CLARAty: Coupled Layer Architecture for Robotic Autonomy

Issa A.D. Nesnas

Ames Research Center
Carnegie Mellon University
Jet Propulsion Laboratory

December 17, 2002

Mars Technology Program Technology Infusion Review

Presentation Overview

- Background and Overview
- Technology Support and Infusion
- Challenges of Interoperability
- Selected Examples
  - Motion Control
  - Locomotion
  - Navigation
- CLARAy Packages
- CLARAty testbed
- Summary

Background and Overview

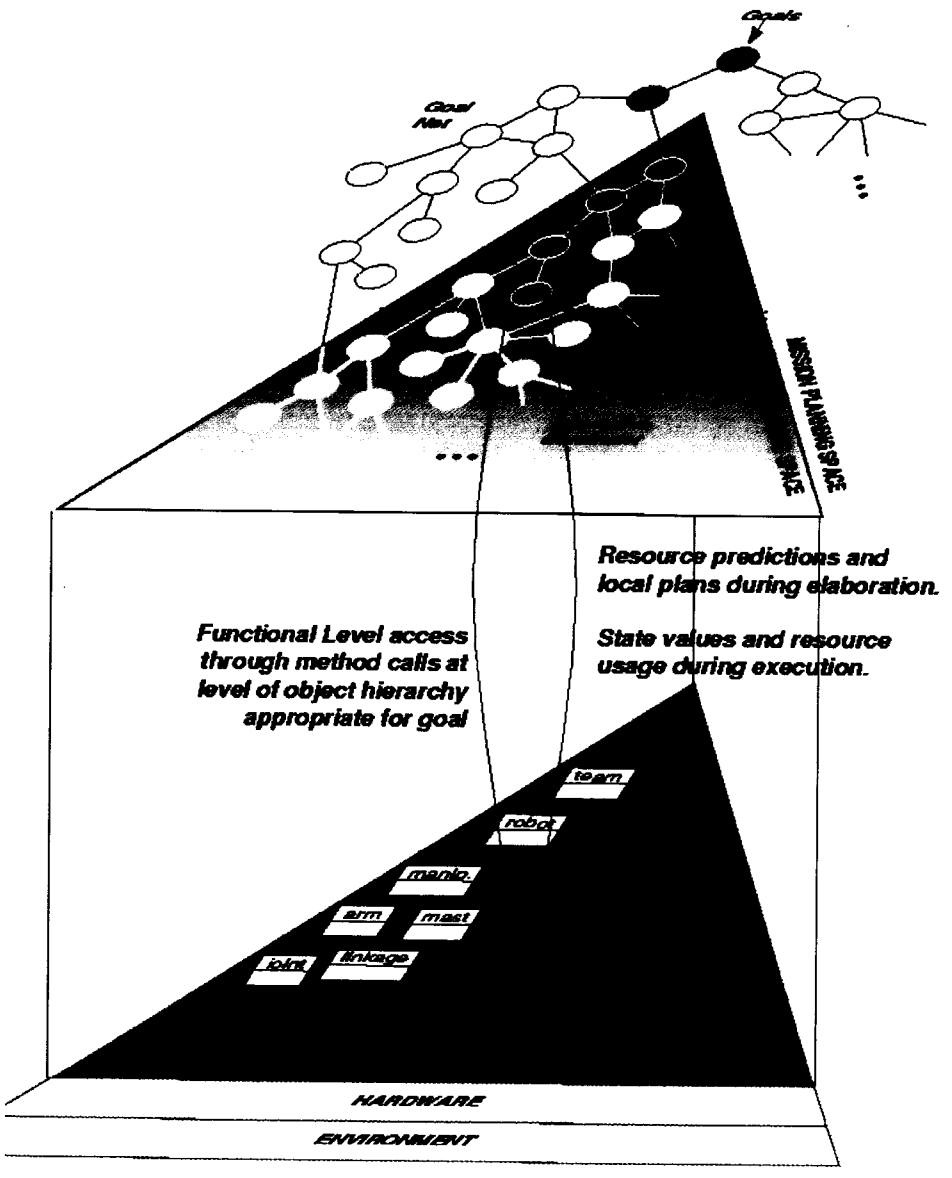
What is CLARAty?

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

A Two-Layered Architecture

CLARAty = Coupled Layer Architecture for Robotic Autonomy



THE DECISION LAYER:

Declarative model-based
Mission and system constraints
Global planning

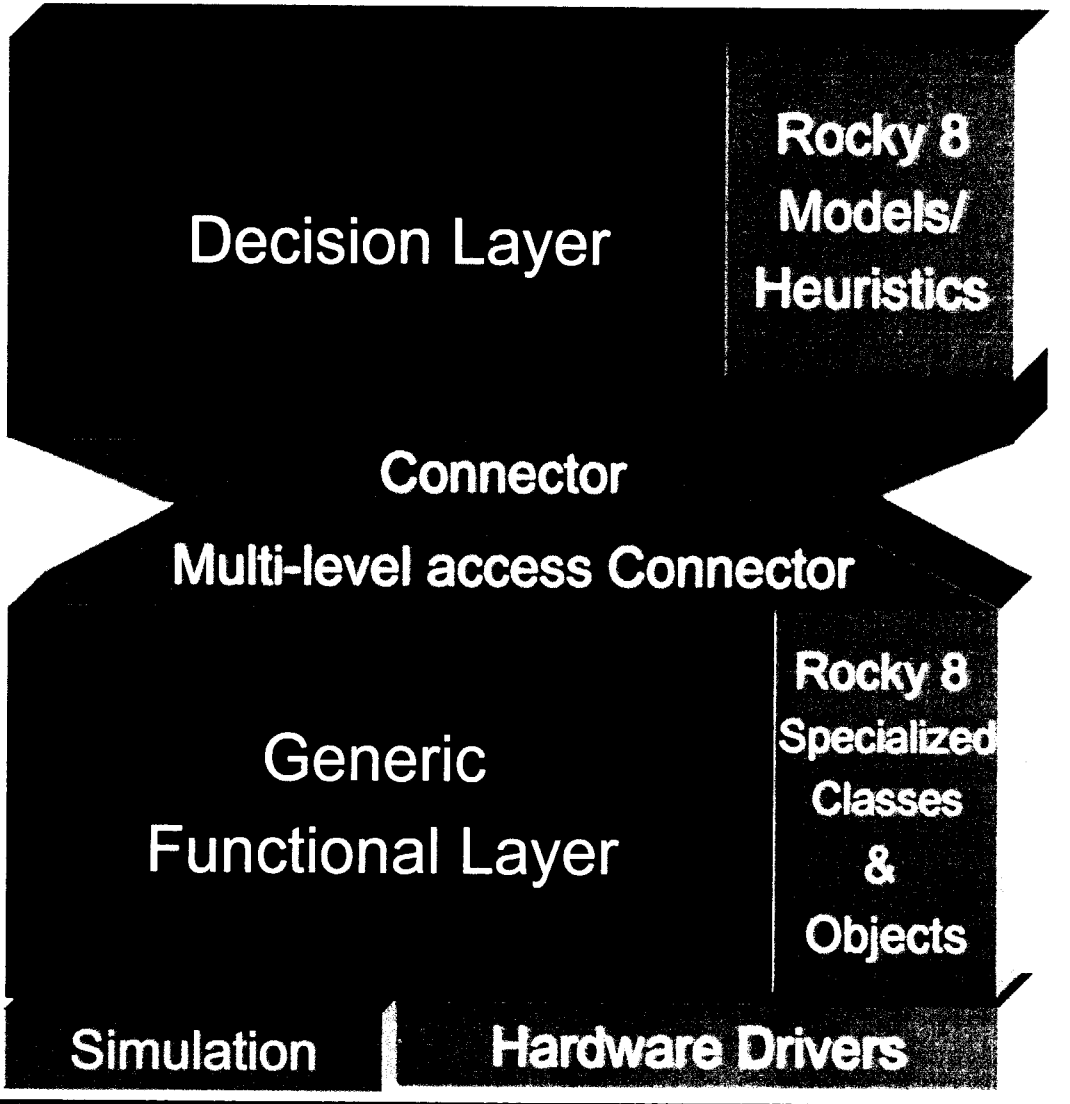
[redacted]

THE FUNCTIONAL LAYER:

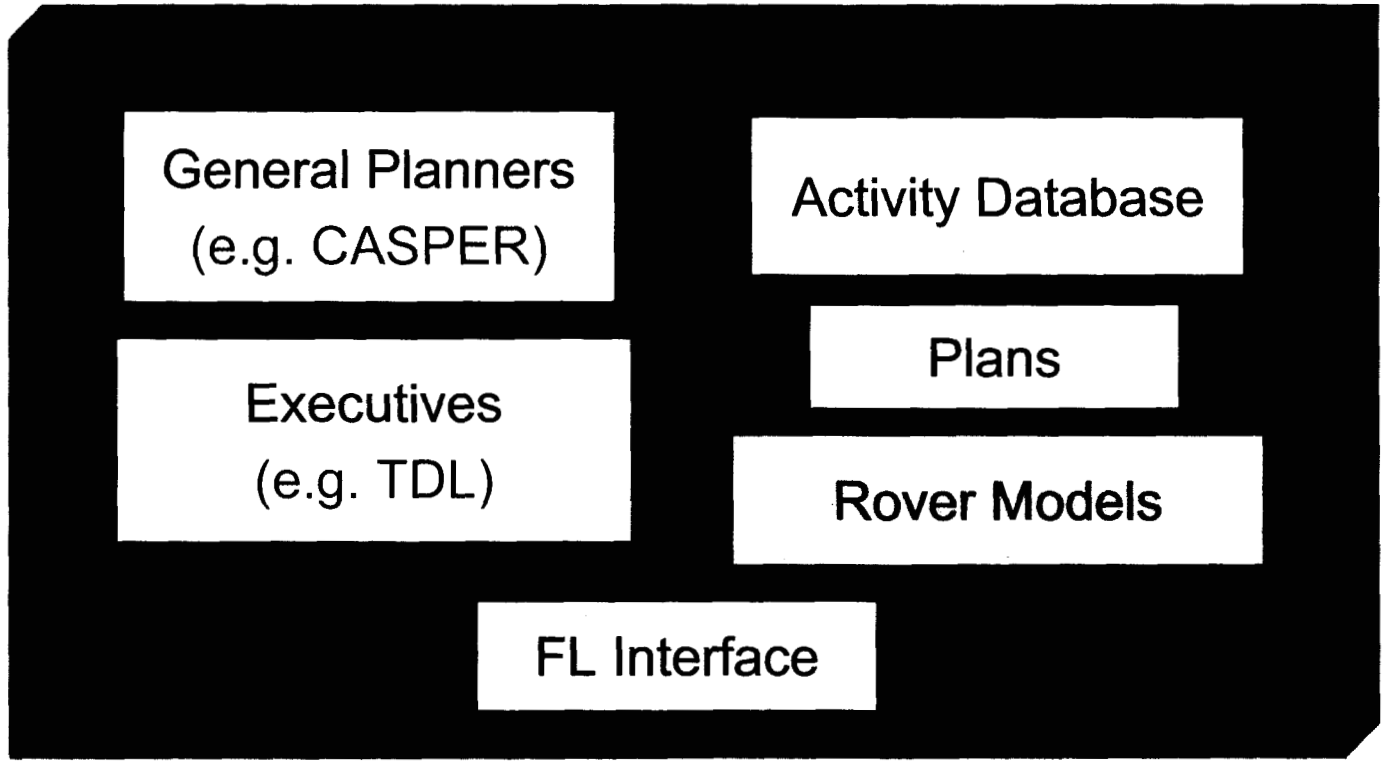
Object-oriented abstractions
Autonomous behavior
Basic system functionality

Adaptation to a system

Adapting to a Rover



The Decision Layer



The Functional Layer

	Rover		Adaptations
Navigation	Behaviors	Simulation	Rocky 8
Path Planning	Manipulation	Locomotion	FIDO
Estimation	Vision	Science	K9
Transforms	Math	Sensor	Rocky 7
Motion Control	Communication		
Input/Output	Hardware Drivers		

CLARAty Components

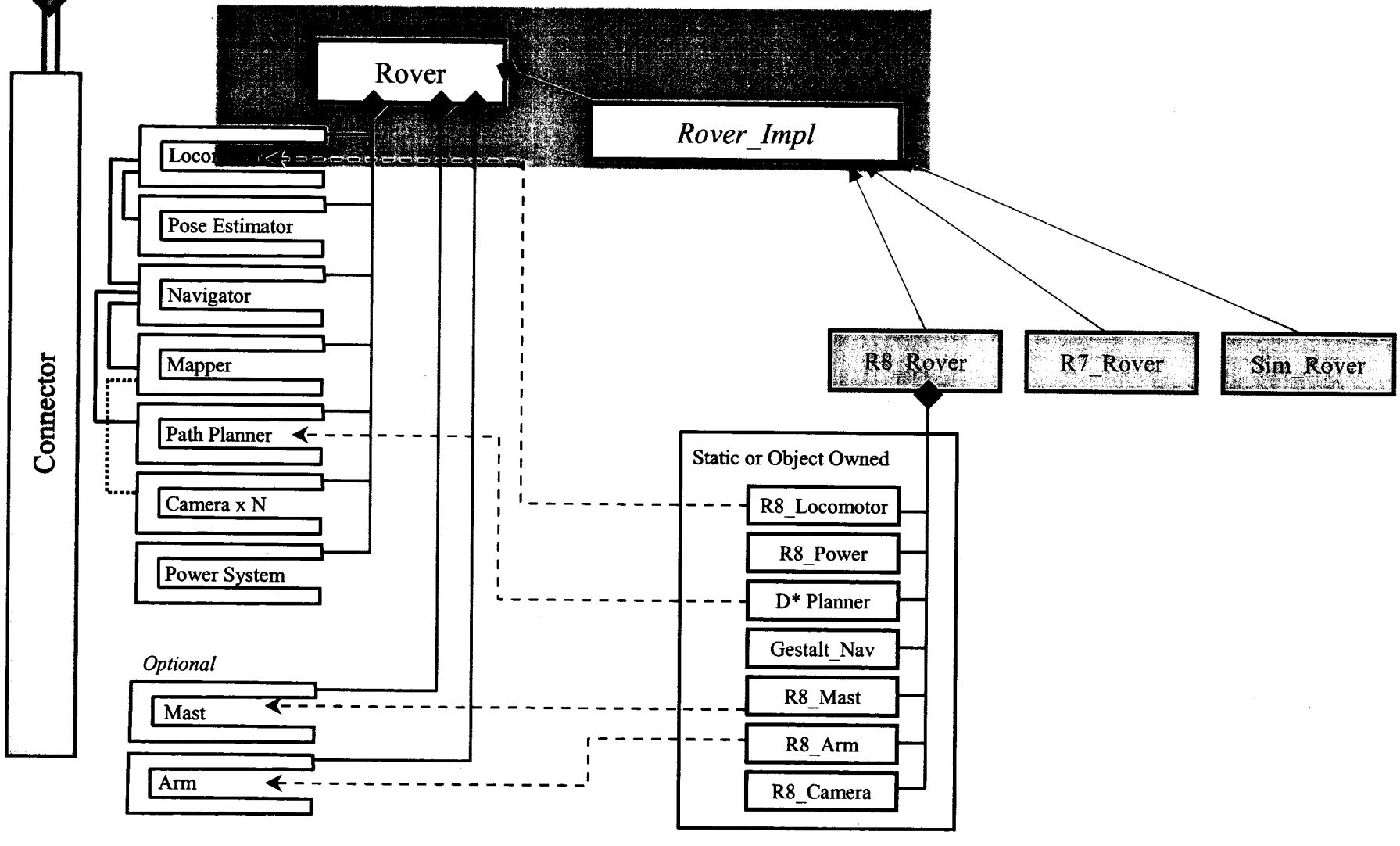
- Data Structure Components
  - Array, Vector, Matrix, Map, Container, LinkedList, Bit
  - Image, Message, Resource
- Generic Physical Components (GPC)
  - Locomotor, Arm, Mast, Spectrometer, …
- Specialized Physical Components (SPC)
  - K9_Locomotor, K9_Arm, R8_Mast, FIDO_Motor,…
- Generic Functional Components (GFC)
  - ObjectFinder, VisualNavigator, StereoProcessor, Localizer
- Specialized Functional Components (SFC)

