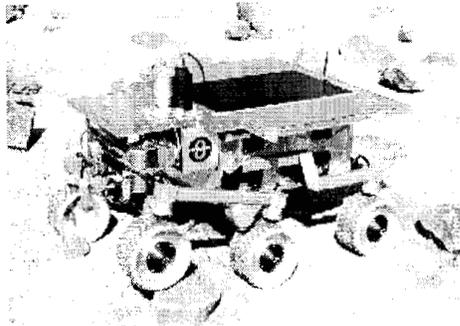


# FL Architectural Components

- Generic Physical Components (GPC)
  - Locomotor, Arm, Mast, Spectrometer, ...
- Specialized Physical Components (SPC)
- Generic Functional Components (GFC)
  - ObjectFinder, LayerDetector, VisualNavigator, StereoProcessor, ...
- Specialized Functional Components (SFC)
- Data Structure Components (DSC)
  - Array, Vector, Matrix, Map, Container, LinkedList, Bit
  - Image, Message, Resource

# **Multi Platform Support**

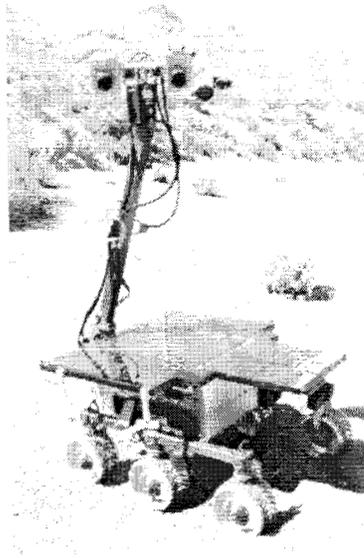
# Currently Supported Platforms



*Rocky 8*

VxWorks x86

JPL

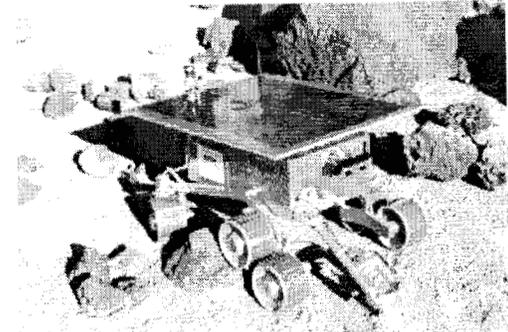


*K9*

Linux

x86

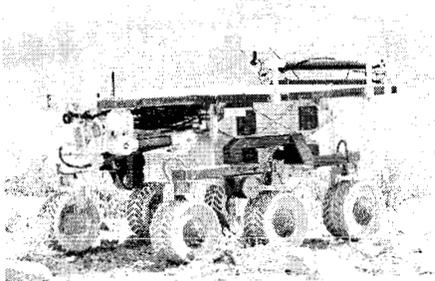
Ames



*Rocky 7*

VxWorks ppc

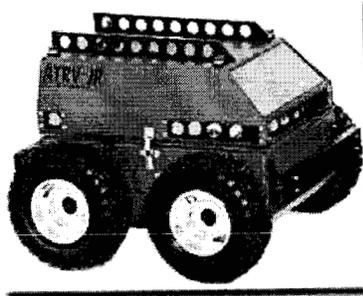
JPL



*FIDO*

VxWorks x86

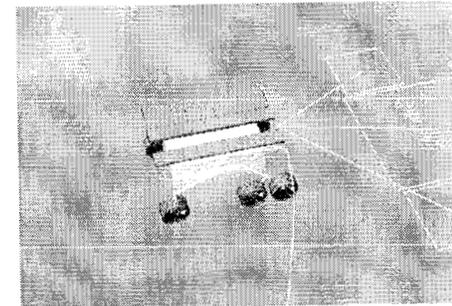
JPL



*ATRV*

Linux x86

CMU

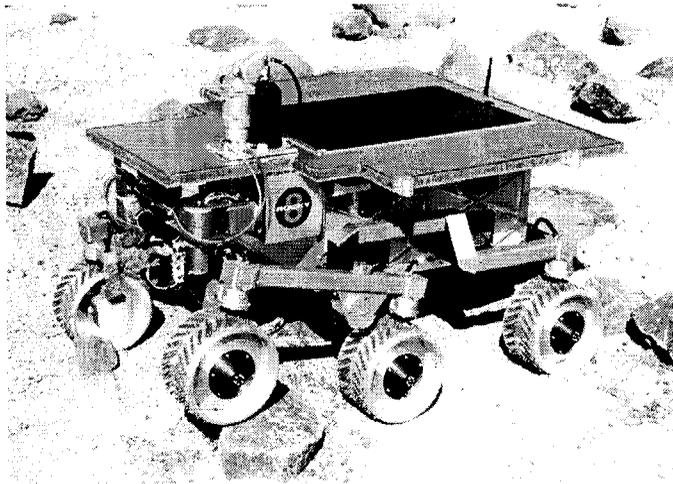


*ROAMS*

Solaris Linux

JPL

# Distributed Hardware Architecture



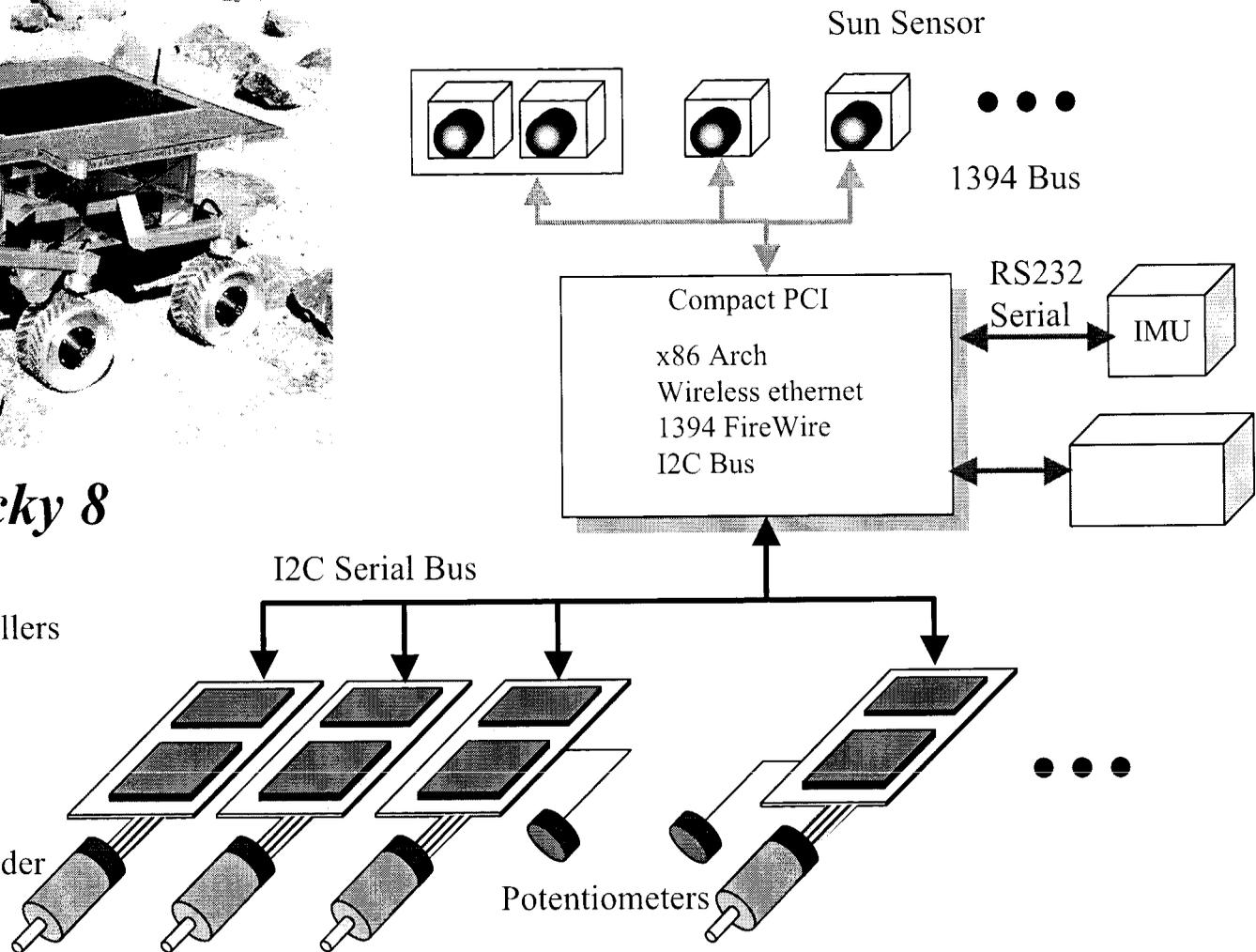
*Rocky 8*

## Widgets

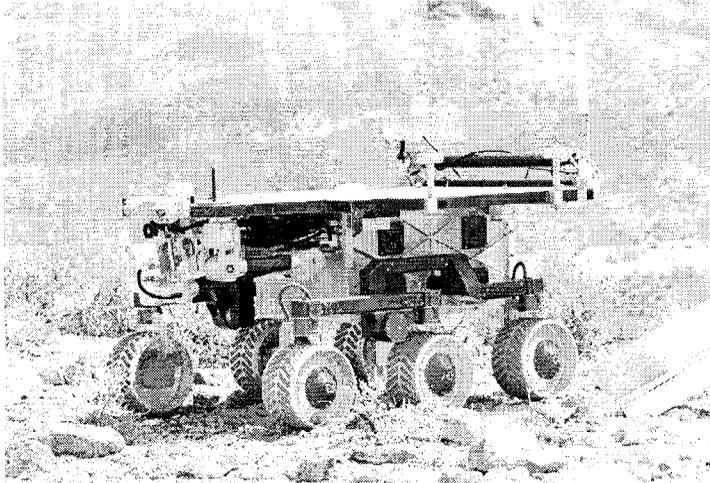
- Single Axis Controllers
- Current Sensing
- Digital I/O
- Analog I/O

Actuator/Encoder  
s

Potentiometers

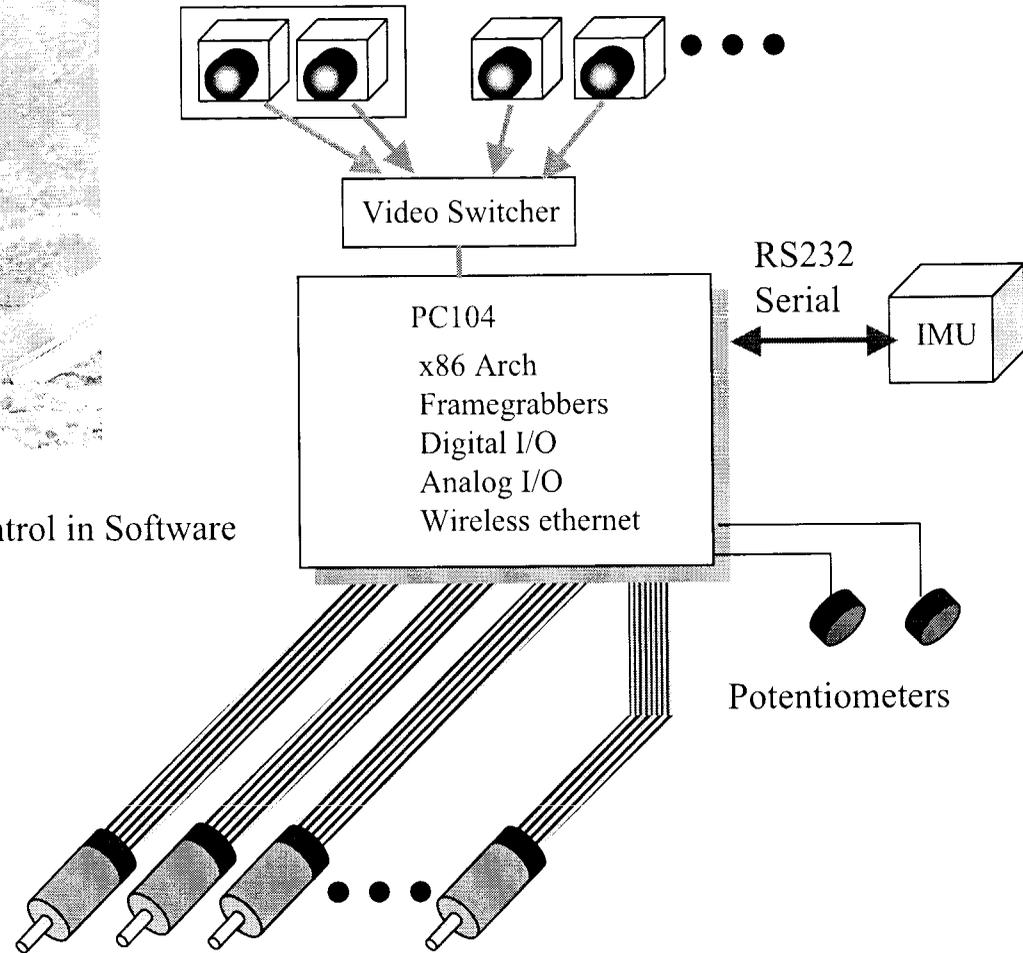


# Centralized Hardware Mapped Architecture

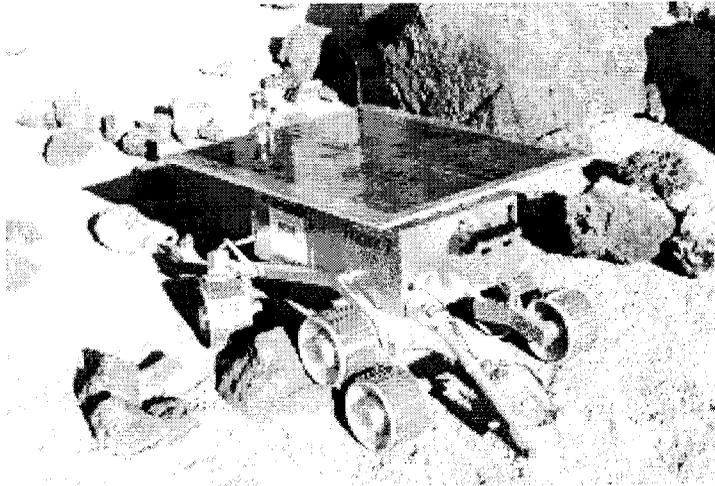


*Fido*

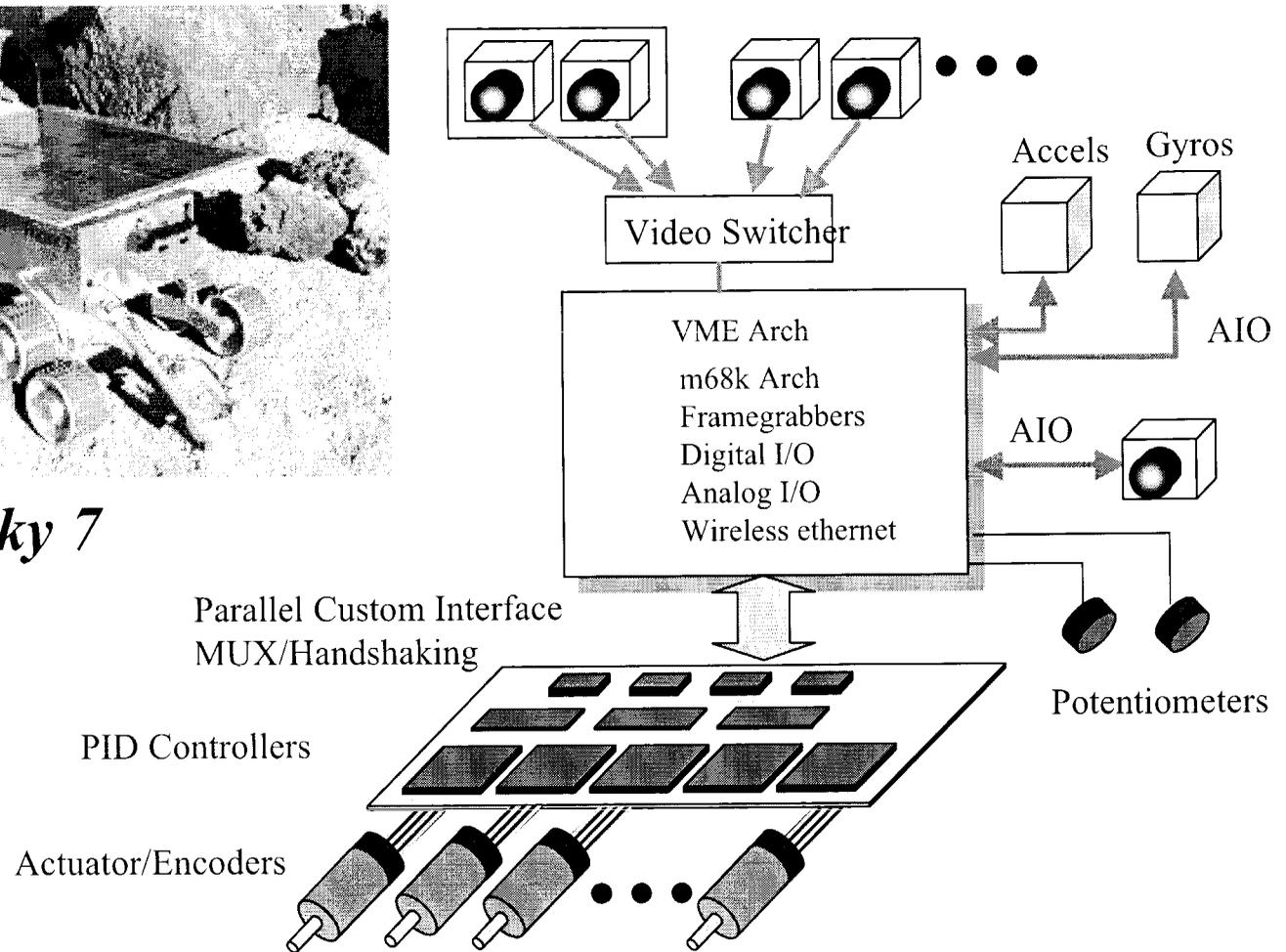
PID Control in Software



# Custom Architecture / Variability

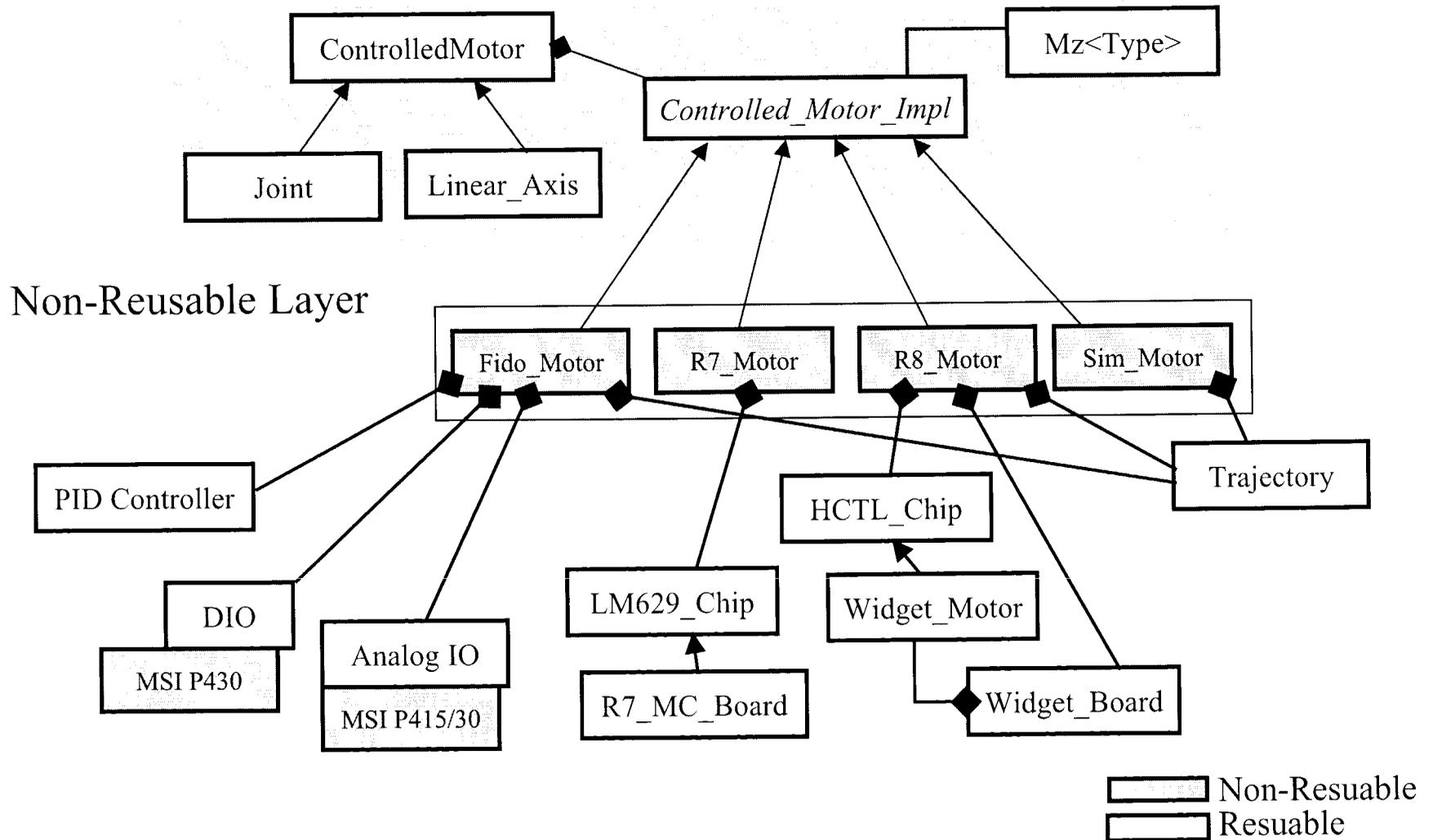


*Rocky 7*



# Supporting Different Platforms

*CLARAty adaptation process...*



# Code Reusability

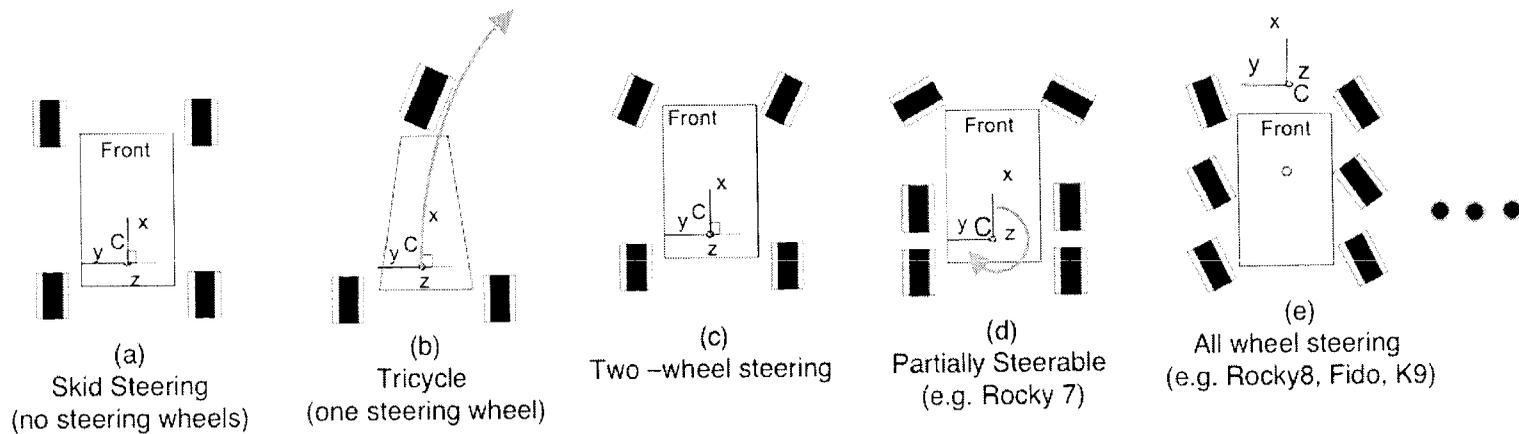
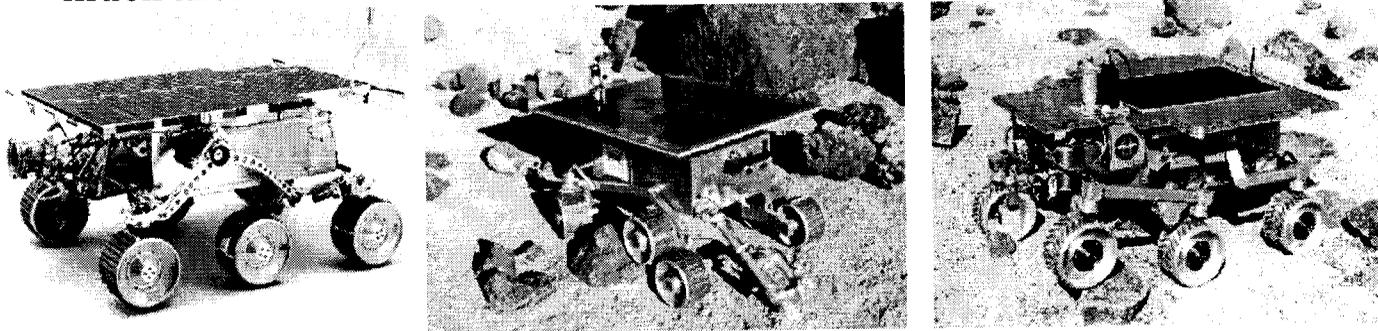
Rocky 7 Modules	Lines of Code	Status
Controlled Motor	2,652	Reusable
Input Output	2,690	Reusable
Bits	1,580	Reusable
Resources (Timers, etc)	725	Reusable
Rocky 7 Motor	927	Non-reusable
Rocky 7 H/W Maps	841	Non-reusable