Background and Overview

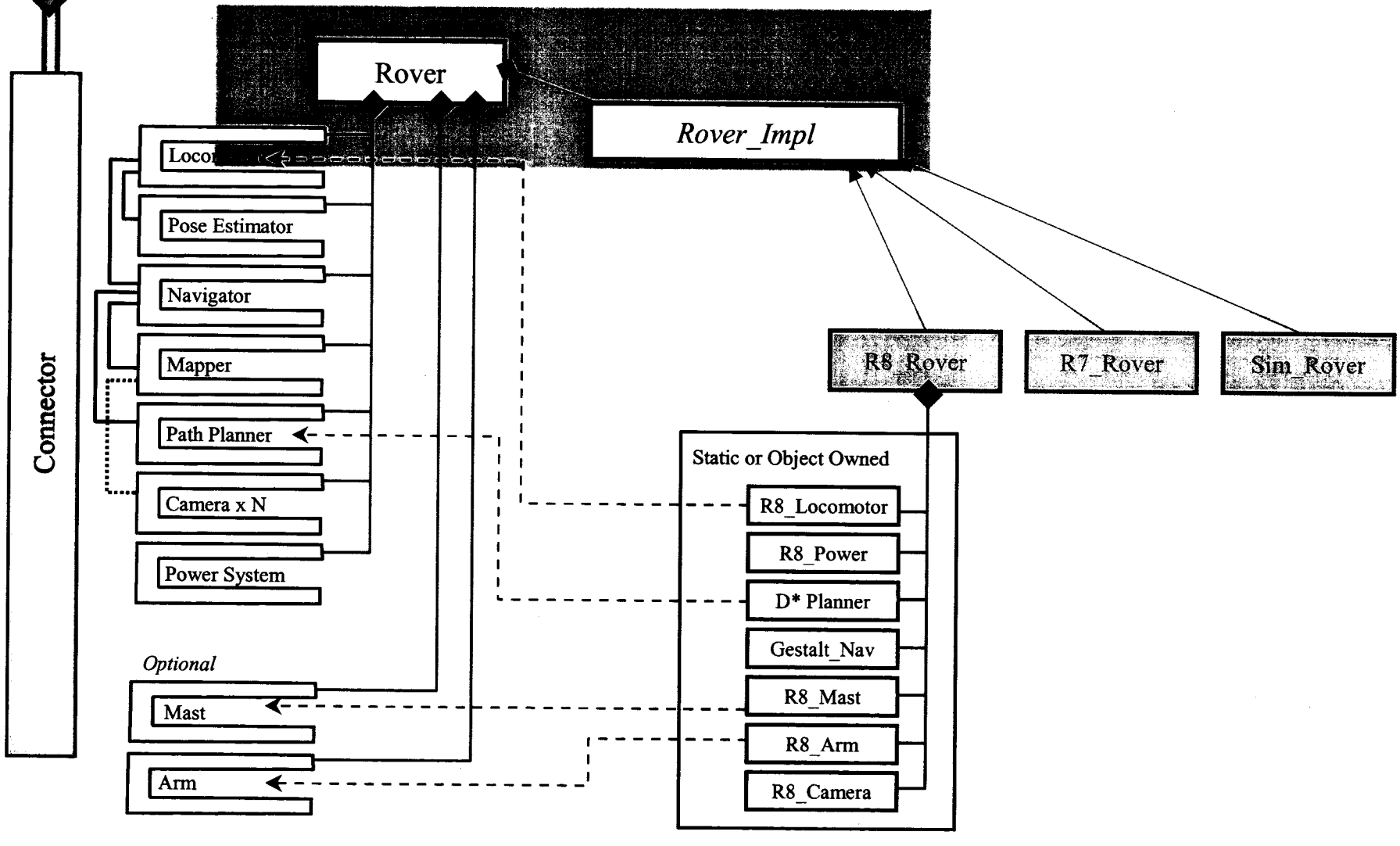
The Functional Layer

	Rover		Adaptations
Navigation	Behaviors	Simulation	Rocky 8
Path Planning	Manipulation	Locomotion	FIDO
Estimation	Vision	Science	K9
Transforms	Math	Sensor	Rocky 7
Motion Control	Communication		
Input/Output	Hardware Drivers		

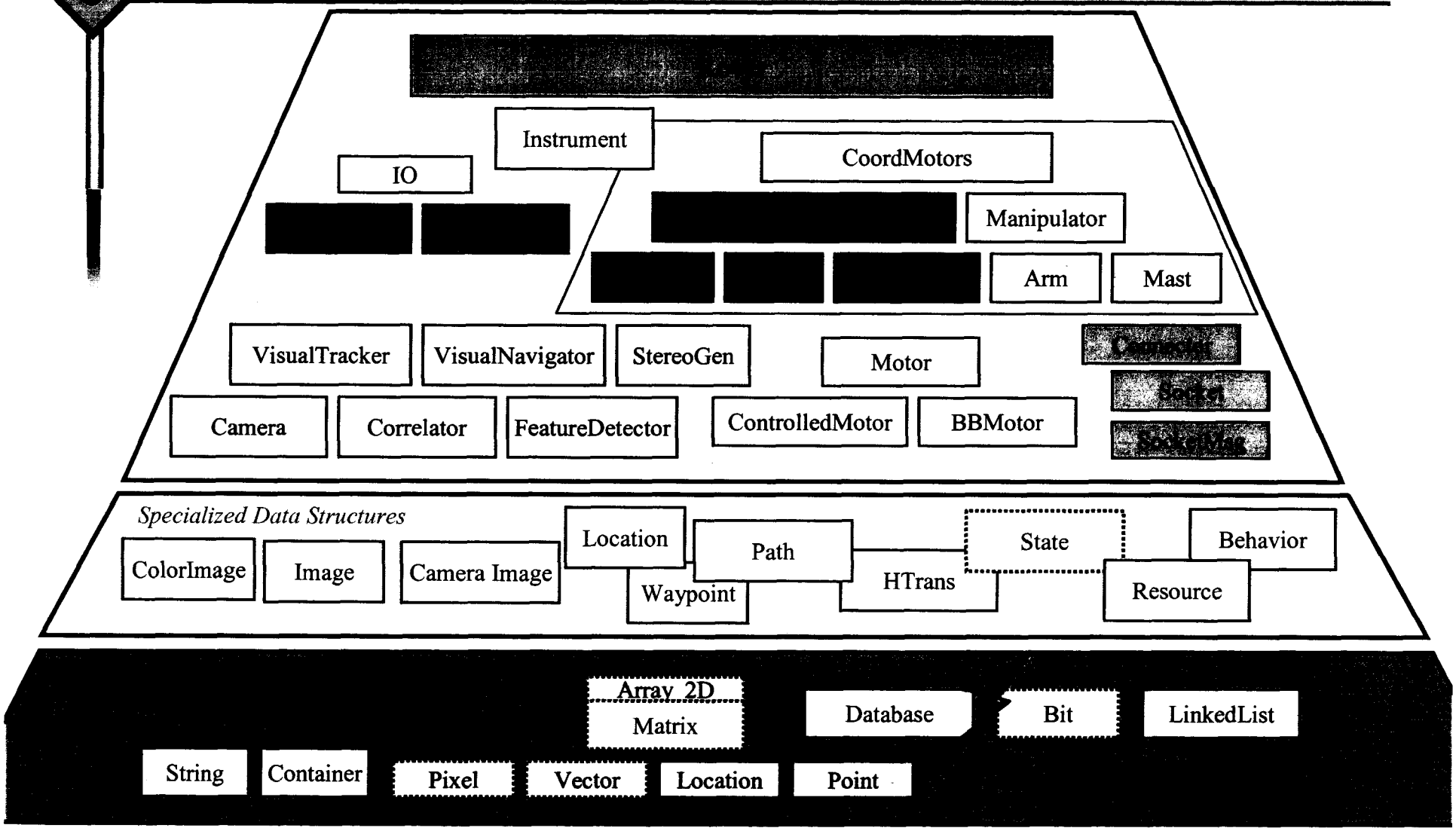
Top Level - Rover Abstraction in FL



Top Level - Rover Abstraction in FL



Functional Layer Components



Technology Support & Infusion

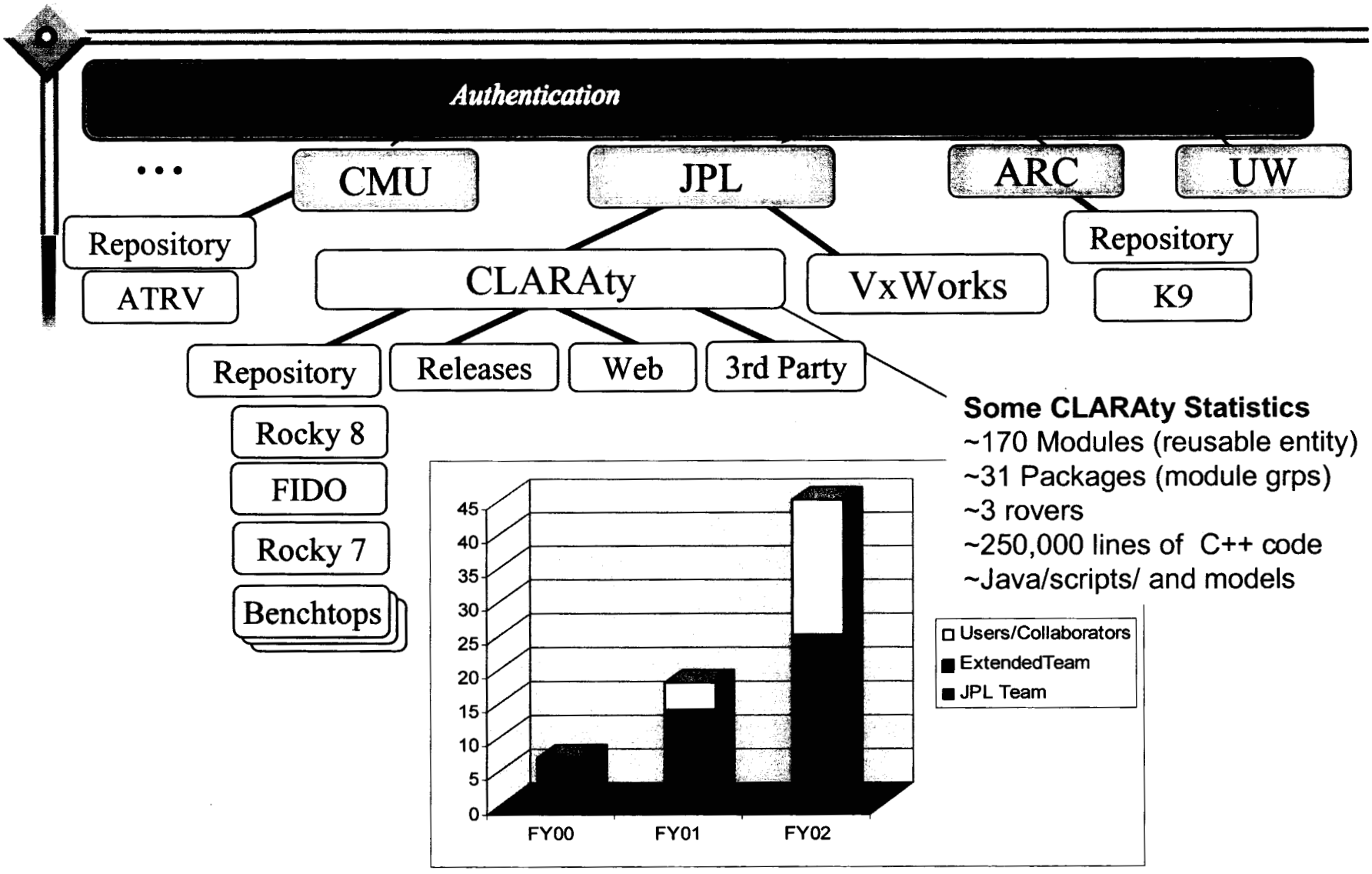
Generic Technologies & Algorithms

- Technologies that are generic by design should not be constrained by the software architecture & implementation
- Non-generic technologies should be accommodated on the appropriate platforms
  - Example (Generic): if you are working in navigation, you would not care about H/W architecture difference among different rovers
  - Example (Specific): if you are doing wheel/terrain interaction research, you might require specific hardware which one of the vehicles would support
- Assumptions are made explicit

Levels of Integrations

- Encapsulated
  - Uses CLARAty components to interact with rover
  - Does not support a CLARAty API
  - Runs on multiple platforms
- Encapsulated & Integrated:
  - Supports a CLARAty API
  - No standarized access to internals of code
- Refactored
  - Uses all CLARAty abstractions and components
  - Reusable code internals
  - Access to all data products/supports messaging

Software Development Process



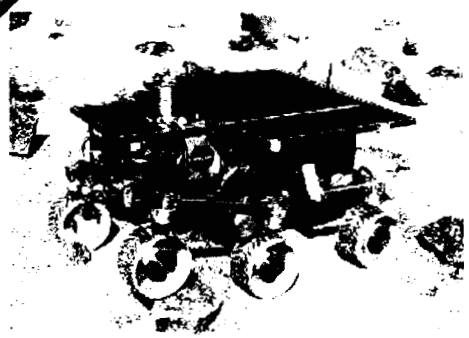
Infrastructure & Platform Support

- AFS backbone
- YaM for collaborative distributed environment
- ACE for OS independence
- Qt for Graphics Display
- Operating Systems & Architectures:

Operating System	Arch
VxWorks - Tornado 2.2 Beta	ppc
VxWorks - Tornado 2.02	x86
VxWorks - Tornado 2.02	sparc
Solaris 2.7	sparc
Linux - Red Hat	x86