Motor Controller LM629	1,143	Reusable
Digital I/O Board (S720)	576	Reusable
PCI Components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

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
Motor Controller HCTL	900	Reusable - HCTL
I2C Master	1,165	Reusable - I2C
I2C Master Tracii	1,223	Reusable - Tracii
<b>Total</b>	<b>16,314</b>	
<b>Total Reusable</b>	<b>89%</b>	
<b>Total Reusable - Strict</b>	<b>56%</b>	

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
<b>Total</b>	<b>17,341</b>	
<b>Total Reusable</b>	<b>79%</b>	

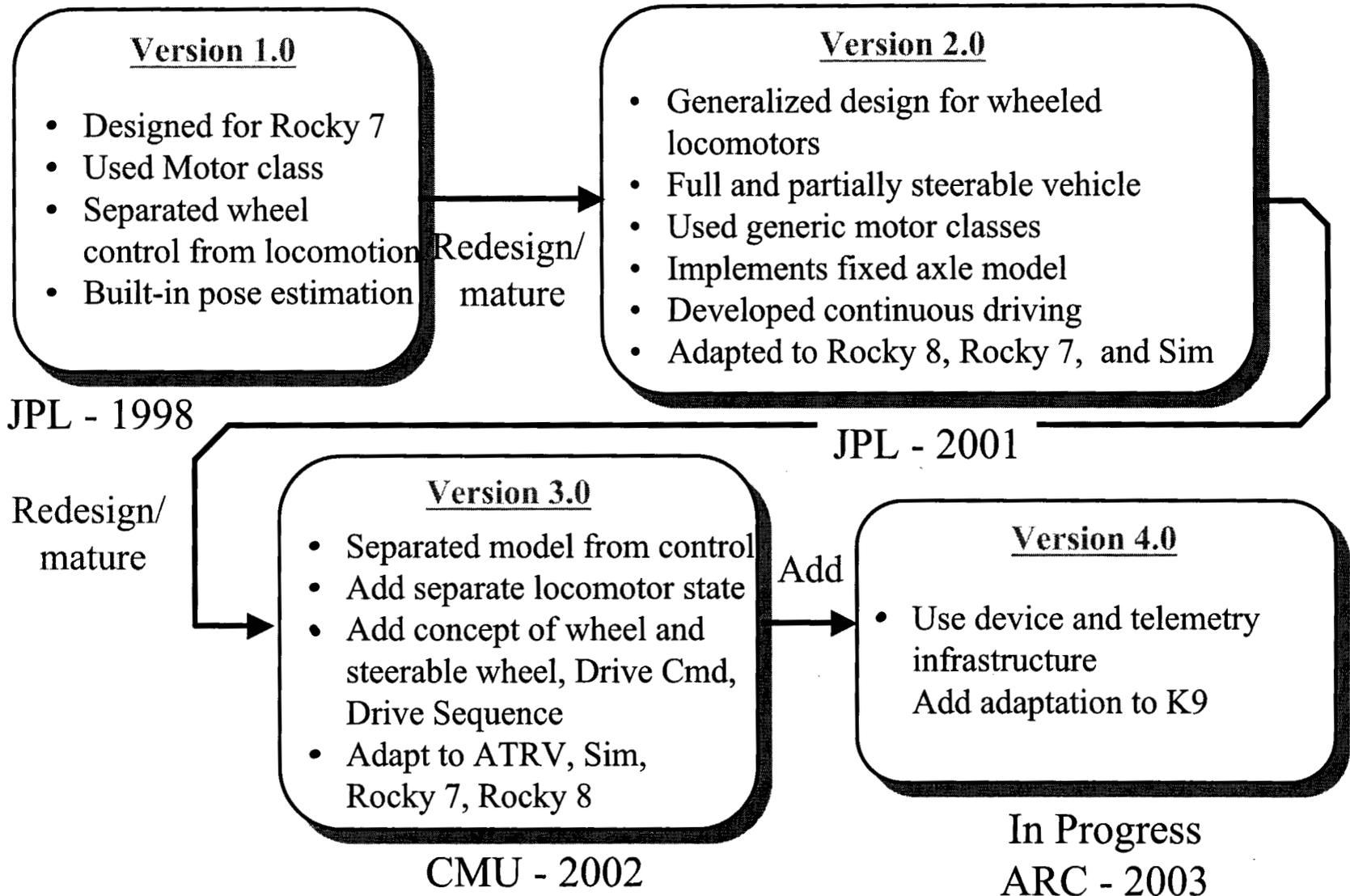
# Generic Algorithms

- Actual Example of Code Reusability for software modules:
  - Wheeled Locomotor Hierarchy – works for Rocky 8, Rocky 7, Fido, K9, and much more