Challenges in Interoperability

Currently Supported Platforms



Rocky 8

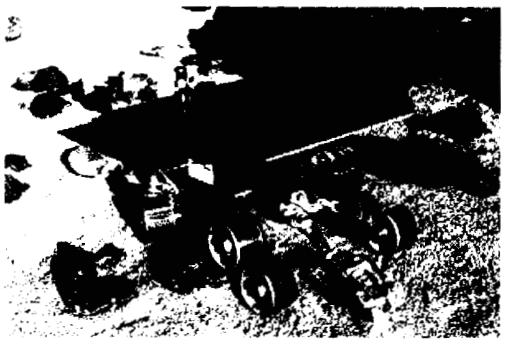
VxWorks

JPL

K9

Linux

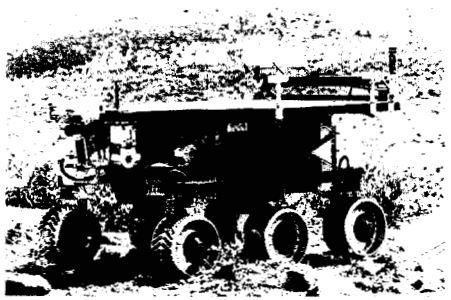
Ames



Rocky 7

VxWorks

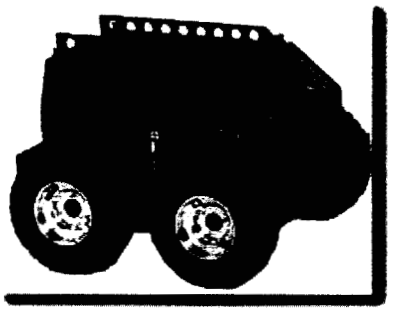
JPL



FIDO

VxWorks

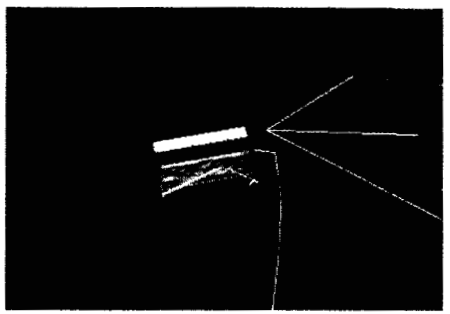
JPL



ATRV

Linux

CMU

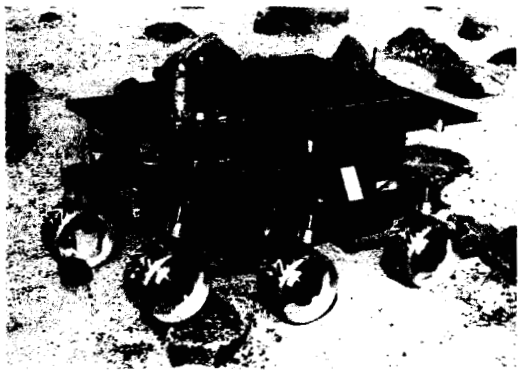


ROAMS

Solaris Linux

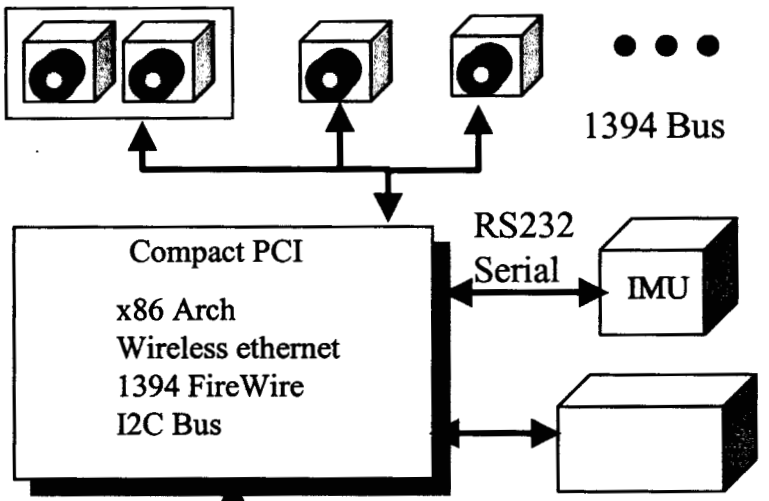
JPL

Distributed Hardware Architecture



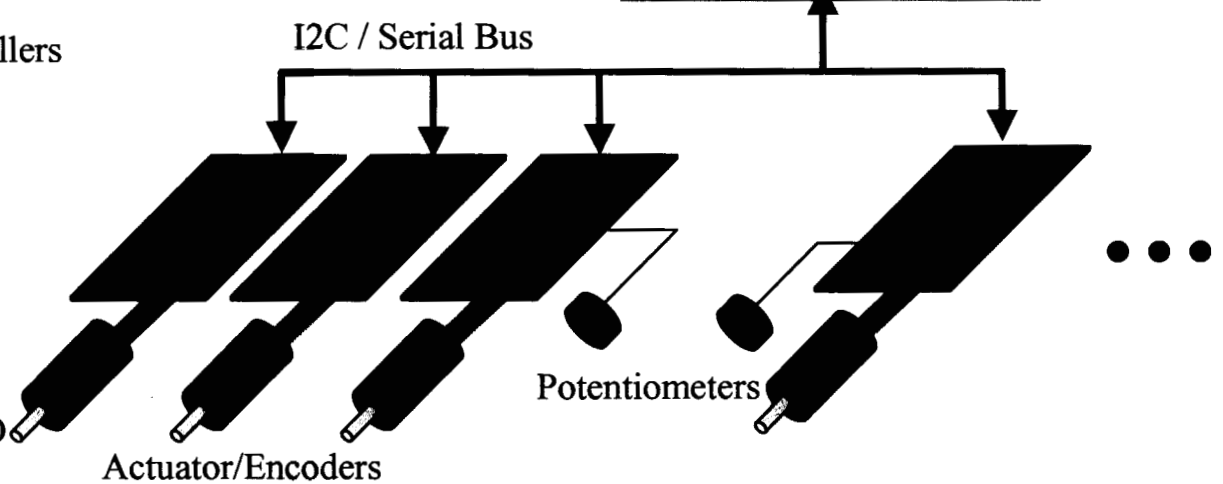
Rocky 8

K9



Rocky Widgets

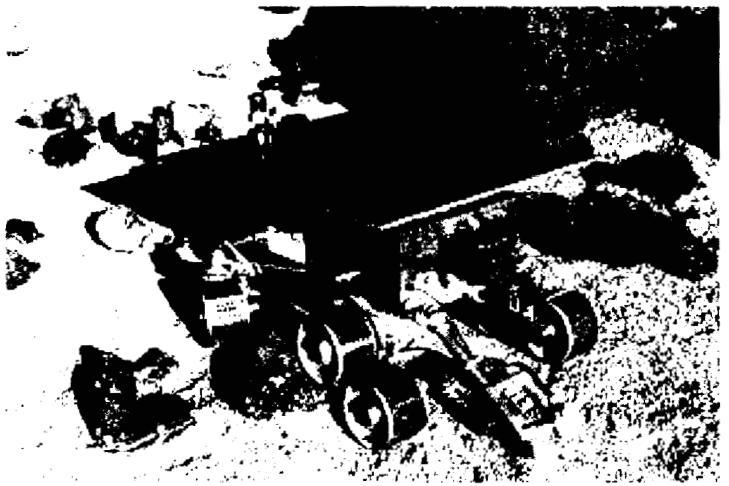
- Single Axis Controllers
- Current Sensing
- Digital I/O
- Analog I/O



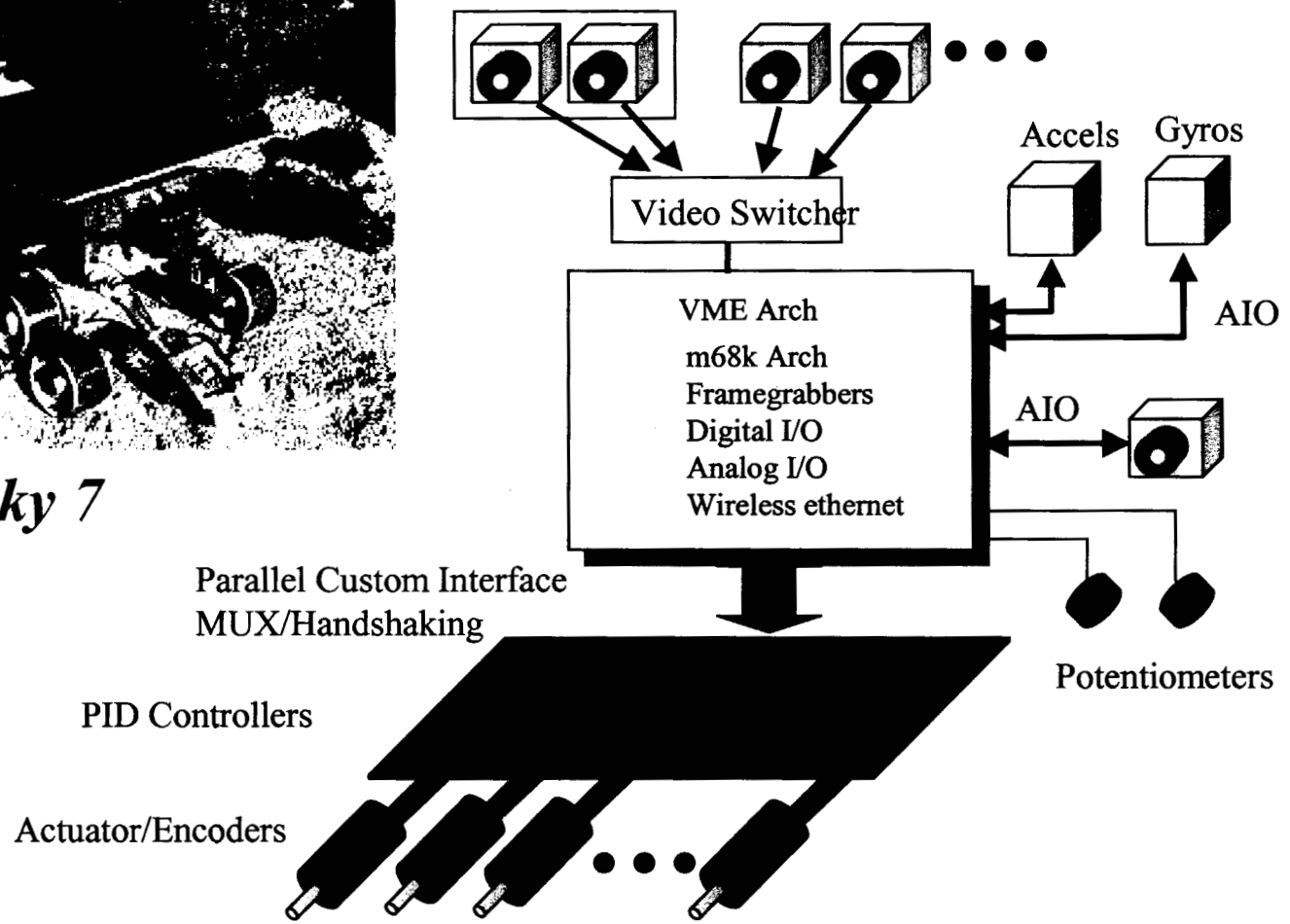
K9 Controllers

- PIC Servos
- Current Sensing
- Digital & Analog I/O

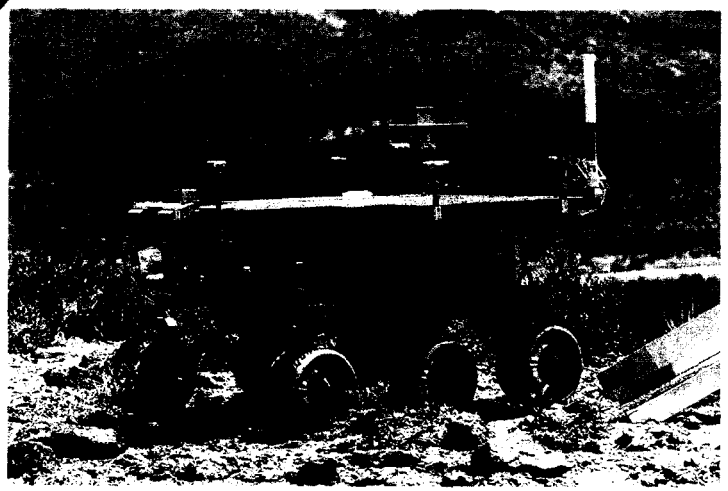
Custom Architecture/Variability



Rocky 7

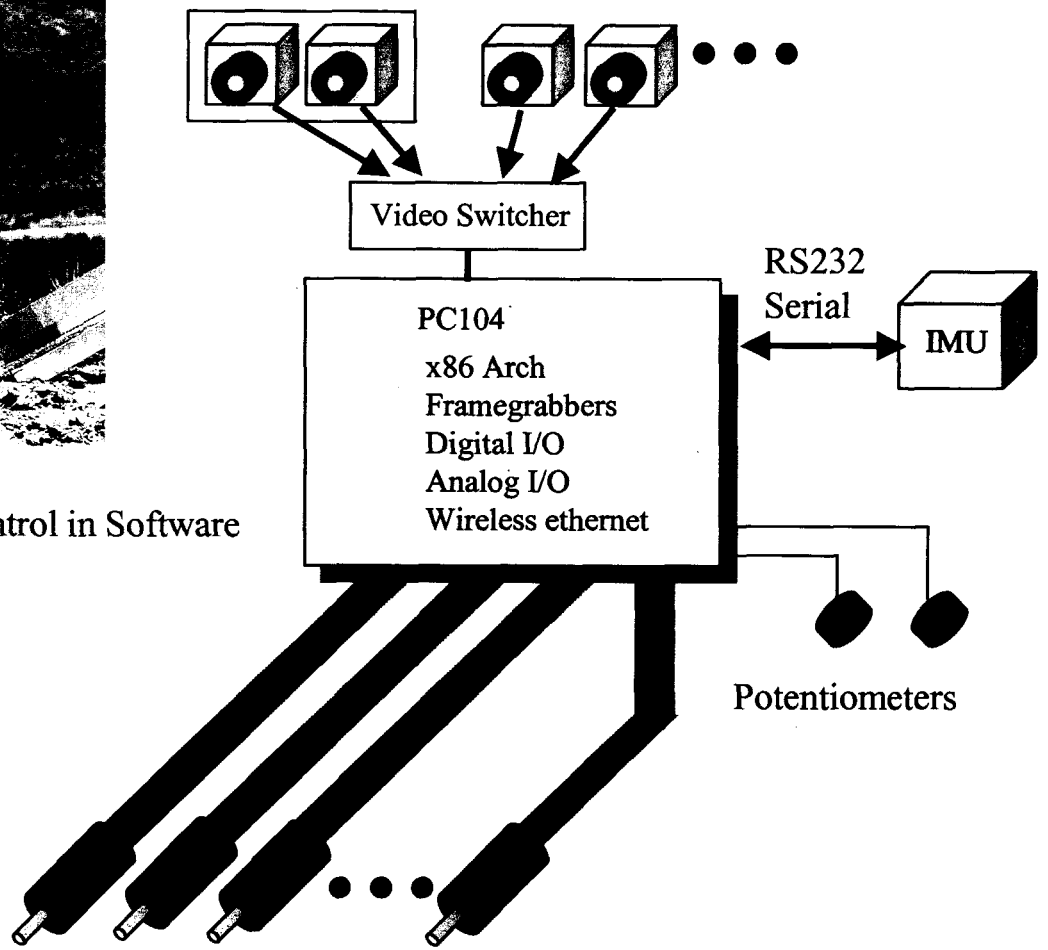


Centralized Hardware Mapped Architecture



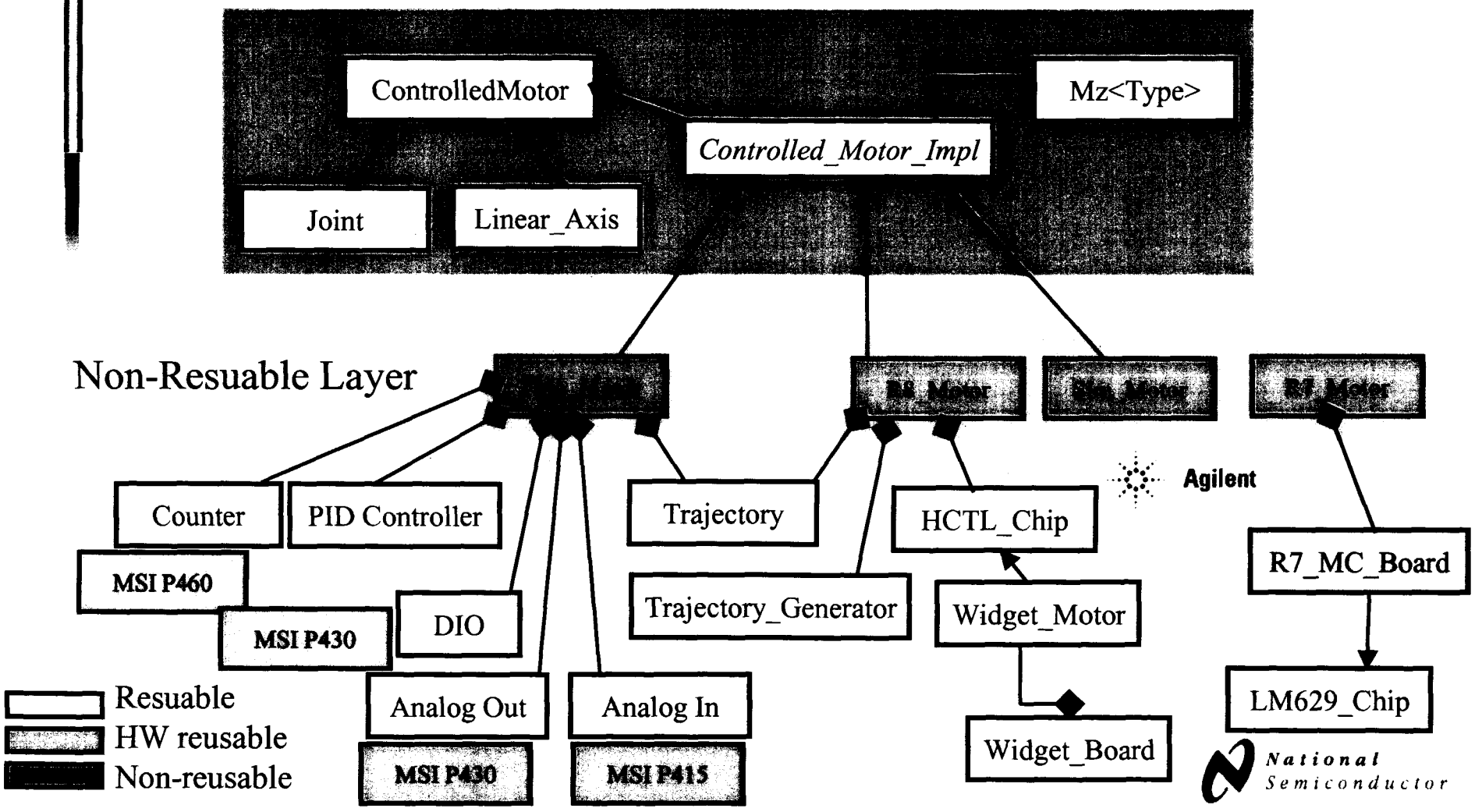
Fido

PID Control in Software



Selected Examples

Examples of CLARAty Reusability



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
Motor Controlller LM629	1,143	Reu
Digital I/O Board (S720)	576	Reu
PCI Components	329	Reu
Total	11,463	
Total Reusable	85%	
Total Reusable - Strict	67%	

Rocky 8 Modules	Lines of Code	
Controlled Motor	2,652	
Trajectory Generator	691	
Input Output	2,690	
Bits	1,580	
Resources (Timers, etc.)	725	
Bits	756	
Widget Board Software	2,126	
Motor Controller HCTL	900	Reusable - HCTL
I2C Master	1,165	Reusable - I2C
I2C Master Tracii	1,223	Reusable - Tracii
Total	16,314	
Total Reusable	[illegible]	
Total Reusable - Strict	[illegible]	

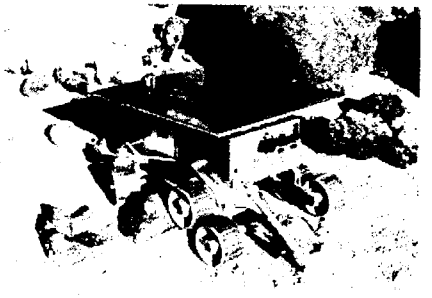
FIDO Modules	Lines of Code	Status
Controlled Motor	2,652	Reusable
Trajectory Generator	691	Reusable
PID Controller	997	Reusable
Input Output	2,690	Reusable
Bits	1,580	Reusable
Resources (Timers, etc)	725	Reusable
Common Definitions	2,380	Reusable
Encoder Counter (ISA P 400)	463	Reusable - H/W
Analog 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%	

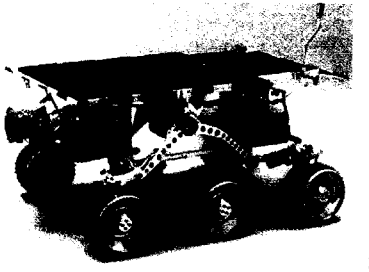
Capabilities of Wheel Locomotor

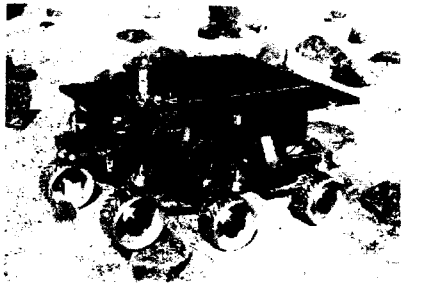
- Type of maneuvers:
  - Straight line motions (fwd / bkwd)
  - Crab maneuvers
  - Arc maneuvers
  - Arc crab maneuvers
  - Rotate-in-place maneuvers (arc turn r=0)
- Driving Operation
  - Non-blocking drive commands
  - Multi-threaded access to the Wheel_Locomotor class – e.g. one task can use Wheel_Locomotor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

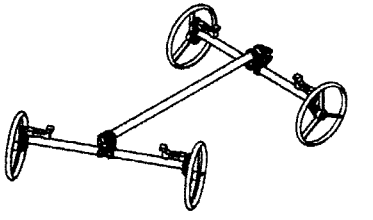
Generic Reusable Algorithms

- **Wheeled Locomotion – works for Rocky 8, Rocky 7, Fido, K9, ...**

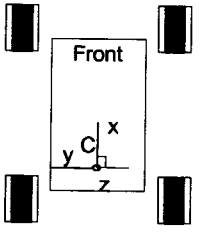




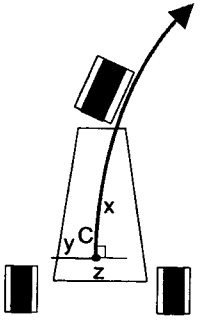




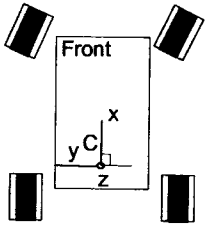
(f)
Steerable Axle
(e.g.Hyperion)



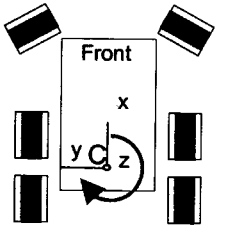
(a)
Skid Steering
(no steering wheels)



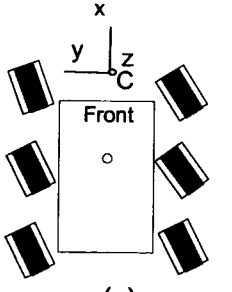
(b)
Tricycle
(one steering wheel)



(c)
Two –wheel steering



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



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

Example: collaborative development for locomotor

Version 1.0

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

JPL - 1998

Redesign/ mature

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

JPL - 2001

Redesign/ mature

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

CMU - 2002

Add

Version 4.0

- Use device and telemetry infrastructure
  Add adaptation to K9

Future

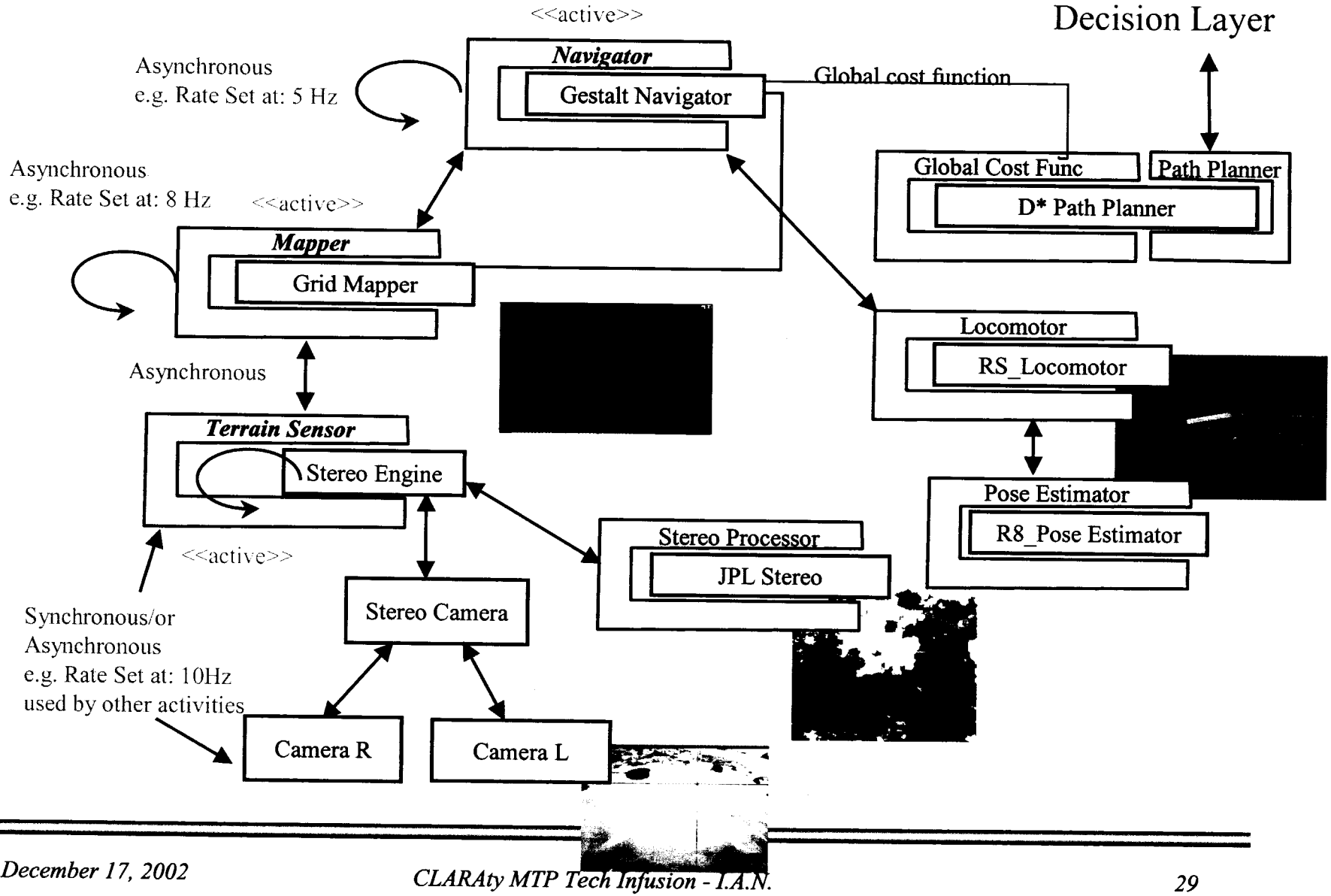
ARC - 2003

Code Resuability Results

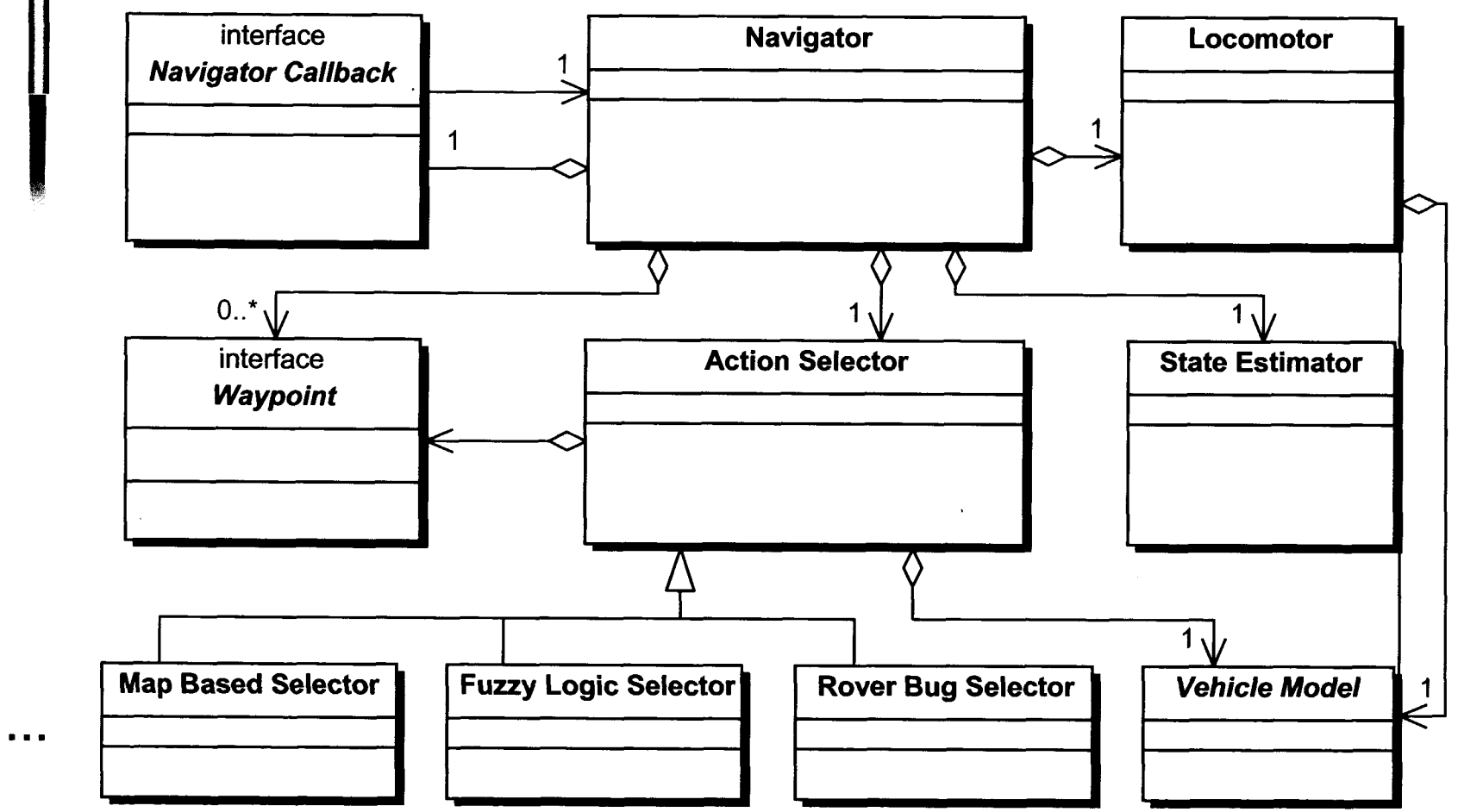
Analysis of amount of resuable code across implementations:

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	
Total	6995	584 (non-reusable)	
Total Reusable	**~92%**		