Example:

## ***Collaborative Development for Locomotor***



# **Multi User Integration**

# ***CLARAty Development Team***

## **NASA Ames Research Center**

- Maria Bualat
- Sal Desiano
- Clay Kunz (*Data Structure Lead*)
- Randy Sargent
- Anne Wright (*Cog-E and core lead*)

## **Carnegie Mellon University**

- David Apelfaum
- Reid Simmons (*Navigation lead*)
- Chris Urmson
- David Wettergreen

## **University of Minnesota**

- Stergios Roumeliotis
- Master Student

## **MIT**

- Brian Williams
- Greg Sullivan

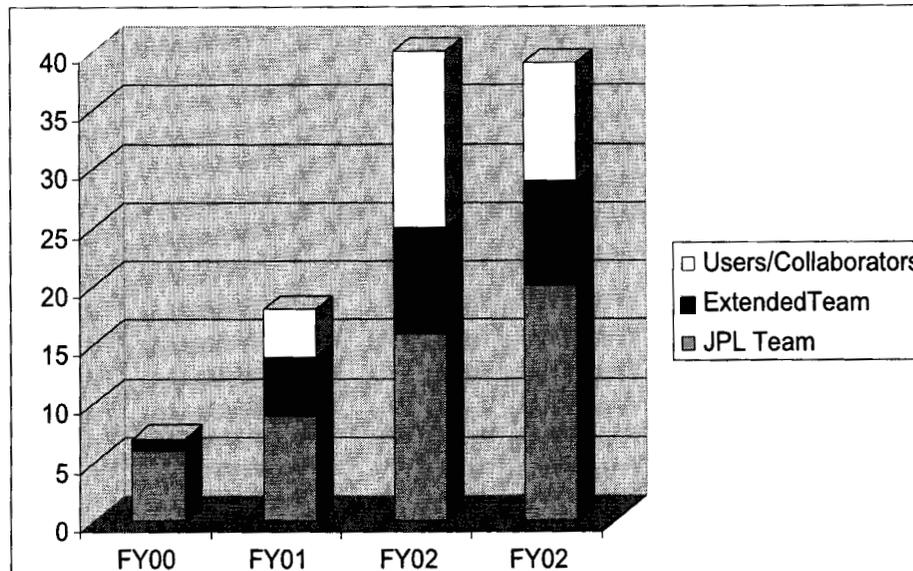
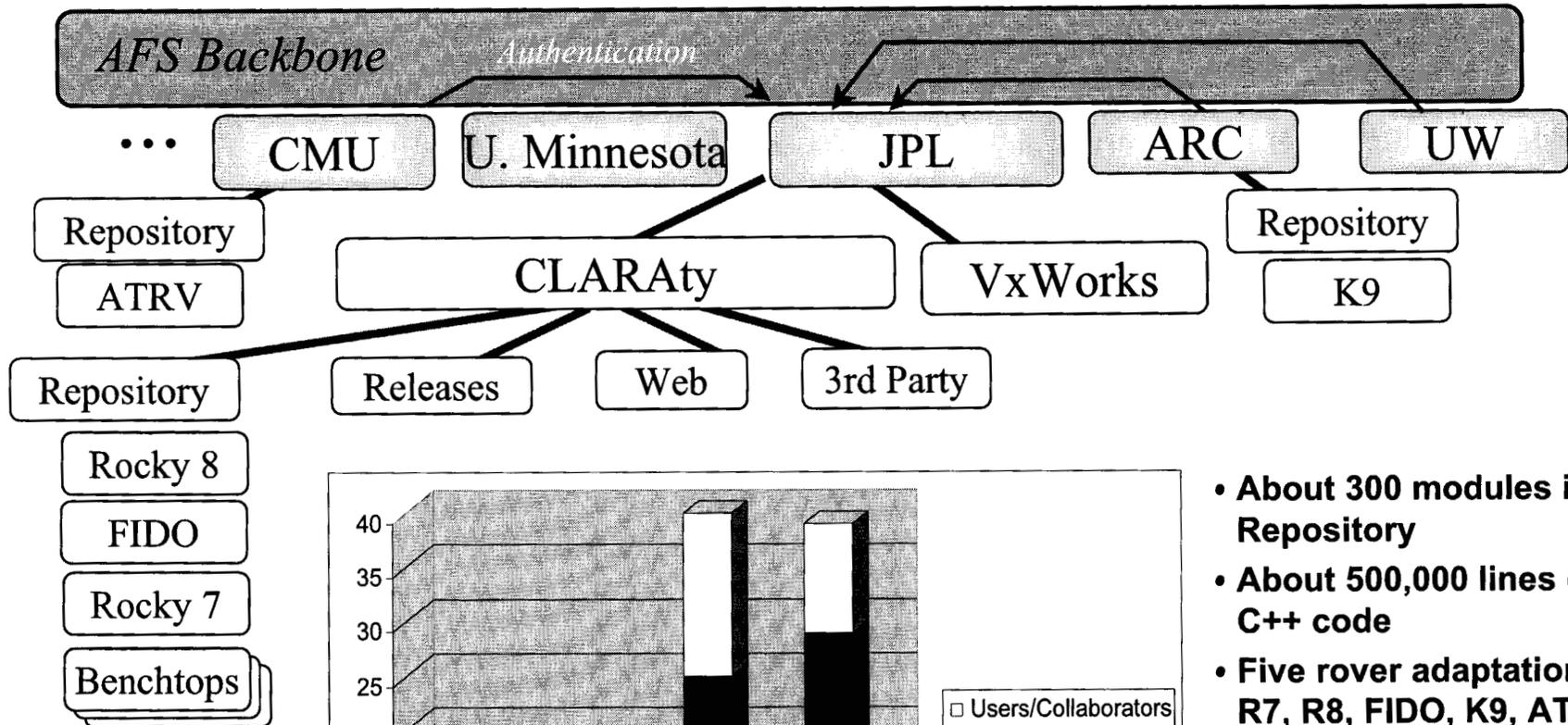
## **Jet Propulsion Laboratory**

- Max Bajracharya (*Cog-E & vision lead*)
- Edward Barlow (34)
- Antonio Diaz Calderon (34)
- Caroline Chouinard (36)
- Gene Chalfant (34)
- Tara Estlin (36) (*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**

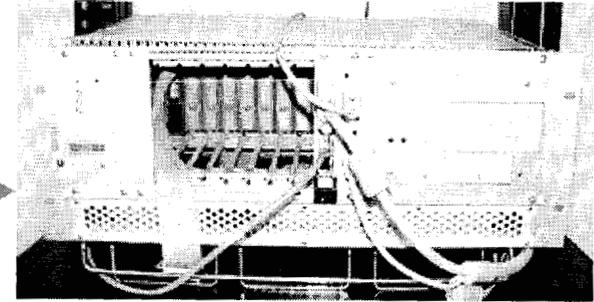
- Hari Das

# Software Development Network

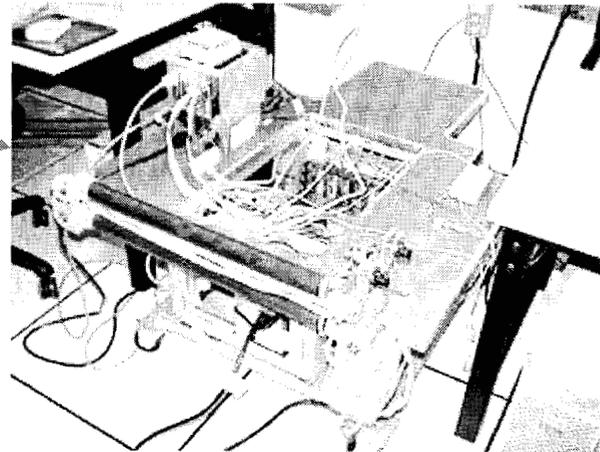


- About 300 modules in Repository
- About 500,000 lines of C++ code
- Five rover adaptations: R7, R8, FIDO, K9, ATRV
- Most technology modules are at Level I and Level II integration

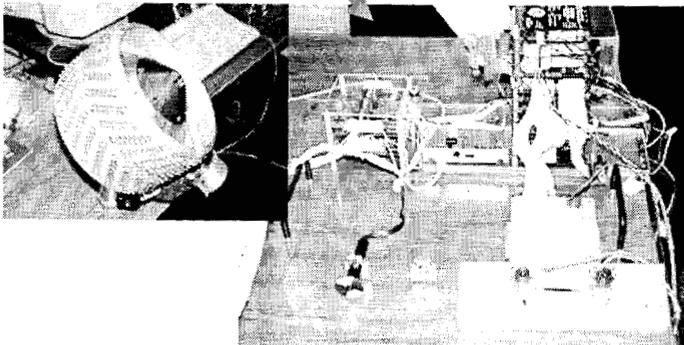
# CLARAty Testbeds



Rocky 8 Benchtop

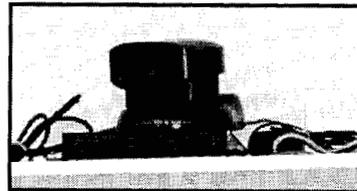


Dexter Manipulators  
Rocky 7 Benchtop

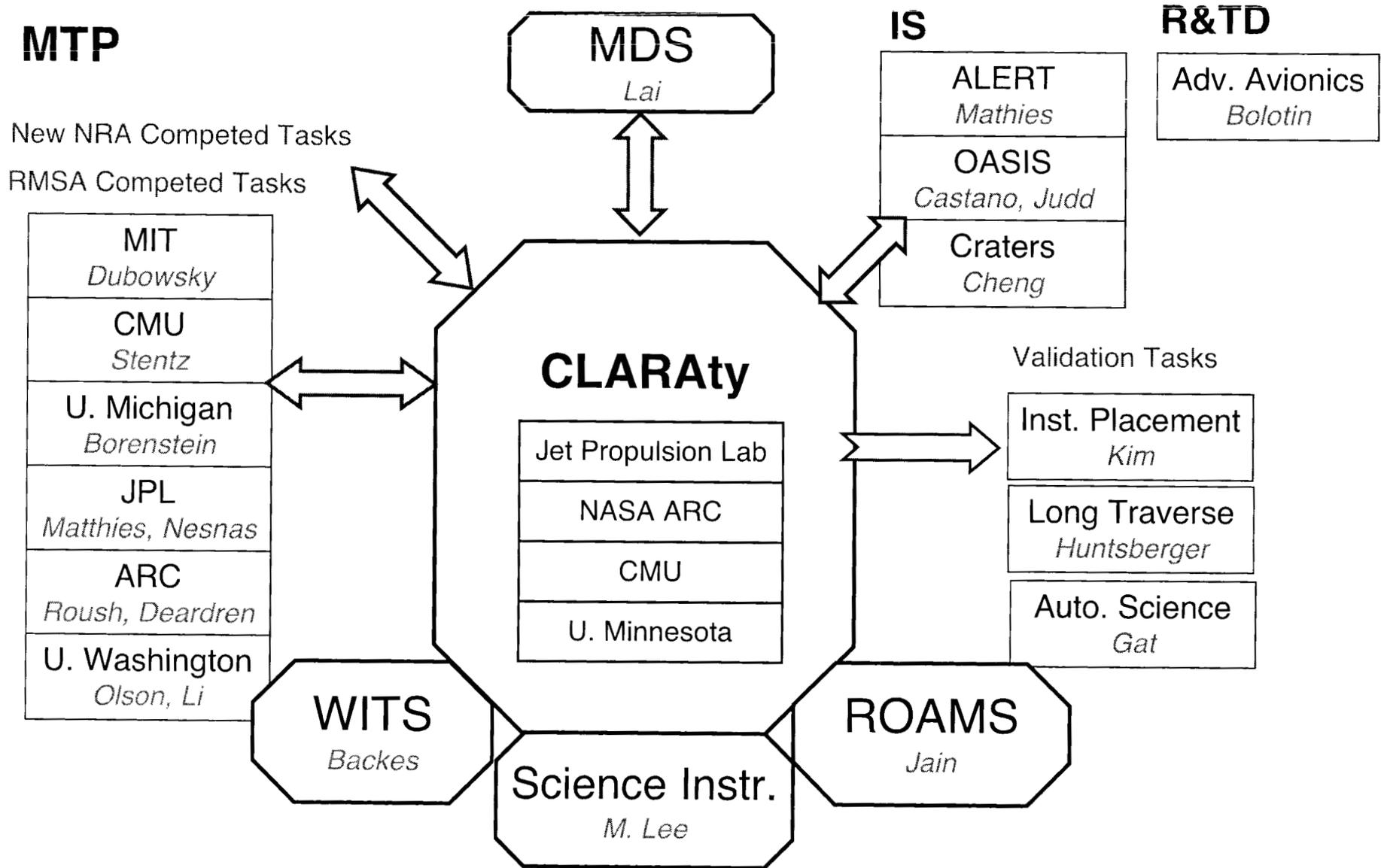


FIDO Benchtop

Lab Webcam



# CLARAty Team Interactions



# Levels of Integration

## **Level I - Deposited**

- Code exists in CLARAty repository - all Intellectual Properties items cleared
- Compiles as a standalone application - no dependencies to other modules
- Have test programs and user documentation for getting started

## **Level II - Encapsulated**

- Integrated with other CLARAty modules
- Uses CLARAty components to interact with rover
- Does not support a CLARAty API
- Runs on at least one robot platform

## **Level III - Integrated**

- Conforms to a generic CLARAty API (or parent class)
- Has no unsupported 3rd party dependencies
- Runs on all applicable rover platforms

## **Level IV - Refactored and Reviewed**

- Software reviewed by committee to ensure internal/external consistency
- Uses all applicable CLARAty classes
- Internally conforms to CLARAty conventions and coding standard

## **Level + - Reused**

- Re-used by other modules in CLARAty - dependent module
- Provides access to all internal data products

# *Is CLARAty Paying Off?*

## Some Data Points

- After integrating and tuning EKF into new estimation framework on Rocky 8 (3 months), integrating algorithm for FIDO took **3 days!**
- After testing CLARAty/Morphin navigator on Rocky 8, integrating and testing on FIDO took **2 days** and on K9 **1 week**
- After getting locomotion working on FIDO, moving to K9 took **4 days**
- After getting mast software to work on Dexter, moving to Rocky 8 took **2 weeks** and to FIDO **4 days**
- Adaptation of entire locomotion, motion control, I/O control and communication onto completely new avionics FPGA hardware with PPC405 took **2 weeks**

**N.B. These results are from professional CLARAty developers and should not be used to access development for new users**

# Technology Algorithms in CLARAty

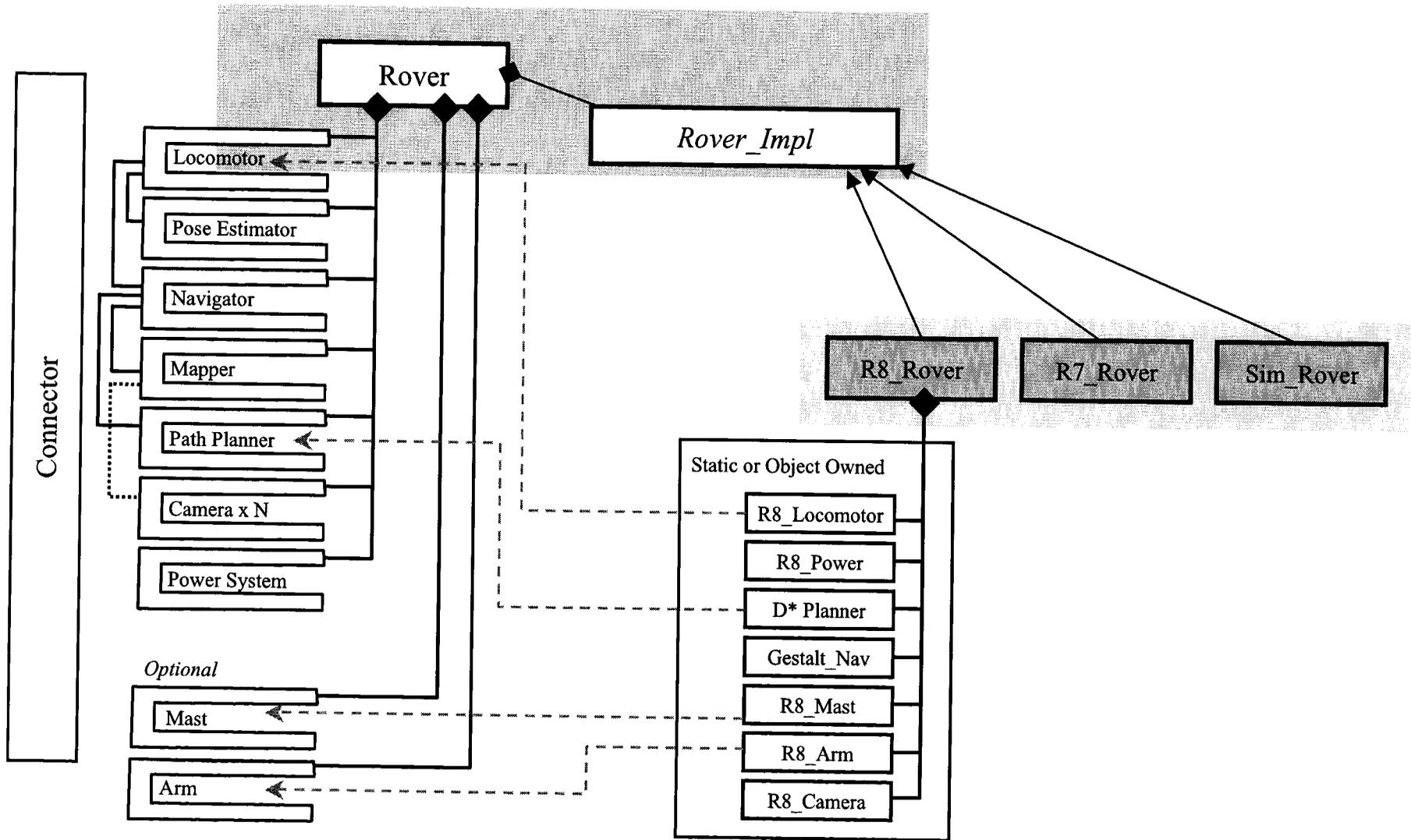
The header of the CLARAty website features a dark background with the NASA logo on the left and the Jet Propulsion Laboratory California Institute of Technology logo on the right. The word "CLARAty" is prominently displayed in the center. Below the main title is a navigation menu with the following items: Overview, Project, Software, Hardware, Testbed, How to, Packages, Documentation, Tools, and Conventions.