Navigation Architecture and Modularity

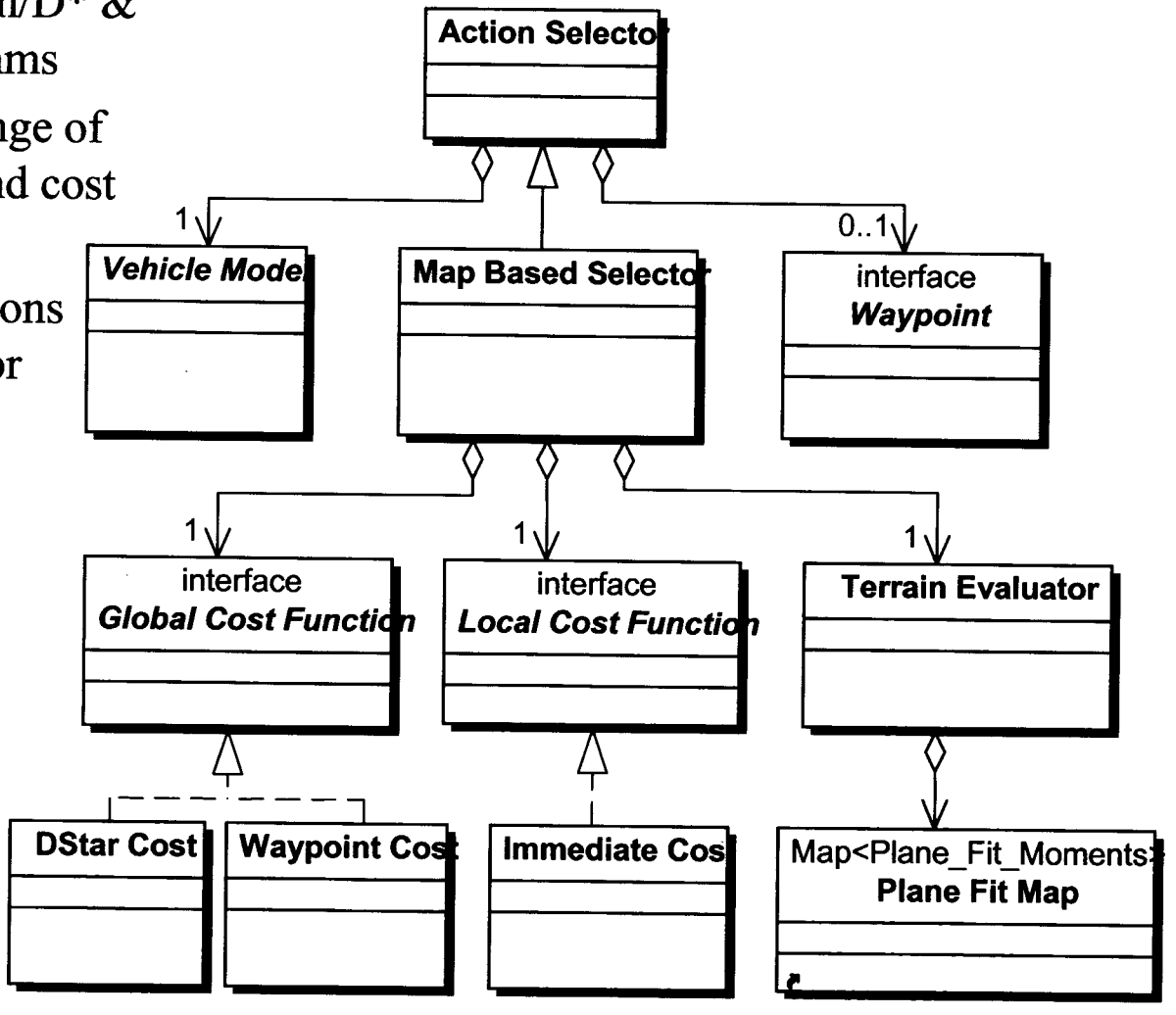


Navigation Architecture

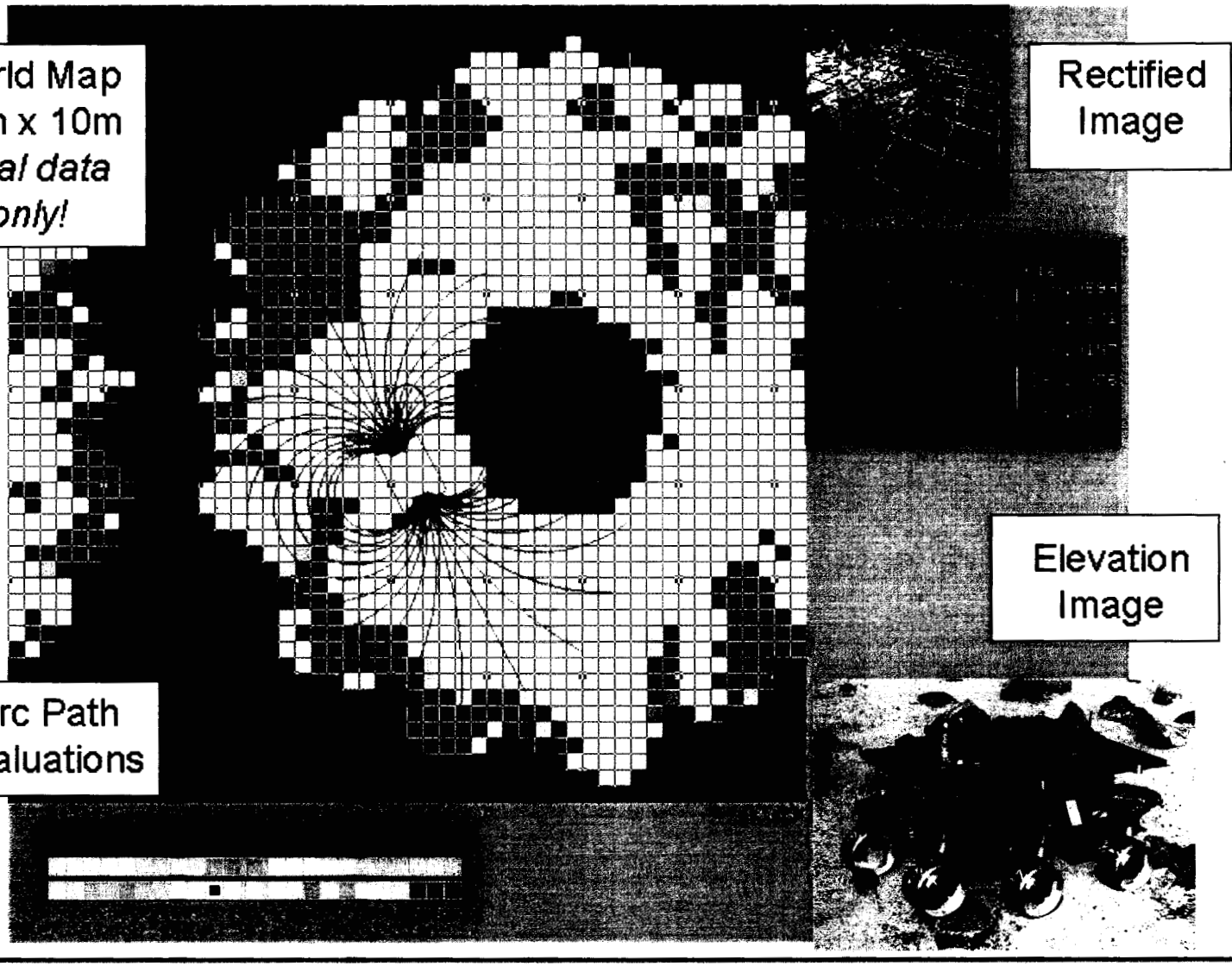


Map Based Selector

- Implements Morphin/D* & Gestalt type algorithms
- Enables easy exchange of terrain evaluation and cost functions
- Combine cost functions through arbitration or summation



GESTALT Navigator on Rocky 8



CLARAty Software Packages

Decision Layer Package

Decision Layer Package

General Planners
- Basic planner infrastructure

Activity Database
- Generic Activity database structure

Planner Implementations
- CASPER
- ASPEN

Executives
- TCM - Task Control Management used by TDL
- TDL - Task Description Language

Interfaces
- Functional Layer Interface
- ...

•••

Base Package

Base Package

Share
- Common Definitions
- Math Constants
- Function Operators
- Display Functions

Frame
- Coordinate Frames
- Locations (Poses)

Numeric Solvers
- Newton Rhapson
- Nedler-Mede

Array
- 1D Arrays
- 2D Arrays
- Sub Arrays

Point
- 2-D Points
- 3-D Points

GPC & GFC
- Device
- Telemetry
- Periodic Objects

Matrix
- 1D matrices (math)
- 2D matrices (math)
- Fixed size matrices
- Rotation matrices

Resource
- Timers
- Tasks

State
- State data logging
- Database

Transforms
- Homogeneous Transforms
- Quaternions

•••

Greyed Out – in progress
Orange – non-JPL/non-CLARAty development
Green – JPL-non-CLARAty development

Input Output Package

Input Output Package

I/O
- I/O
- Digital I/O
- Composite Digital I/O
- Analog Input
- Analog Output

Bits
- Bits data structure
- Bit manipulation/masking

Motion Control Package

Motion Control Package

Motor
- Open Loop Motors
- Controlled Motors

Trajectory
- Trajectories
- Trajectory Generators
- Continuous Profiling

Coordinated Systems
- Motor Coordinators

•••

Locomotion Package

Locomotion Package

Wheel Locomotion
- Wheels/Steerable Wheels
- Locomotor models
- Locomotor State
- Drive Commands
- Motion Sequences
- 2-D Wheel Locomotion algorithms
- 3-D Rocker Bogie Locomotion
- Continuous driving modules

Legged Locomotion
- Legged Locomotors

Hybrid Locomotion
- Wheel/Legged Locomotors

...

Vision Package

Vision Package

Camera
- Framegrabbers
- Cameras (single, multi)
- Stereo Pairs
- Camera models

Data Structures
- Images
- Image Pyramids
- Image I/O
- Color Images
- Camera Images
- Point Clouds
- Meshes

Stereo Vision
- StereoProcessor
- StereoEngine

Basic Image Processing
- 2-D Image processing
  - Filtering & smoothing
  - Edge detection
  - Blob detection
  - Corner feature detection

Visual Tracking
- 2-D Feature extraction
- 3-D Feature extraction
- Affine Tracking
- 2-D Tracking algorithms
- Shaped based tracking
- Combined 2-D & 3-D tracking
- Visual odometry

•••

Stereo Implementations
- JPL Stereo engine
- SRI Stereo engine (commercial)
- Ames Stereo engine

Estimation Package

Estimation Package

Data Structure
- Measurements
- System models
- Generic Kalman Filter

Kalman Filter Algorithms
- 3 DOF EKF pose estimation
- 6 DOF EKF pose estimation

Pose Estimators
- Wheel-based Pose Estimators
- Legged-based Pose Estimators

•••

Navigation Package

Navigation Package

Local Navigators
- Action Selectors
- Mapped-base Action Selector
- Waypoints
- Generic local navigator structure

Path Planner
- Generic path planner structure

Navigation Implementations
- MER/GESTALT Navigator
- Morphin (CMU) Navigator
- Fuzzy Navigator

•••

Manipulation Package

Manipulation Package

Manipulators
- Joints
- Links
- Generic Manipulators
- Generalized Fwd Kinematics
- Generalized Inverse Kinematics
- Limited DOF manipulators
- Redundant manipulators
- End effectors

Sensor-based Manipulation
- Stereovision based manipulation
- Vision/tactile manipulation

•••

Simulation Package

Simulation Package

Simulator Connectors

- Interfaces to ROAMS simulation
  - Rover Sim
  - Navigator Sim
  - Locomotor Sim
  - Motor Sim
  - I/O Sim
  - IMU Sim

Simulation Implementations

- ROAMS Rover Simulation (JPL)

•••

Sensor Package

Sensor Package

IMUs
- Generic inertial measurement unit infrastructure

Voltage & Current Sensors

Tilt Sensors
- Generic Electronic Tilt sensors
- Generic Accelerometers

Rate Sensors
- Generic Gyros structures

Temperature Sensors

•••

Science Package

Science Package

Spectrometers
- Generic Spectra classes
  - Calibrated Reflectance Spectrum
  - Uncalibrated Spectrum
- Generic Spectrometer classes

Analysis Algorithms
- Carbonate Analyzer
- Edge Layer detector

Science Database

Science Cameras
- Special science cameras
  (inherits for Cameras of Vision Package)

•••

Communication Package

Communication Package

Ethernet
- Socket Communication
  - TCP Comm
  - UDP Comm

Serial I/O
- Serial Comm

Messages
- Object serialization
- Socket messages

Connector
- Client/server comm.
- Executives

•••

Path Planning Package

Path Planning Package

Global Path Planners
- Generic path planner structure

Path Planner Implementations
- D* Path Planner (CMU)
- Tangent Graph Path Planner (JPL)
- ISE Path Planner (CMU)

•••

Behavior Package

Behavior Package

Behaviors
- Basic rover behaviors

Behavior Coordination
- Arbiters
- Behavior combinators

•••

Hardware Drivers

Hardware Drivers

Motor Controllers

- HCTL 1100 Motor Controller
- LM 629 Motor Controller
- •••

Framegrabbers/Vision

- Imagenation CX100 Framegrabber
- Imagenation PX610 Framegrabber
- 1394 Camera Driver
- •••

I/O Boards/Chips

- VPAR10 – Digital I/O Board
- Sensoray S720 Digital I/O Board
- VADC20 – Analog Input Board
- MAX186 Analog to Digital Conv.
- MAX528 Digital to Analog Conv.

Custom Boards

- Widget Board – Motion Control
- Widget Board Firmware
- •••

Other

- VISA – Vme to ISA Converter
- I2C Master Controller
- PCI Controller
- Crossbow DMU – Interial Measurement Unit
- •••

Rocky 8 Package

Rocky 8 Package

Rocky 8 Specialized Physical Components

- Rocky 8 Motors
- Rocky 8 Cameras
- Rocky 8 IMUs
- Rocky 8 Hardware Map
- Rocky 8 Digital I/O
- Rocky 8 Analog I/O

Rocky 8 Specialized Functional Components

- R8 Locomotor
- R8 Contrained Locomotor
- R8 Navigator
- R8 Rover

•••

K9 Package

K9 Package

K9 Specialized Physical Components

- K9 Motors
- K9 Cameras
- K9 IMUs
- K9 Hardware Map
- K9 Digital I/O
- K9 Analog I/O
- K9 Locomotor
- K9 Contrained Locomotor
- K9 Rover

K9 Specialized Functional Components

- K9 Navigator

•••

Rocky 7 Package

Rocky 7 Package

Rocky 7 Specialized Physical Components

- Rocky 7 Motors
- Rocky 7 Cameras
- Rocky 7 Accels
- Rocky 7 Gyros
- Rocky 7 Hardware Map
- Rocky 7 Digital I/O
- Rocky 7 Analog I/O

Rocky 7 Specialized Functional Components

- R7 Locomotor
- R7 Contrained Locomotor
- R7 Navigator
- R7 Rover

•••

FIDO Package

FIDO Package

FIDO Specialized Physical Components

- FIDO Motors
- FIDO Cameras
- FIDO Accels
- FIDO Gyros
- FIDO Hardware Map
- FIDO Digital I/O
- FIDO Analog I/O

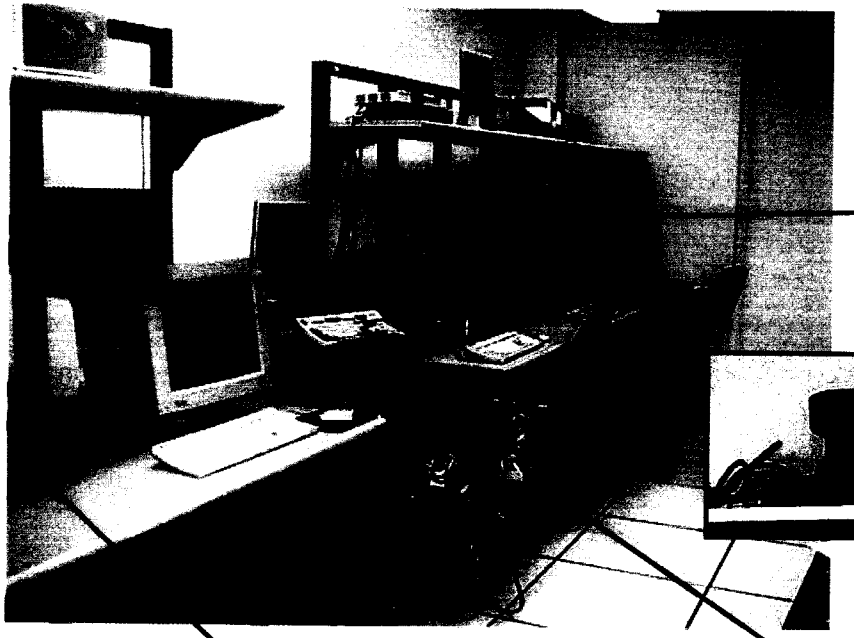
FIDO Specialized Functional Components

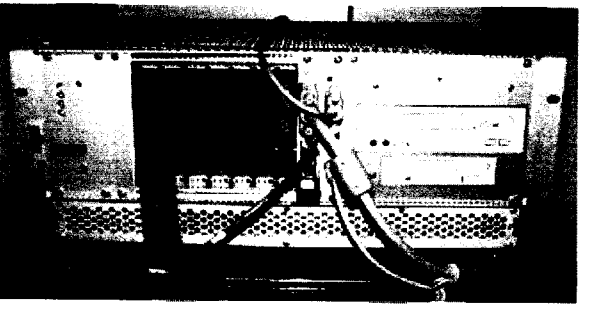
- FIDO Locomotor
- FIDO Contrained Locomotor
- FIDO Navigator
- FIDO Rover

•••

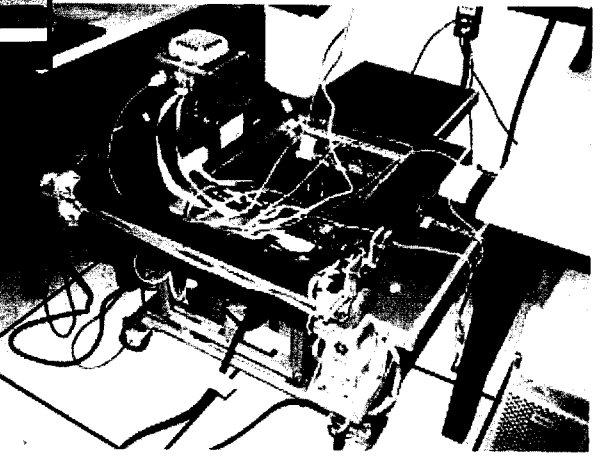
CLARAty Testbed

CLARAty Testbed (1/2)

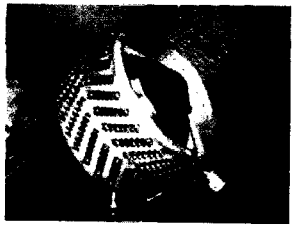


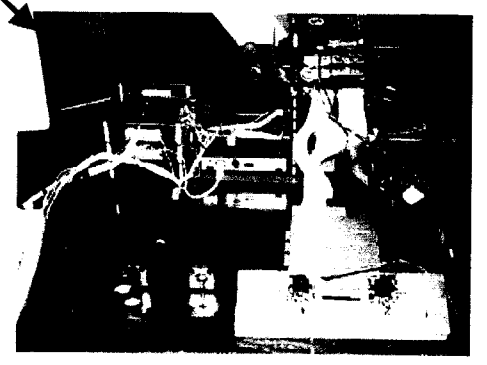


Rocky 8 Benchtop



Dexter Manipulators
Rocky 7 Benchtop





FIDO Benchtop

CLARAty Team

- **NASA Ames Research Center**
  - Maria Bualat
  - Clay Kunz
  - Randy Sargent
  - Anne Wright (lead)
- **Carnegie Mellon University**
  - Kam Lasater
  - Reid Simmons (lead)
  - Chris Urmson
  - David Wettergreen
  - Andy Yang
- **Jet Propulsion Laboratory**
  CLARAty on FIDO Team
  - Hrand Aghazarian (35/34)
  - Terrance Huntsberger (35/34)
  - Chris Leger (35/34)
  - Matthew Robinson (35/34)

CLARAty/Rocky8/Rocky7

- Max Bajracharya (34)
- Edward Barlow (34)
- Caroline Chouinard (36)
- Gene Chalfant (34)
- Hari Das (34)
- Tara Estlin (36)
- Dan Gaines (36)
- Mehran Gangianpour (34)
- Dan Helmick (34)
- Won Soo Kim (34)
- Michael Mossey (31)
- Issa A.D. Nesnas (34) (Task Manager)
- Ashitey Trebi-Ollennu (35/34)
- Richard Petras (34) (Rocky lead)
- Stergios Roumeliotis (Univ. of Minnesota)
- Kevin Watson (34)

Five Summer Students

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

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

Thank you for your Attention

Background & Objectives

Why are we doing this work?