## Selected Robotics Algorithms (in CLARAty)

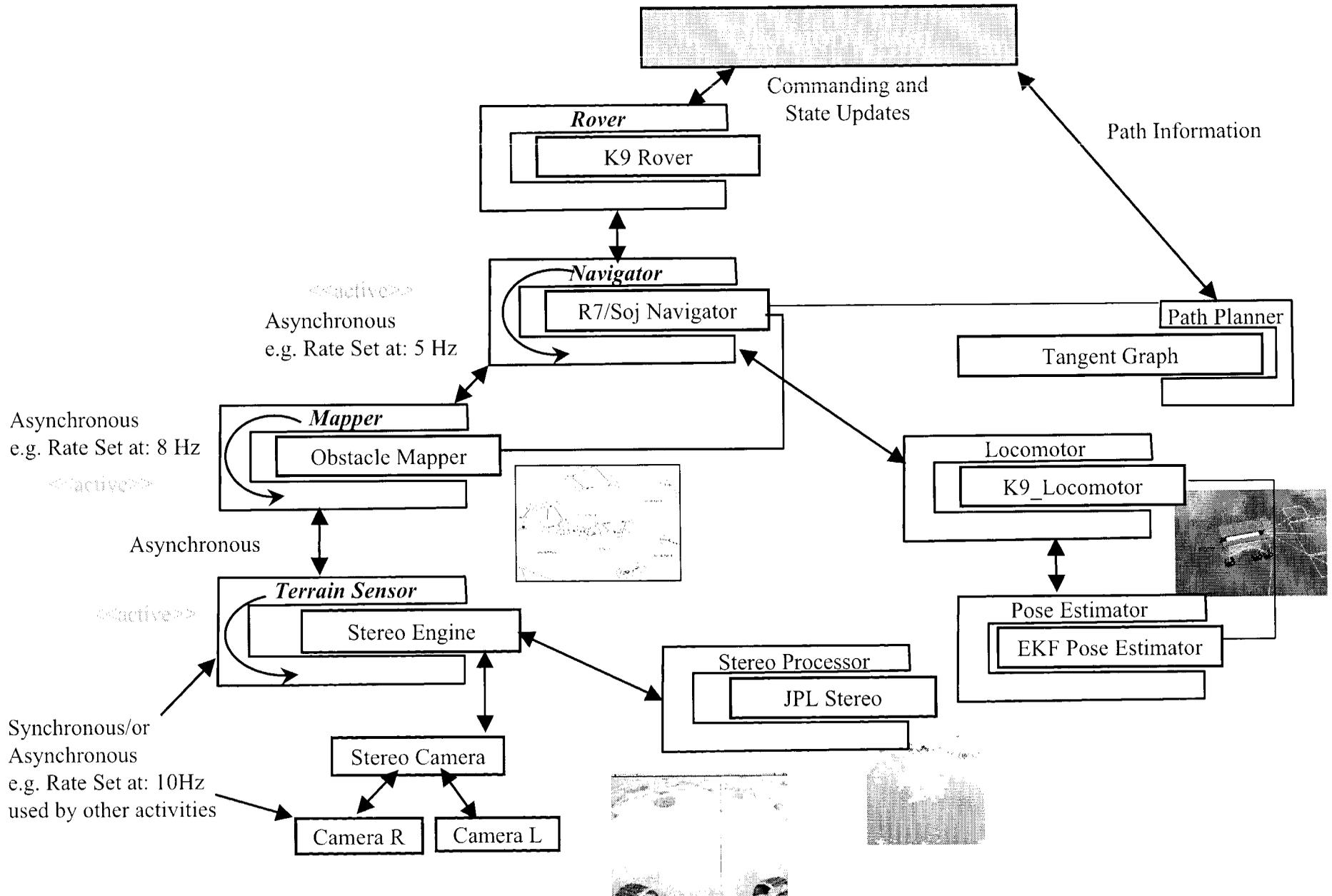
Package	Algorithm	Integration Level	Provider	Supported Platforms	Tested Platforms	Comments
Navigation	Morphin	III	Simmons	All	R8	
	GESTALT	II	Maimone	R8	R8	
	Grid Mapper	III+	Urmson	All	R8	Converts point cloud
Path Planning	D*	II	Stentz	All	R8	Not fully exercised
	Tangent Graph	I+	Laubach	Sim	Sim	
Pose Estimation	EKF	II+	Baumgartner	R8	R8	Migrating to new support all platforms
	Visual Odometry	II+	Matthies, Cheng	All	FIDO, R8, R7	R7 (only older version)
	Wheel Odometry	III+	Many	All	FIDO, R8	Fixed Steering
Vision	Corner Detection	III+	Many	All	R8, R7, K9	Harris
	Edge Detection	III+	Many	All	-	Canny
	Image Rectification	III+	Many	All	-	
	Image Pyramiding	III+	Many	All	-	
	Stereo processing JPL	II+	MV - many	All	R8, R7	
	Stereo processing SVS	II+	SRI	x86	x86	Binaries
	2D Feature Tracker	III	Bajracharya, Bandari	All	R8, R7	2D
2D/3D Feature Tracker	III	Nesnas et al	All	-	Expected Date: /	
Wide Baseline Stereo	II - III	Olson	All	-	Expected Date: /	
Locomotion	Wheel vehicles forward kinematics	III+	Many	All	R8, R7, FIDO, ATRV	
	Wheel vehicles inverse kinematics	III+	Many	All	R8, R7, FIDO,	

# **Competitive and Complementary Algorithm Integration**

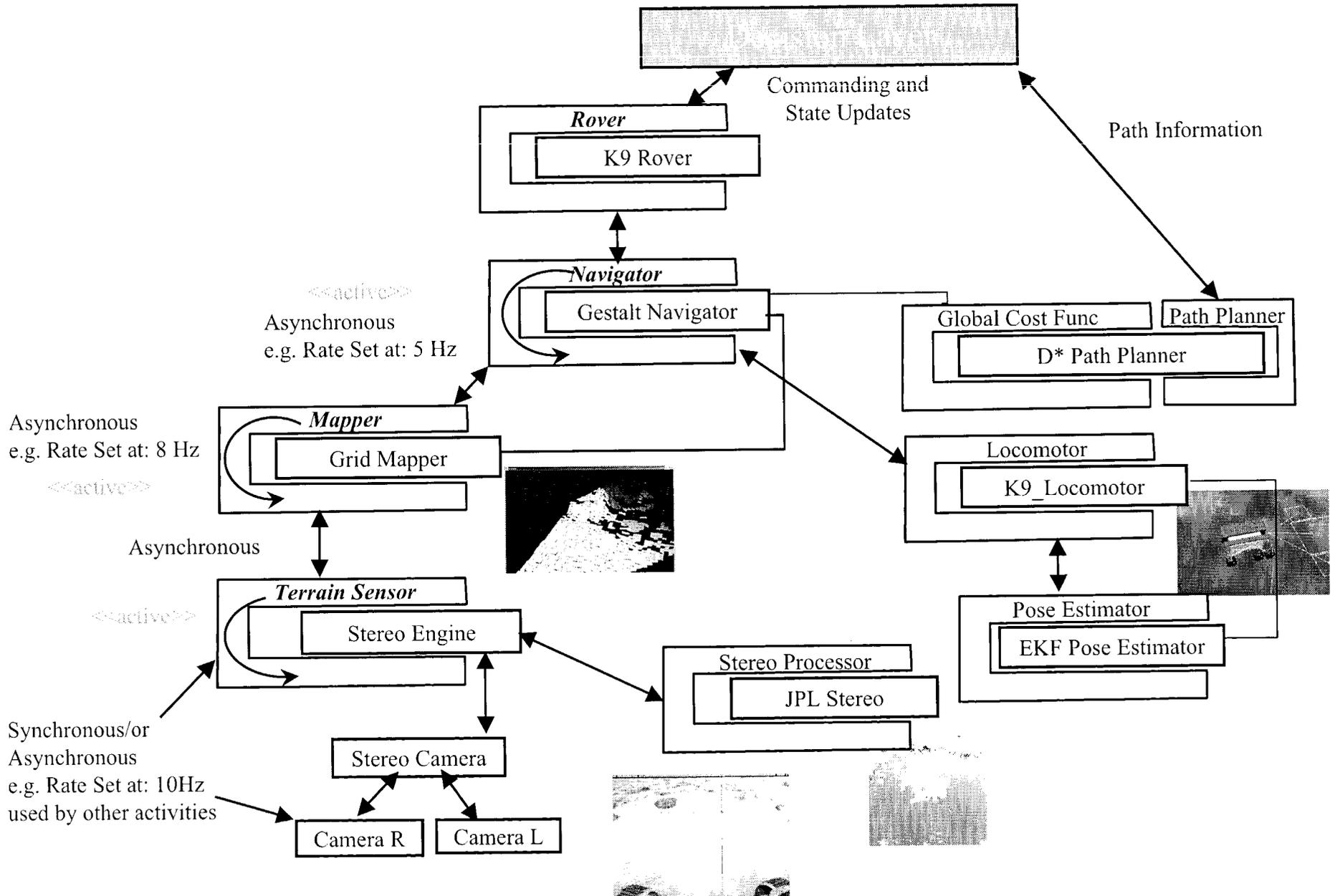
# Navigation Subsystem: *Top Level Abstractions*



# Architectural Traverse Example (1)



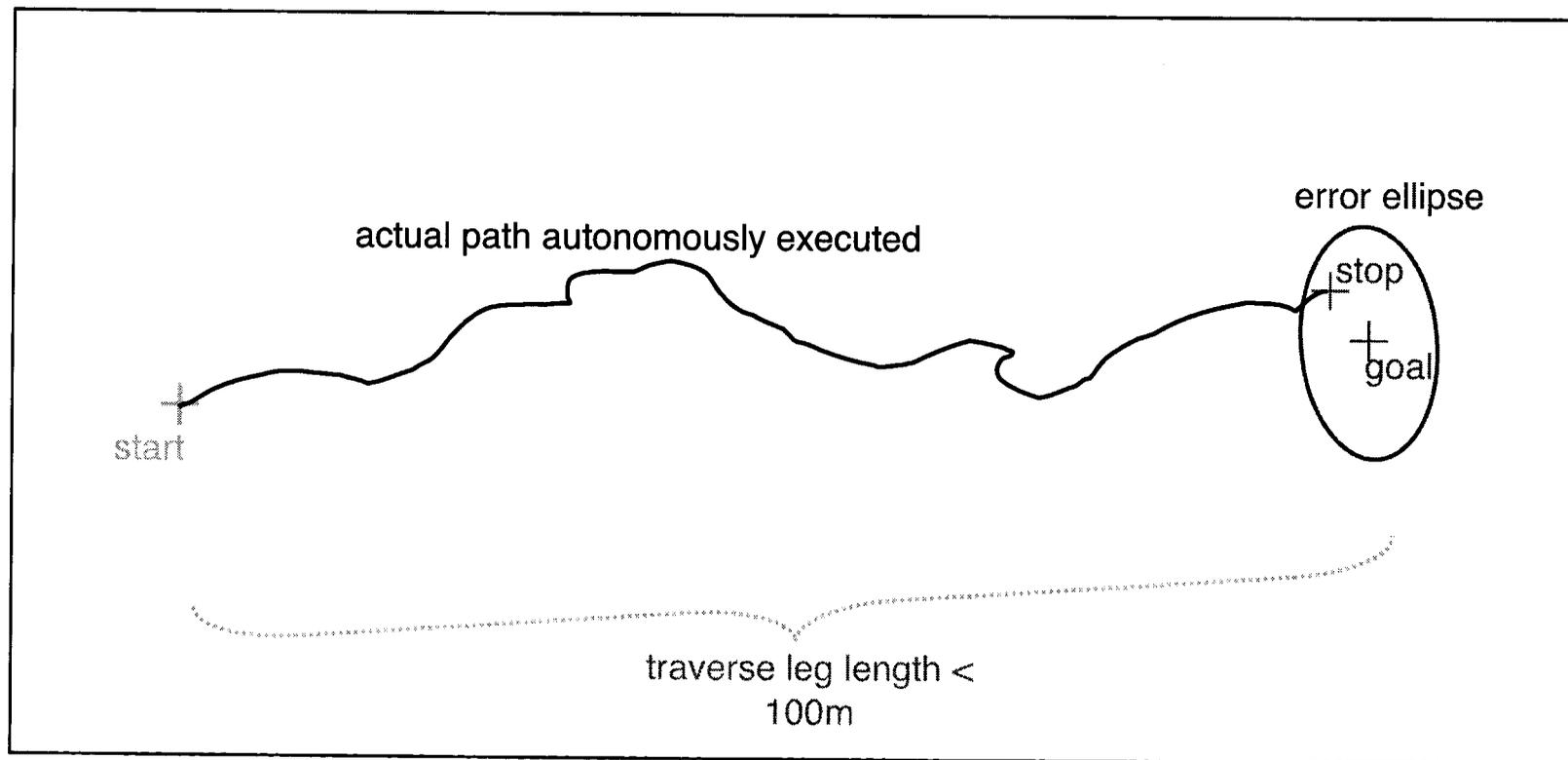
# Architectural Traverse Example (2)