- To capture and preserve robotics expertise from JPL and other centers
- To provide a framework for future NASA rover technology development and integration
- To reduce the cost of integrating new technologies
- To operate various robots from a unified framework
- To eliminate cost of redeveloping basic robotic infrastructure

Goals

- Capture and integrate a wide range of technologies
- Leverage existing tools
- Leverage experience and tools of the larger software development community
- Apply appropriate design patterns to the domain
- Provide an infrastructure that enables rapid robotic development
- Capture experience of technologists implementations

How?

- Study software implementation of various robots
- Study interactions of elements in various systems
- Identify reusable/recurring elements in robotic systems
- Identify implicit assumptions made
- Project potential advances to these elements
- Design a generic/flexible implementation of these elements
- Adapt to a number of robotic systems
- Test and study the limitations of the design
- Go back to design and iterate
- Modify/extend/redesign to address limitations and variability across systems

Your generic base is reusable

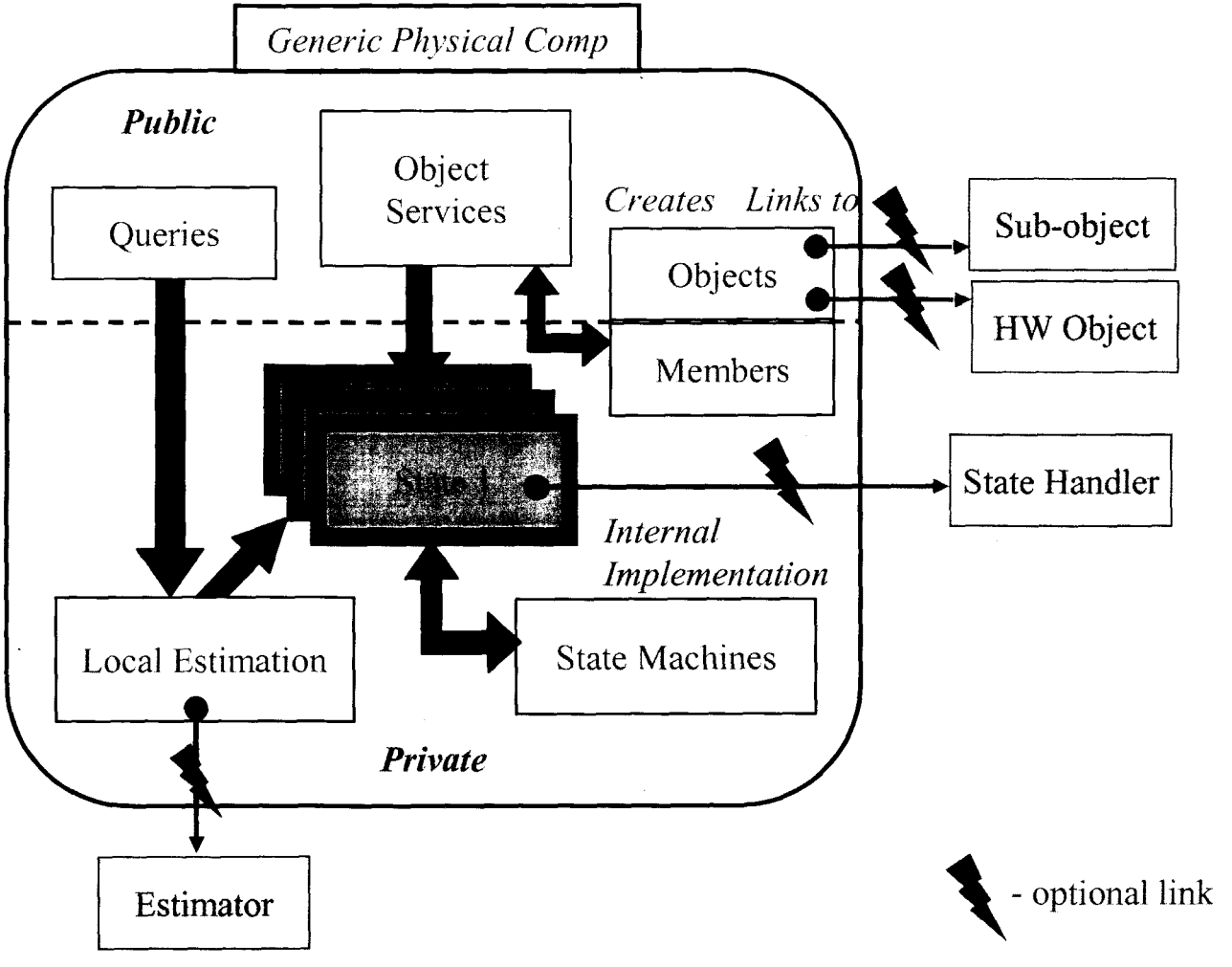
Approach

- Domain knowledge guides design
- Layers of abstraction help master complexity
- Abstractions also provide a classification of various technology elements
- Information hiding protects implementation variability
- Small modular components are more reusable than monolithic blocks
- Interfaces define behavior of various elements

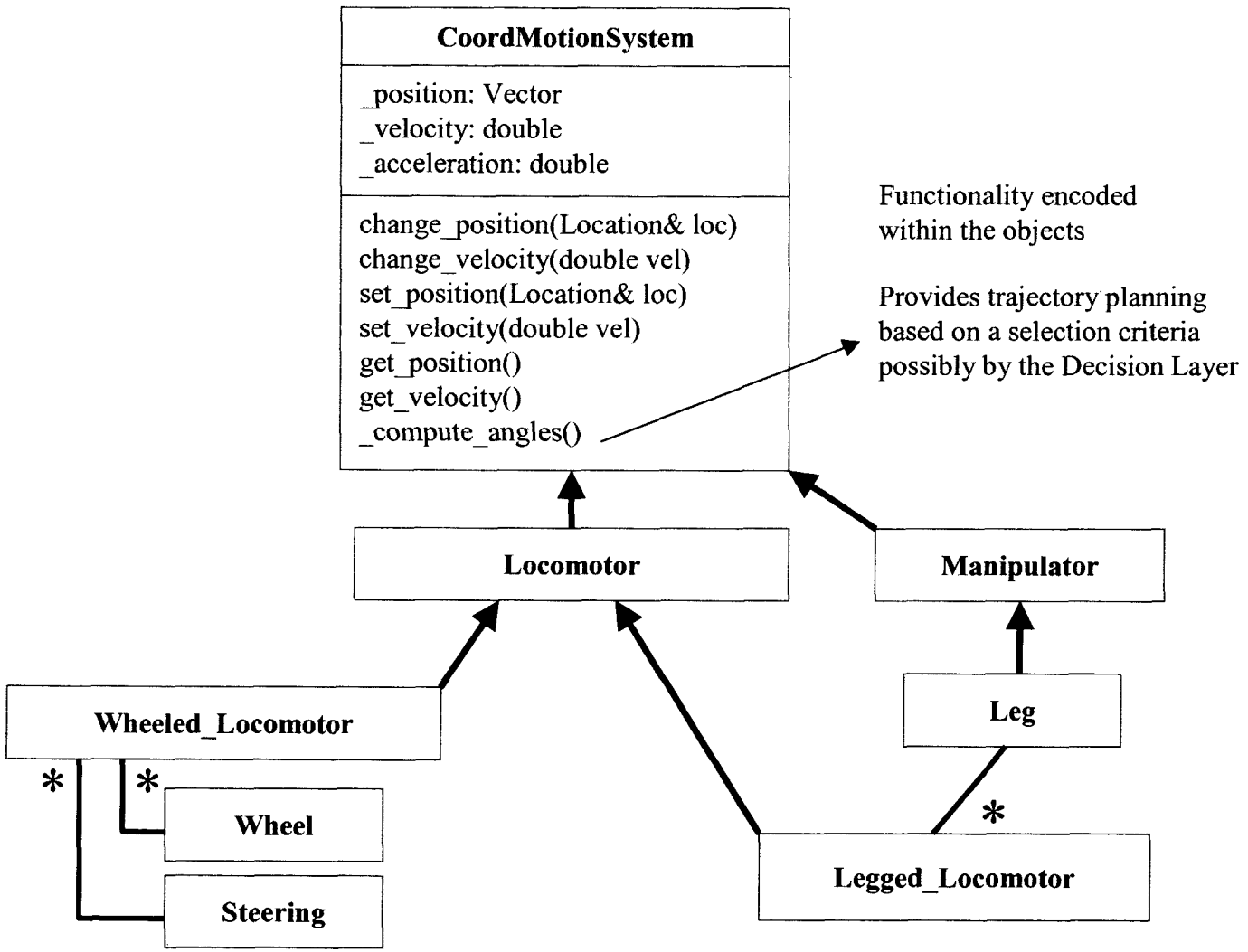
Things to be aware of

- Over-generalizing leads to ineffectiveness
  - More general -> less functionality -> more work for results
  - Number of abstractions vs. complex hierarchies
  - Modular elements with strongly typed interfaces
  - Algorithm generality influences abstraction design
- Runtime models vary across systems
  - Challenges in combining hardware/firmware/software architectures in most effective manner
  - Need for both cooperative and pre-emptive scheduling