# **Technology Validation Process for Mission Scenarios**

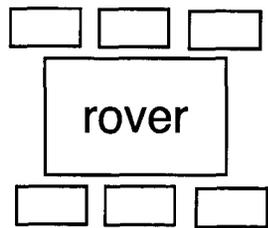
## ***'09 MSL Validation Scenario #1: Long Range Traverse***

**Description:** Enable autonomous traverse, obstacle avoidance, and position estimation providing up to 100m/sol with less than 3% error relative to starting position.



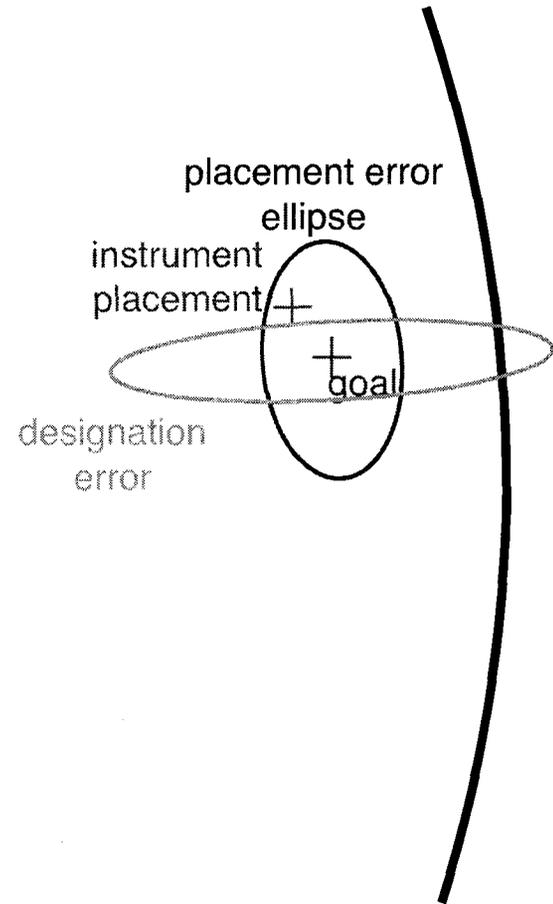
## **'09 MSL Validation Scenario #2: Approach & Instrument Placement**

**Description:** Enable placement of a science instrument on a designated target, specified in imagery taken from a stand-off distance. Placement accuracy to be within 1cm or 0.1%, from a stand off distance not greater than 10m.



partial panorama

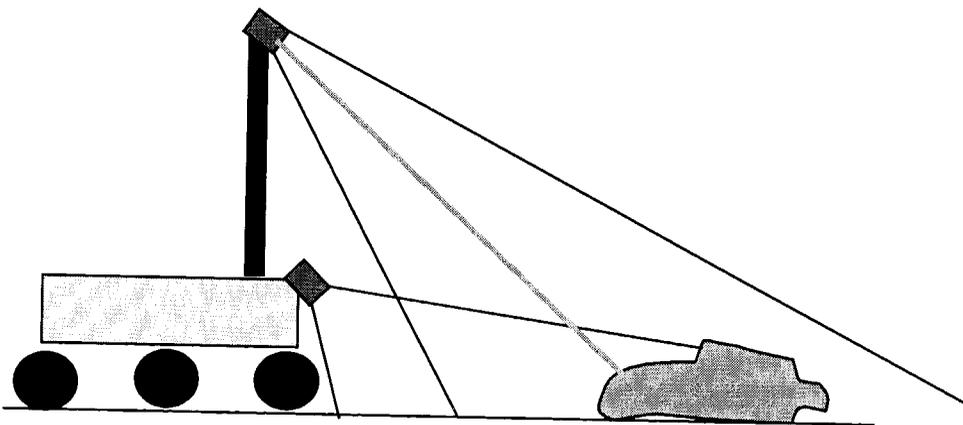
Max designation range < 10m



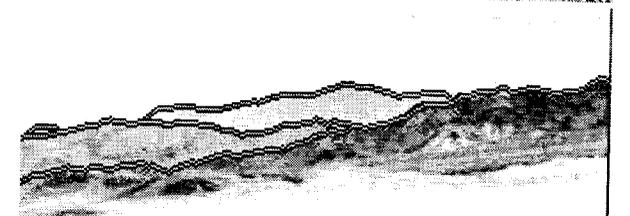
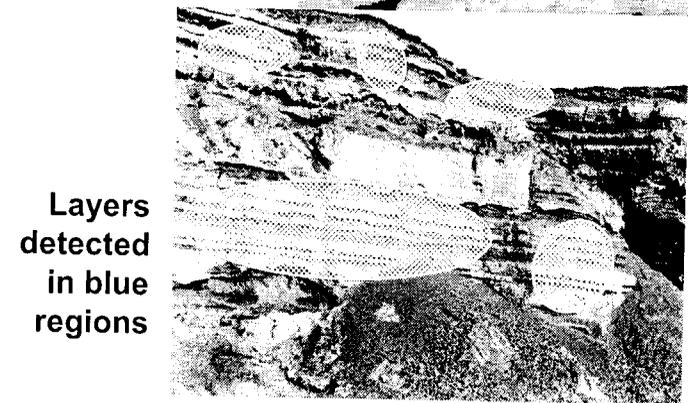
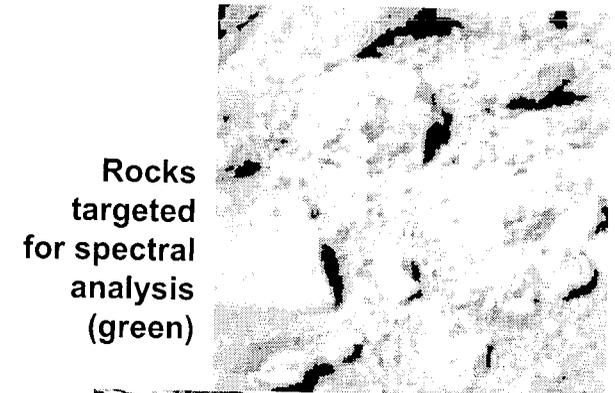
# '09 MSL Validation Scenario #3: Onboard Science Data Processing

**Description:** Enable processing of science data onboard the rover system. This will be used for the progressively more challenging issues of:

- intelligent data compression (inlier detection) and prioritization,
- anomaly recognition (outlier detection) with stop and communicate result,