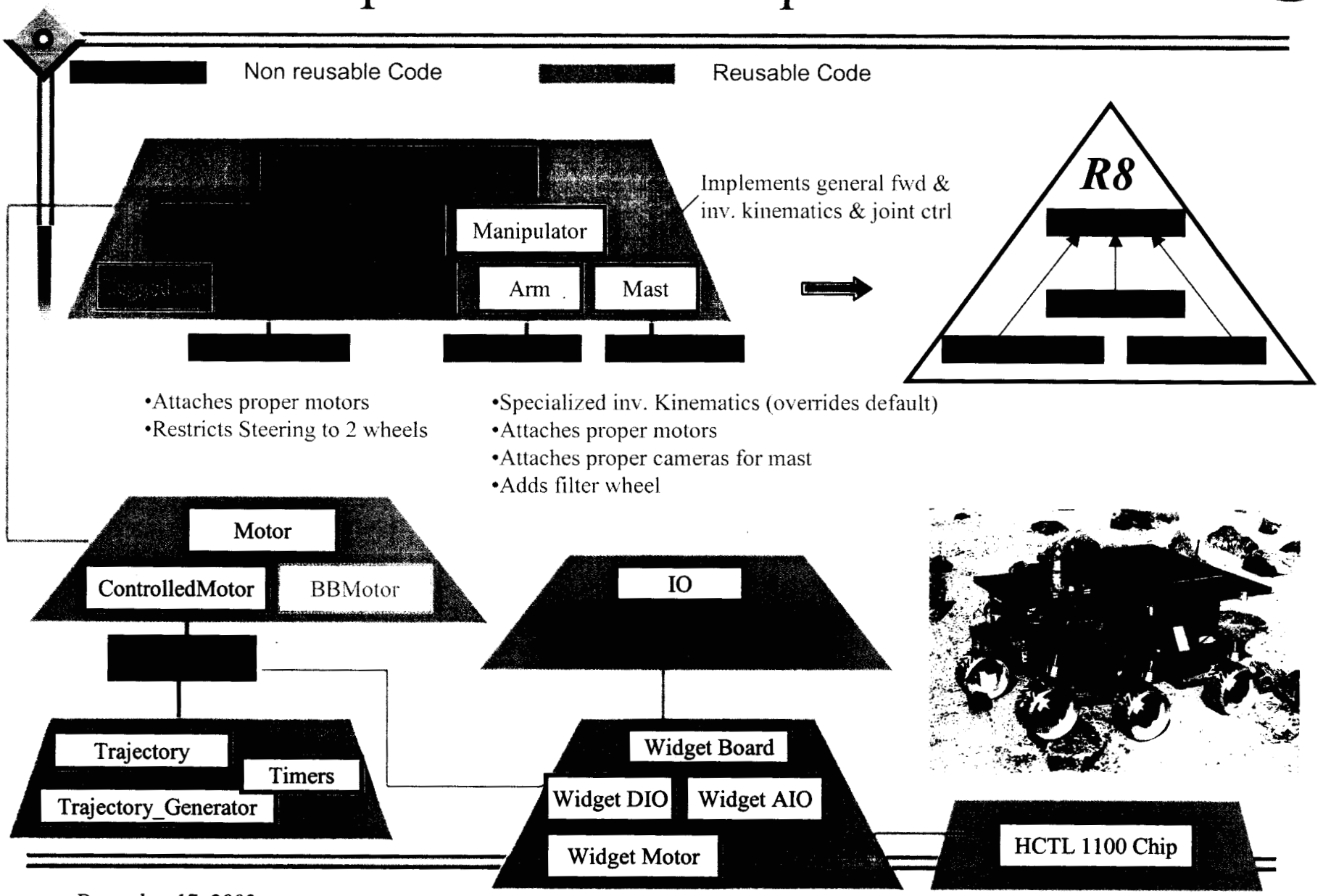
Component Analysis



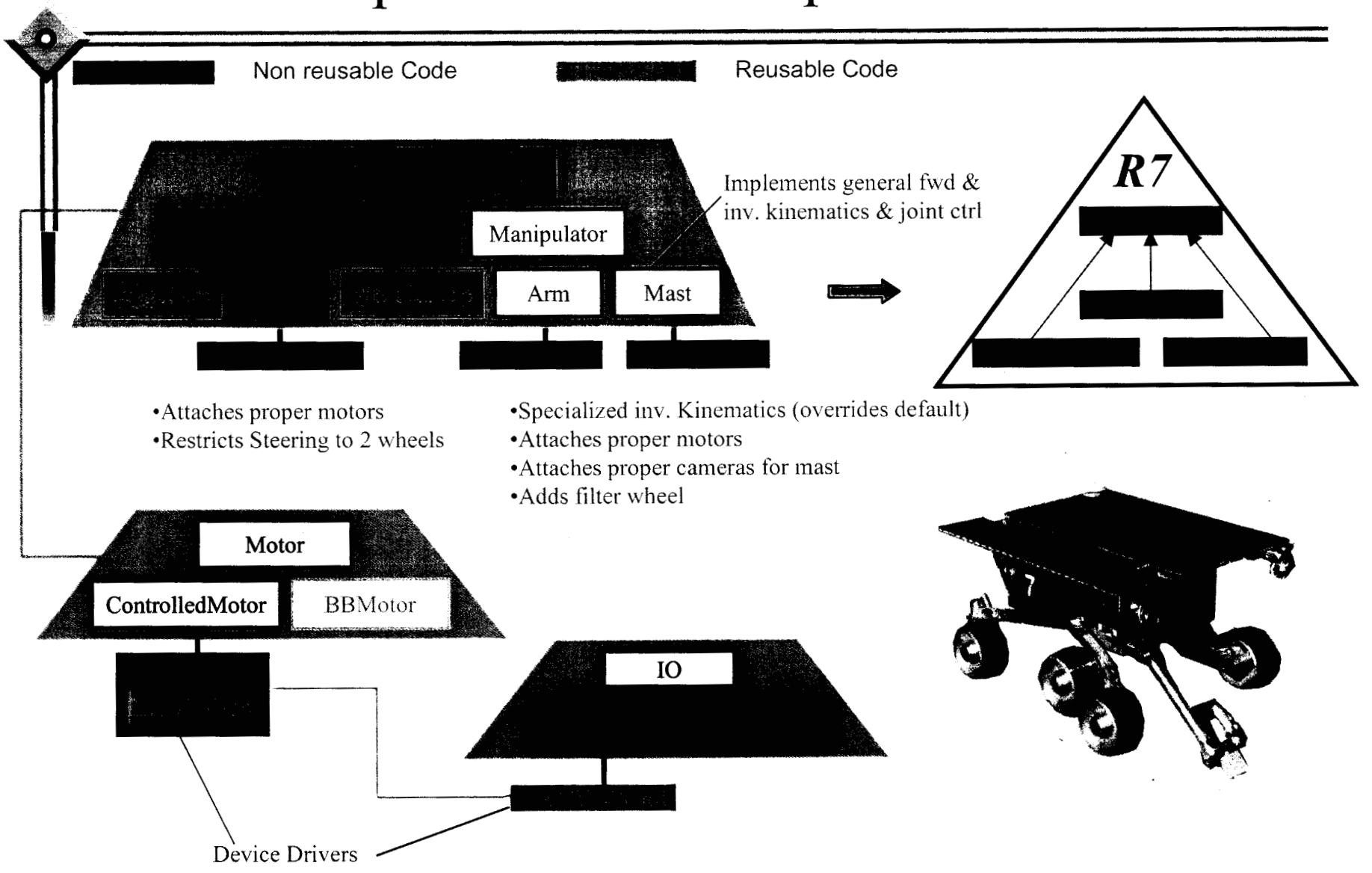
Relationships with Other Components



R8 Specific Rover Implementation

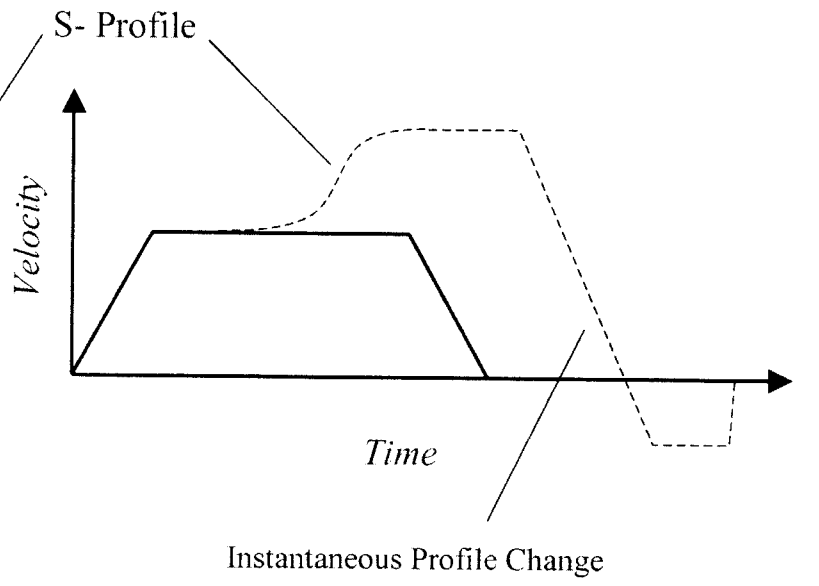


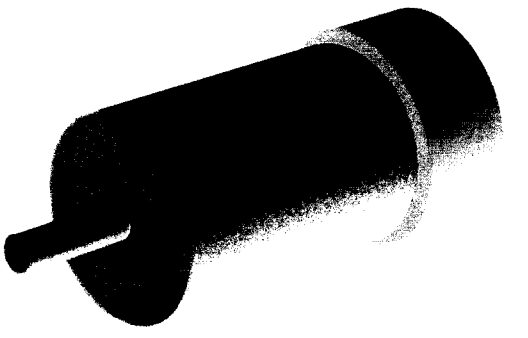
R7 Specific Rover Implementation



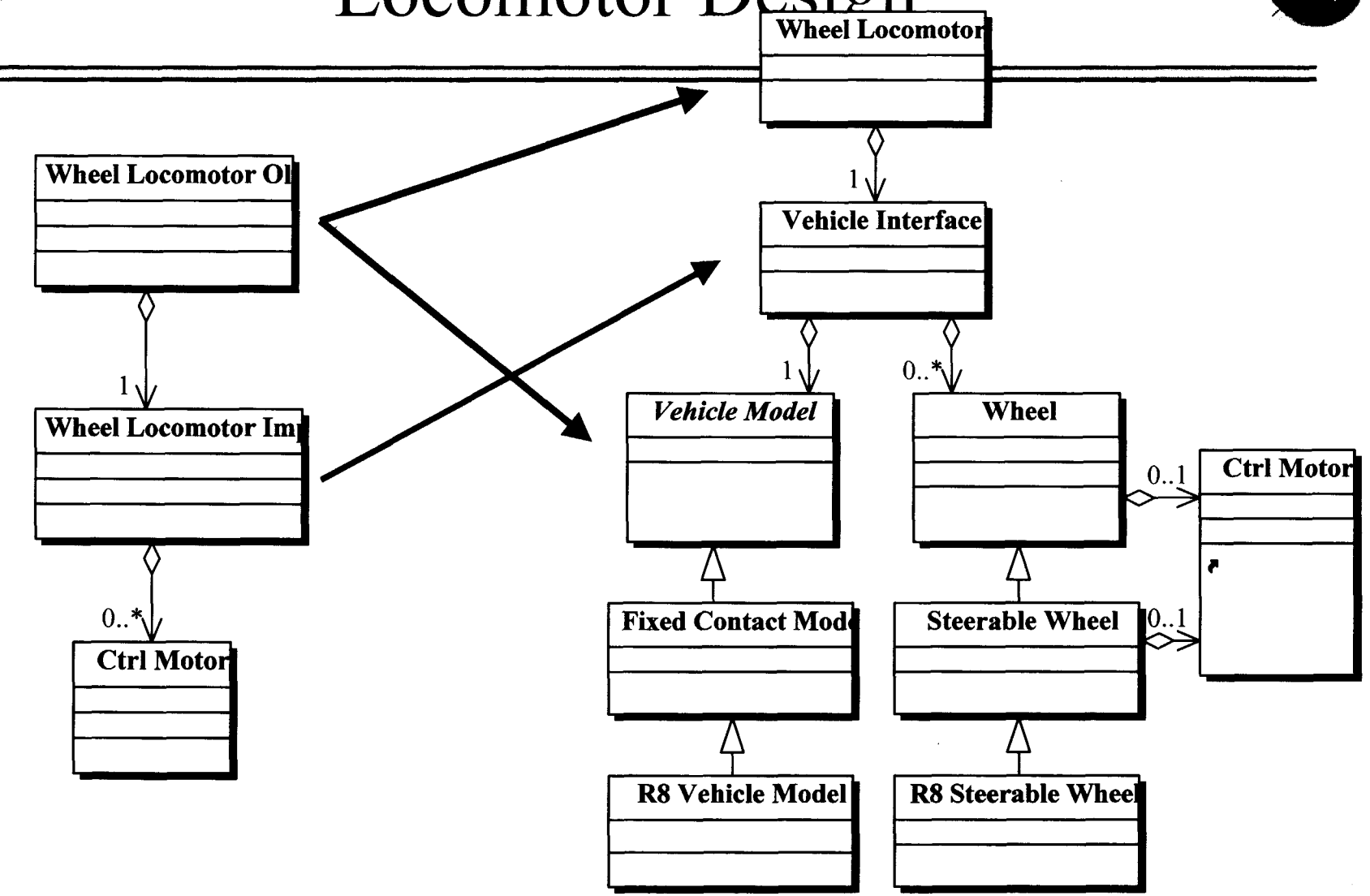
Example: Generic Controlled Motor

- 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





Locomotor Design

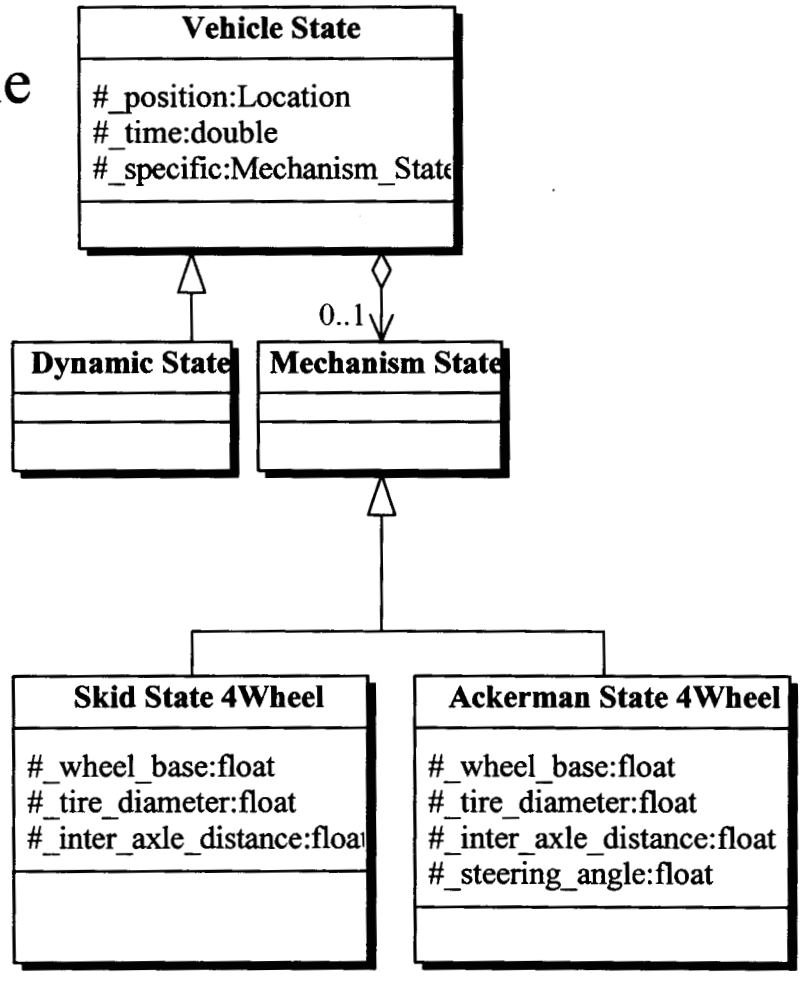


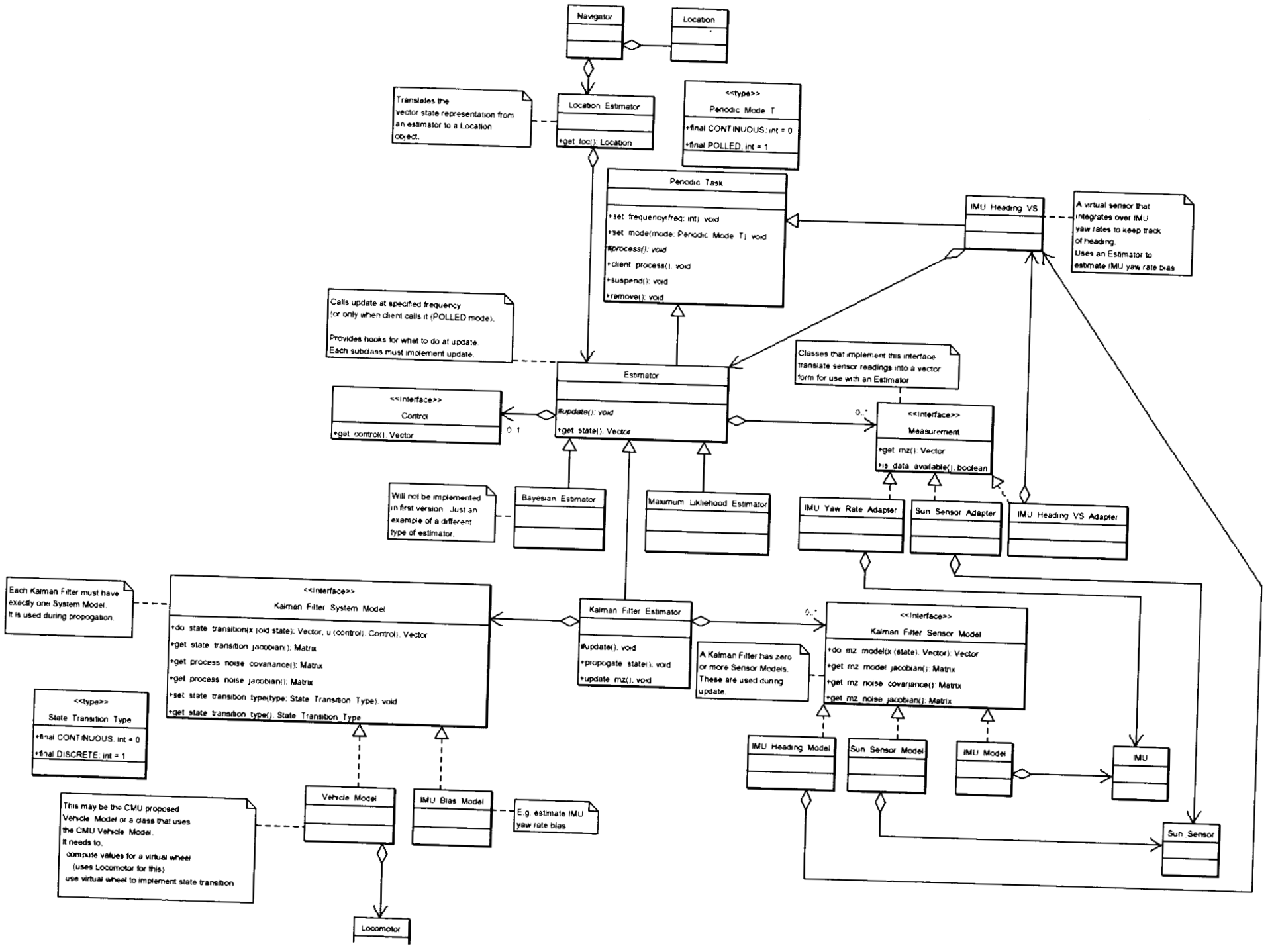
Original Design

New Design

Vehicle State

- Contains 6DOF pose
- Mechanism state contains variable vehicle parameters
- Mechanism information encapsulated in a separate class (bridging)
- Potential for dynamical state or other descendant state classes




Navigator
Location
Translates the vector state representation from an estimator to a Location object.
Location Estimator
+get_loc(): Location
<<type>>
Periodic_Mode_T
+final CONTINUOUS: int = 0
+final POLLED: int = 1
Periodic_Task
+set_frequency(freq: int): void
+set_mode(mode: Periodic_Mode_T): void
#process(): void
+client_process(): void
+suspend(): void
+remove(): void
IMU_Heading_VS
A virtual sensor that integrates over IMU yaw rates to keep track of heading.
Uses an Estimator to estimate IMU yaw rate bias
Calls update at specified frequency (or only when client calls it (POLLED mode).
Provides hooks for what to do at update.
Each subclass must implement update.
Estimator
#update(): void
+get_state(): Vector
Classes that implement this interface translate sensor readings into a vector form for use with an Estimator
<<Interface>>
Control
+get_control(): Vector
0..1
0..*
<<Interface>>
Measurement
+get_mz(): Vector
+is_data_available(): boolean
Will not be implemented in first version. Just an example of a different type of estimator.
Bayesian_Estimator
Maximum_Liklihood_Estimator
IMU_Yaw_Rate_Adapter
Sun_Sensor_Adapter
IMU_Heading_VS_Adapter
Each Kalman Filter must have exactly one System Model.
It is used during propogation.
<<Interface>>
Kalman_Filter_System_Model
+do_state_transition(x (old state): Vector, u (control): Control): Vector
+get_state_transition_jacobian(): Matrix
+get_process_noise_covariance(): Matrix
+get_process_noise_jacobian(): Matrix
+set_state_transition_type(type: State_Transition_Type): void
+get_state_transition_type(): State_Transition_Type
Kalman_Filter_Estimator
#update(): void
+propogate_state(): void
+update_mz(): void
0..*
<<Interface>>
Kalman_Filter_Sensor_Model
+do_mz_model(x (state): Vector): Vector
+get_mz_model_jacobian(): Matrix
+get_mz_noise_covariance(): Matrix
+get_mz_noise_jacobian(): Matrix
A Kalman Filter has zero or more Sensor Models. These are used during update.
<<type>>
State_Transition_Type
+final CONTINUOUS: int = 0
+final DISCRETE: int = 1
IMU_Heading_Model
Sun_Sensor_Model
IMU_Model
IMU
Sun_Sensor
This may be the CMU proposed Vehicle_Model or a class that uses the CMU Vehicle_Model.
It needs to:
compute values for a virtual wheel
(uses Locomotor for this)
use virtual wheel to implement state transition
Vehicle_Model
IMU_Bias_Model
E.g. estimate IMU yaw rate bias
Locomotor

Navigator

- Public interface for navigation capability
- Provides a bridge between evalu and execution
- Manages waypoints
- Multi-threaded
  - Interface available while executing
- Callback provides hook to trigge navigation events
  - Periodic science on the fly

Navigator

#queue < Waypoint* > _waypoints
#Action_Selector & _sit_evaluator
#Locomotor & _locomotor
#State_Estimator & _estimator
#Navigator_Callback * _callback
#bool _active_waypoint
#bool _stop_nav_loop

+Navigator(Locomotor &loc, Action_Selector &eval,StateEstimator
+void set_callback(Navigator_Callback * new_callback)
+int goto_waypoint(Waypoint & wp)
+void abort_waypoint(void)
+bool is_waypoint_active(void)
+void run(void)
+void stop(void)
+void nav_loop(void)

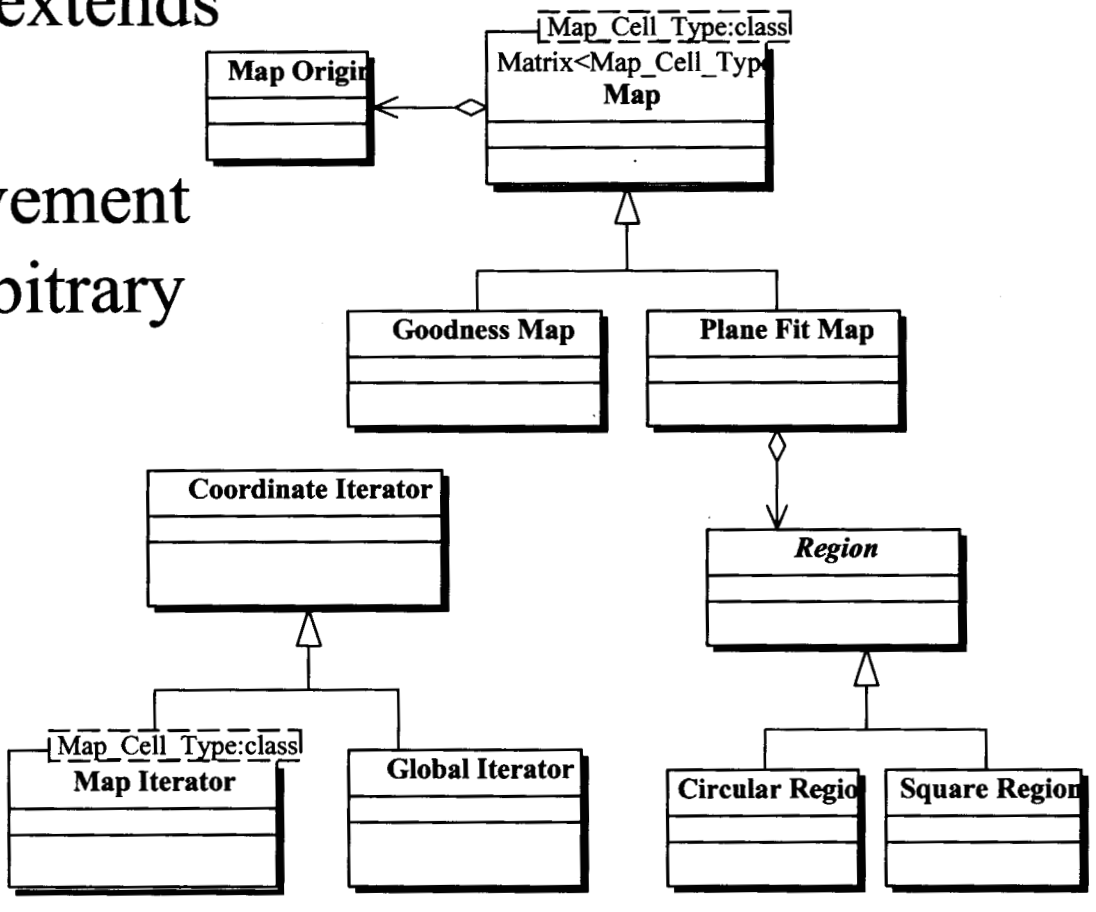
Action Selector

- Navigation algorithms provide this interface
- Used to determine the ne motion the vehicle shoul take

Action Selector
#Vehicle_Model & _model #Waypoint * _cur_way_point
+Action_Selector(Vehicle_Model &mod) +void update(Vehicle_State & robotState) +void set_waypoint(Waypoint & wp) +Drive_Sequence get_best_action(Vehicle_State & curStat

Maps

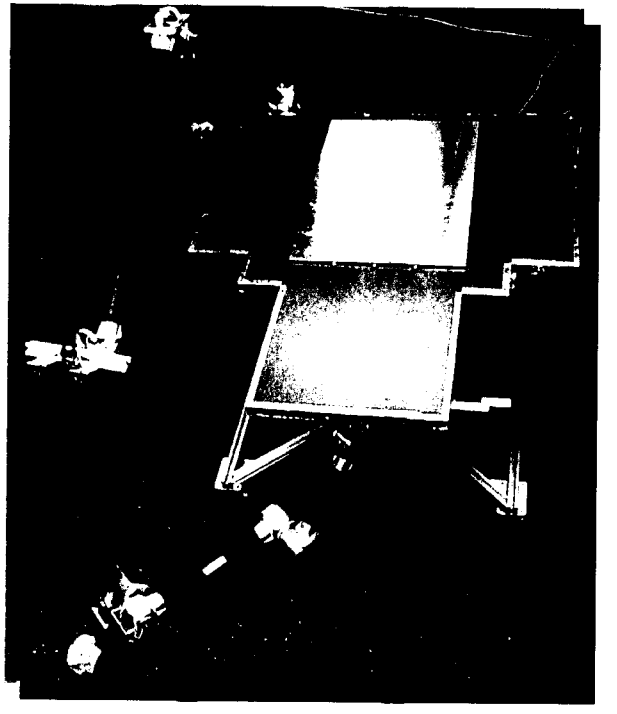
- Generic base class extends matrix
- Iterators allow movement through maps in arbitrary directions



Manipulation Package

Dexter Manipulation platform now part of CLARAty testbed

- Resurrected dexter platform and made part of CLARAty testbed (JPL)
- Some upgrades to manipulation package to support acquisition of multi-tiered panoramic images (JPL)

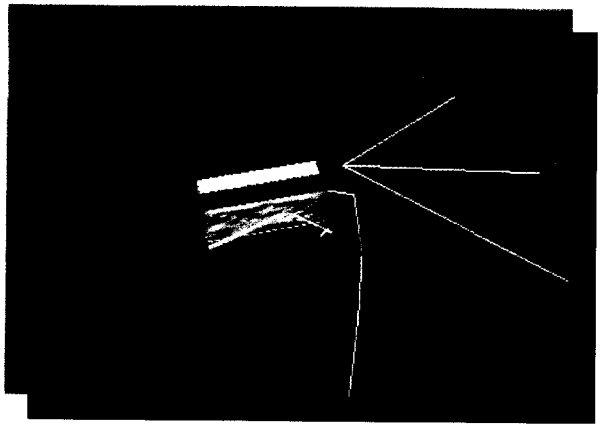


Dexter

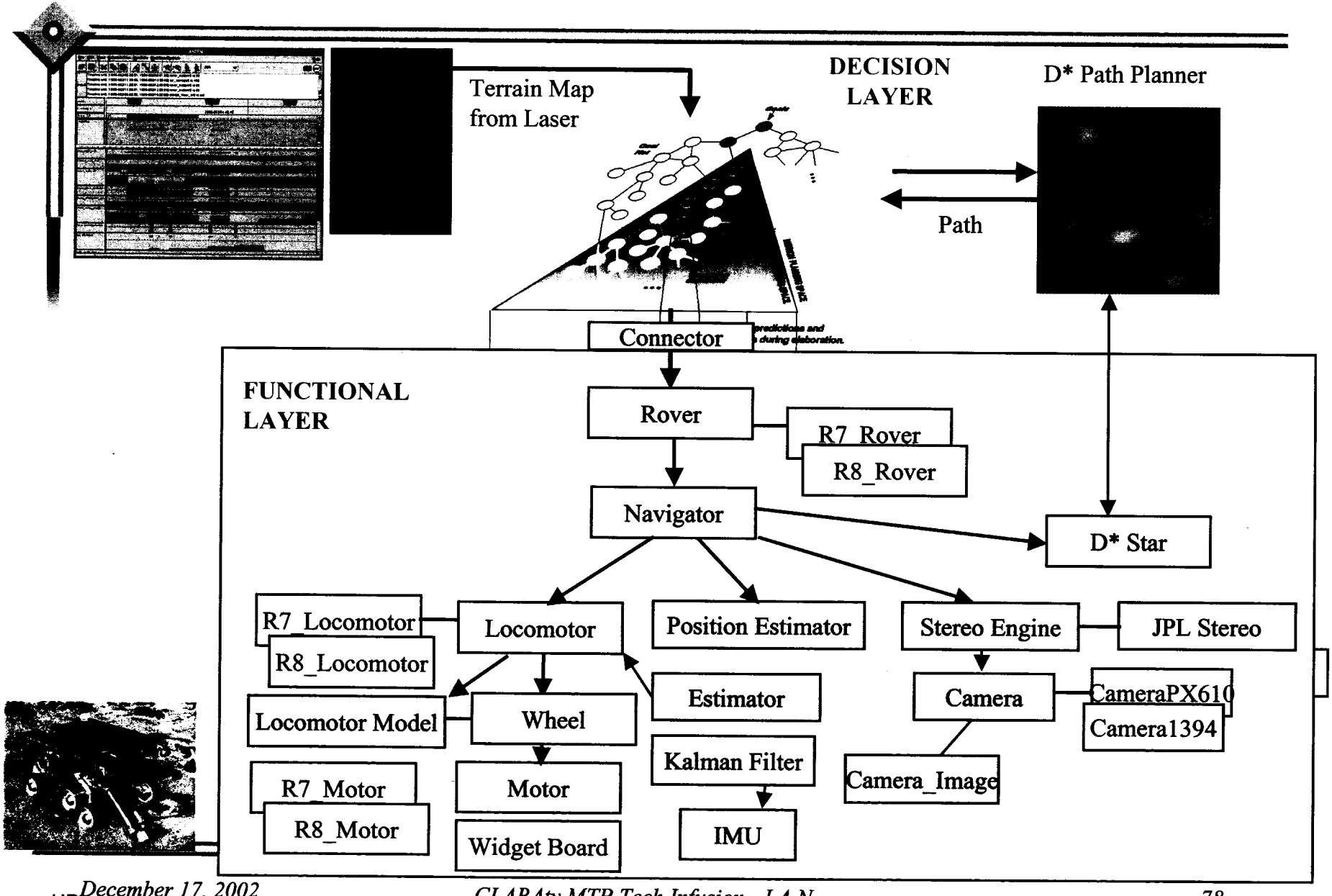
Simulation Package

CLARAty Locomotor talking to ROAMS via motor cmds

- Developed adaptation of the CLARAty controlled motor classes to interface with ROAMS simulation (JPL)
- Tested Locomotor sending commands to wheels which in, turn, send commands to controlled motors (JPL)
- Specialized Sim_Motors turn commands into streams for socket-based communication
- Drove Simulated Rocky 8 rover with new locomotor (CMU)
- Interface between DL/FL (rover-level and ROAMS) is also available



CLARAty Level I Milestone



Working Groups

Working Groups:

- Existing Working groups (no new changes)
  - *CLARAty Navigation Group (Lead: Reid Simmons - CMU)*
  - *CLARAty Vision Group (Lead: Issa Nesnas - JPL)*
  - *CLARAty Mobility & Manipulation Group (Lead: Hari Das - JPL)*
  - *CLARAty Estimation Group (Lead: Ashitey Trebi-Ollennu - JPL)*
  - *CLARAty Decision Layer Group (Lead: Tara Estlin - JPL)*
  - *CLARAty Test/Release Group (Lead: ? - JPL)*
  - *CLARAty Rocky 8 Group (Lead: Richard Petras - JPL)*
  - *K9 CLARAty Developers (Lead: Anne Wright - ARC)*
  - *CLARAty Science Working Group (Lead: Meemong Lee - JPL)*
  - *CLARAty Simulation Group (Lead: Gene Chalfant/Jain - JPL)*

http://claraty.jpl.nasa.gov/groups

Publications & Website

http://claraty.jpl.nasa.gov

- T. Estlin, F. Fisher, D. Gaines, C. Chouinard, S. Schaffer, and I. Nesnas, "Continuous Planning and Execution for an Autonomous Rover," To appear in the *Proceedings of the Third International Workshop on Planning and Scheduling for Space*, Houston, TX, Oct 2002
- I.A.D. Nesnas, "CLARAty - An Architecture for Reusable Robotic Software," CSMISS IT Spotlight Series, June 19, 2002
- I.A.D. Nesnas, R. Volpe, T. Estlin, H. Das, R. Petras D. Mutz, "Toward Developing Reusable Software Components for Robotic Applications" *Proceedings of the International Conference on Intelligent Robots and Systems (IROS),* Maui Hawaii, Oct. 29 - Nov. 3 2001. pdf (8 pages, 2MB)
- T. Estlin, R. Volpe, I.A.D. Nesnas, D. Mutz, F. Fisher, B. Engelhardt, S. Chien, "Decision-Making in a Robotic Architecture for Autonomy." *Proceedings of 6th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS),* Montreal Canada, June 18-21 2001. pdf (8 pages, 72KB)
- R. Volpe, I.A.D. Nesnas, T. Estlin, D. Mutz, R. Petras, H. Das, "The CLARAty Architecture for Robotic Autonomy." *Proceedings of the 2001 IEEE Aerospace Conference, Big Sky Montana, March 10-17 2001.* pdf (12 pages, 470 KB)
- R. Volpe, I.A.D. Nesnas, T. Estlin, D. Mutz, R. Petras, H. Das, "CLARAty: Coupled Layer Architecture for Robotic Autonomy." *JPL Technical Report D-19975*, Dec 2000. pdf (116 pages, 904 KB)

FY03 Plans

Date Ready	Needed By	Technology Description	Receives From	Date Needed
01-Dec-02	CLARAty/ MDS, LT, IP	CLARAty testbed: Rocky 8, Rocky7, six embedded mockups for R8, R7, FIDO		
01-Dec-02	LT, IP	CLARAty/ROAMS open-loop locomotion with simulation of closed-loop wheel control	ROAMS with wheel control simulation	01-Nov-02
15-Dec-02	LT	EKF position estimation for (x,y, heading)		
15-Jan-03	LT, IP	Visual Odometry on Rocky 8 and Rocky 7	RMSA - Slope Navigation	01-Sep-02
31-Jan-03	LT	Sun sensor heading estimation	FIDO - sun sensor code - legacy Rocky 7 - sun sensor code - legacy	15-Nov-02
01-Feb-03	IP	2D Visual Tracker with obstacle avoidance on Rocky 8 and Rocky 7	RMSA - Visual Tracking	01-Nov-02
15-Feb-03	LT	ROAMS/CLARAty with reasonable R8 data (kinematic, mass inertias, etc) with locomotion, wheel control, IMU, and pose estimation	ROAMS version with IMU simulation	01-Dec-02
01-Mar-03	IP	GESTALT navigator on Rocky 8 and Rocky 7 (encapsulated latest MER version)	MER - GESTALT navigation software	01-Dec-02
15-Mar-03		CLARAty/WITS - demonstrate uplink and downlink of a single command (e.g. Hazcam)	WITS version to support interface	01-Feb-03
01-Apr-03	LT, IP	Two navigators designed with CLARAty infrastucture - Morphin++, Gestalt	Morphin++ - navigation software MER - GESTALT navigation software	01-Dec-02
15-May-03	IP	Stereo-based manipulation infrastructure - control/coordination/sensing for Rocky 8, PDM, and K9 Arms and masts	PDM - legacy instrument placement	
01-Jun-03	LT	CLARAty/WITS - demonstrate commanding for locomotion (driving) and navigation		
15-Jun-03	IP	CLARAty/ROAMS interface for rover-based manipulators	ROAMS simulation of manipulator kinematics and interface	01-Apr-03
30-Jun-03	IP	CLARAty/WITS - demonstrate manipulator commanding for Rocky 8 Define format for shared models	WITS version to support Rocky 8 commanding interface	01-Mar-03