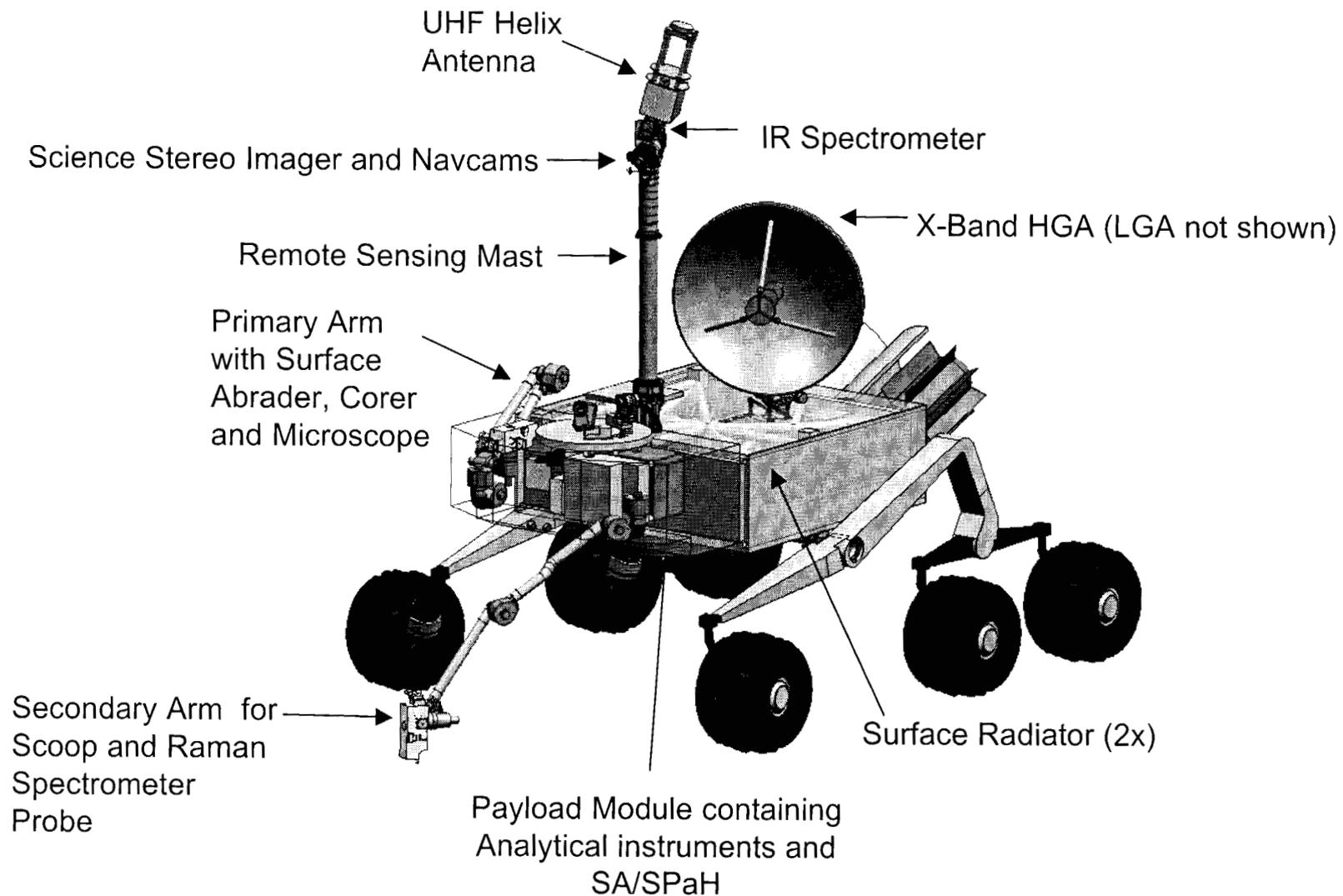
*Example: "Terrain classification while doing long traverse"*



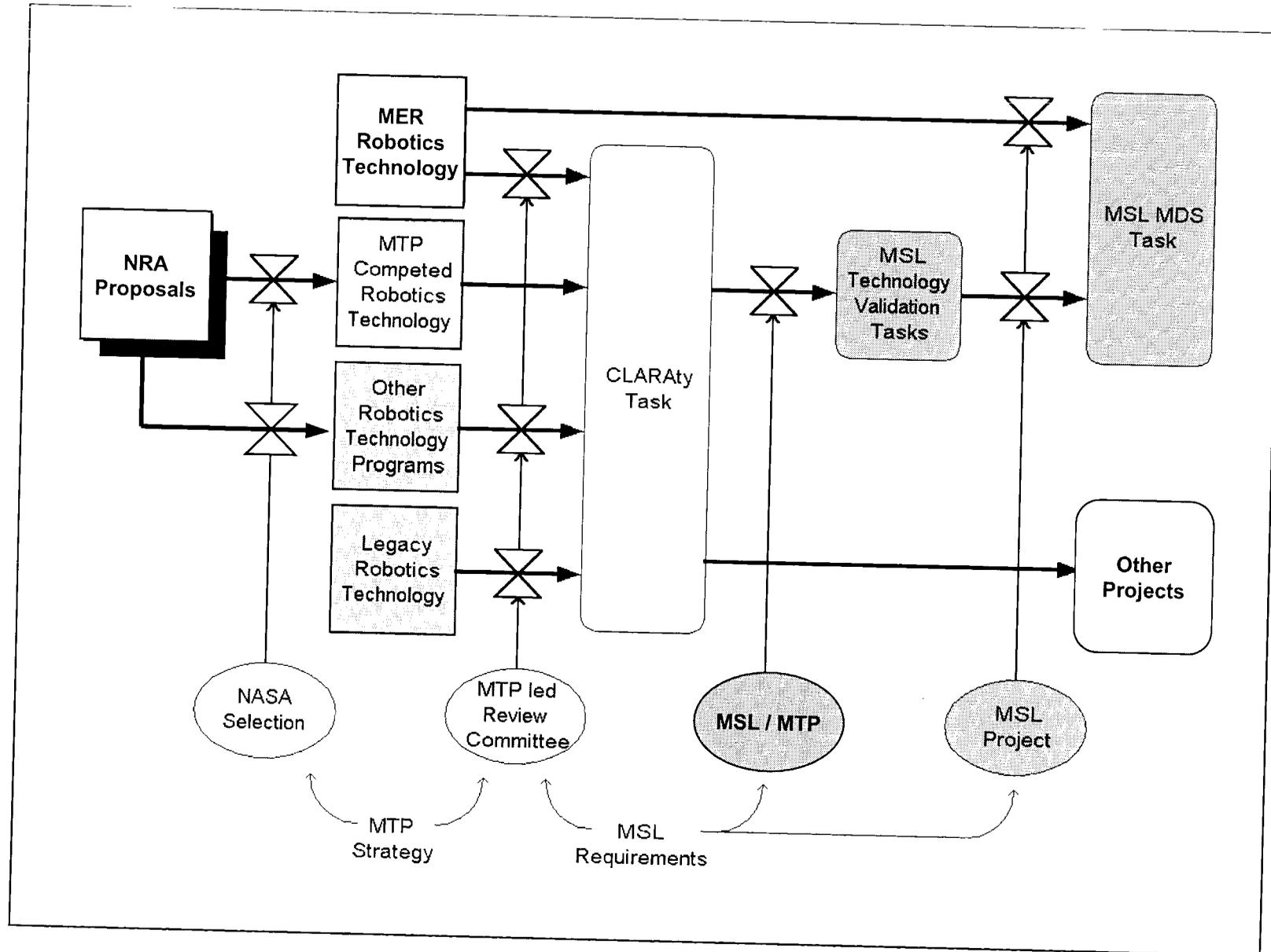
**Compound Horizon  
Determined (red)**

# **Mission Infusion**

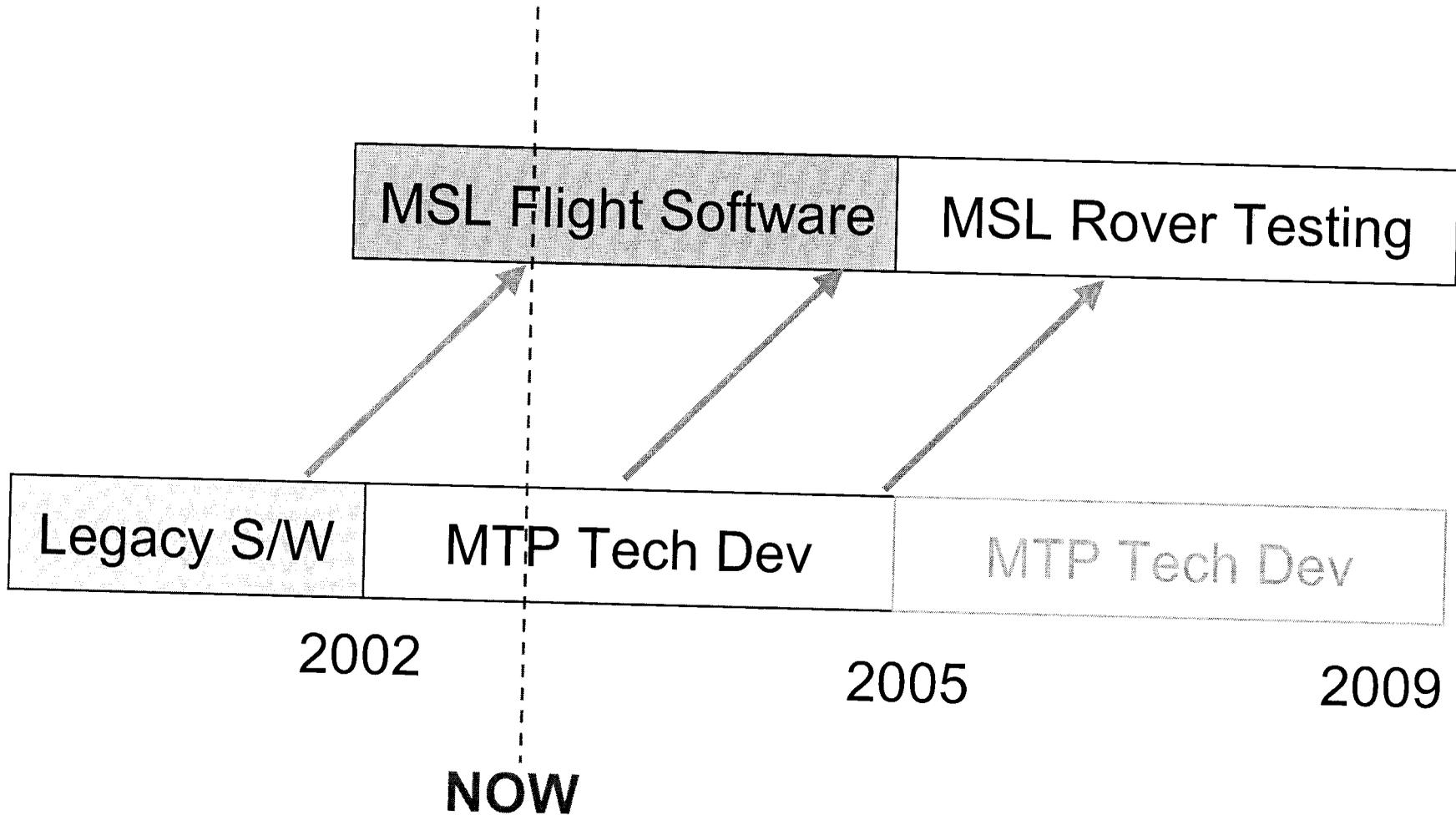
# 2009 Proposed Rover Configuration



# Technology Component Flow



# Overview of Software Flow in Time



## **Summary**

- **Previous point design of systems has yielded some solutions but lack critical mass amongst all NASA robotics efforts.**
- **Cross-leveraging and comparison of solutions requires extensive software infrastructure development, underway.**
- **A major product is validation for mission infusion.**
- **Another major product is building a legacy of robust, documented software and algorithms for future leveraging by technology development efforts.**