Navlab Core Technologies

Jay Gowdy and Rob Maclachlan

SAIC *Robotics Institute*

SAIC?

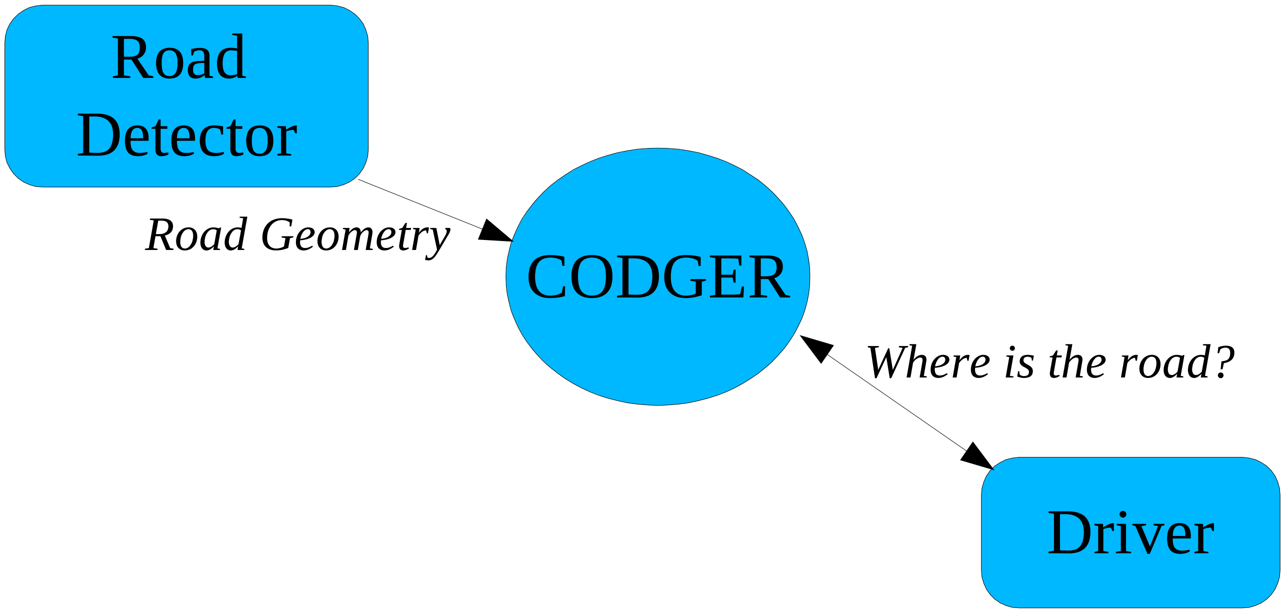
- Center for Intelligent Robotics and Unmanned Systems (CIRUS)
  - Was in Denver
  - Now split between Mclean, VA and here
  - Soon to be in the Collaborative Innovation Center
- 4 employees in Pittsburgh by December
  - Including myself, Karl Kluge, Chris Urmson
- Room for more...
- If you are looking for a big, friendly robotics company for collaboration, talk to me.

Overview

- Navlab: an architectural history
- ModUtils: supporting robot module development
- DATMO: A current core algoritm
- Wrap-up

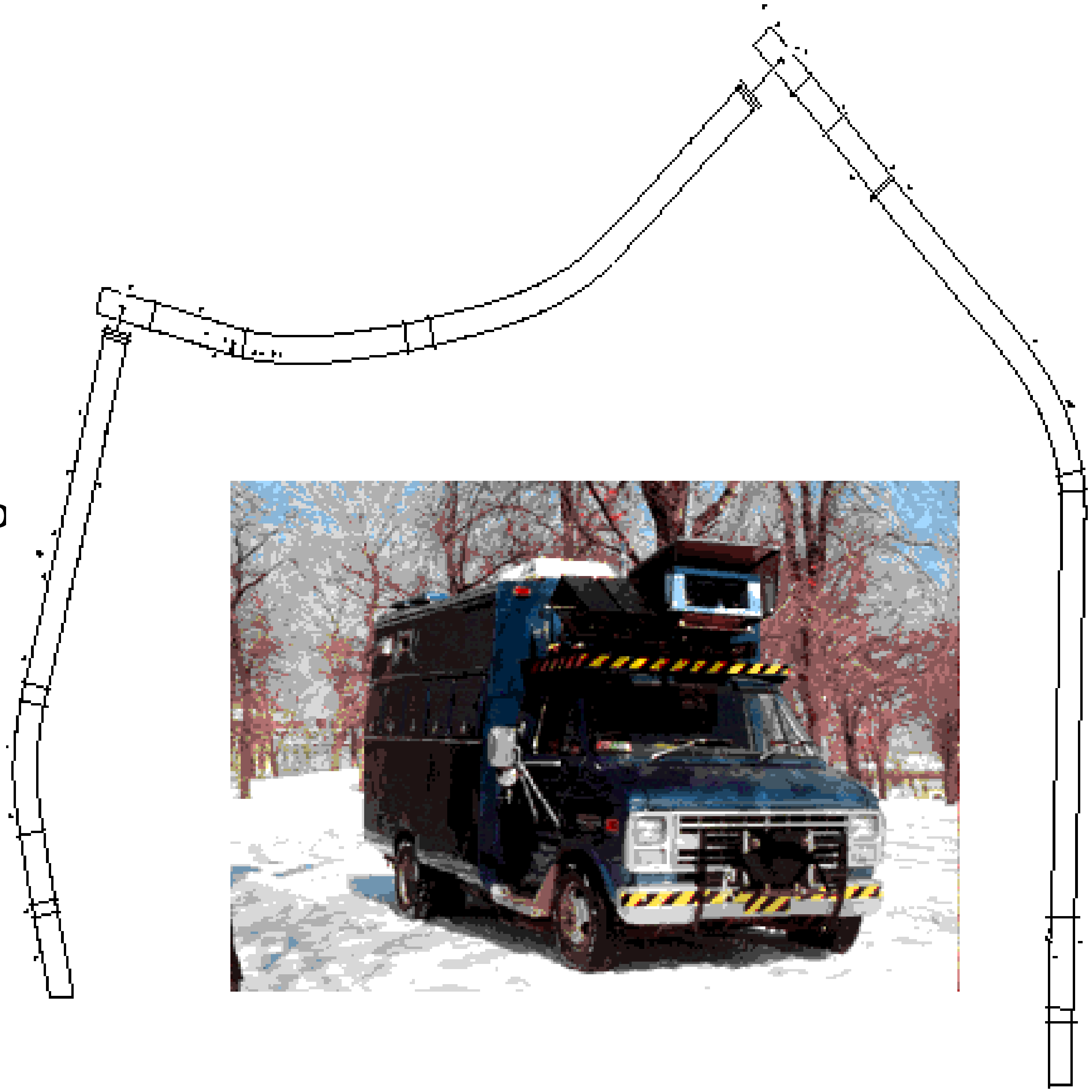
Initial Architecture

- 1985-1988, CODGER
  - Central database
  - Geometric information
  - All communications through the data base



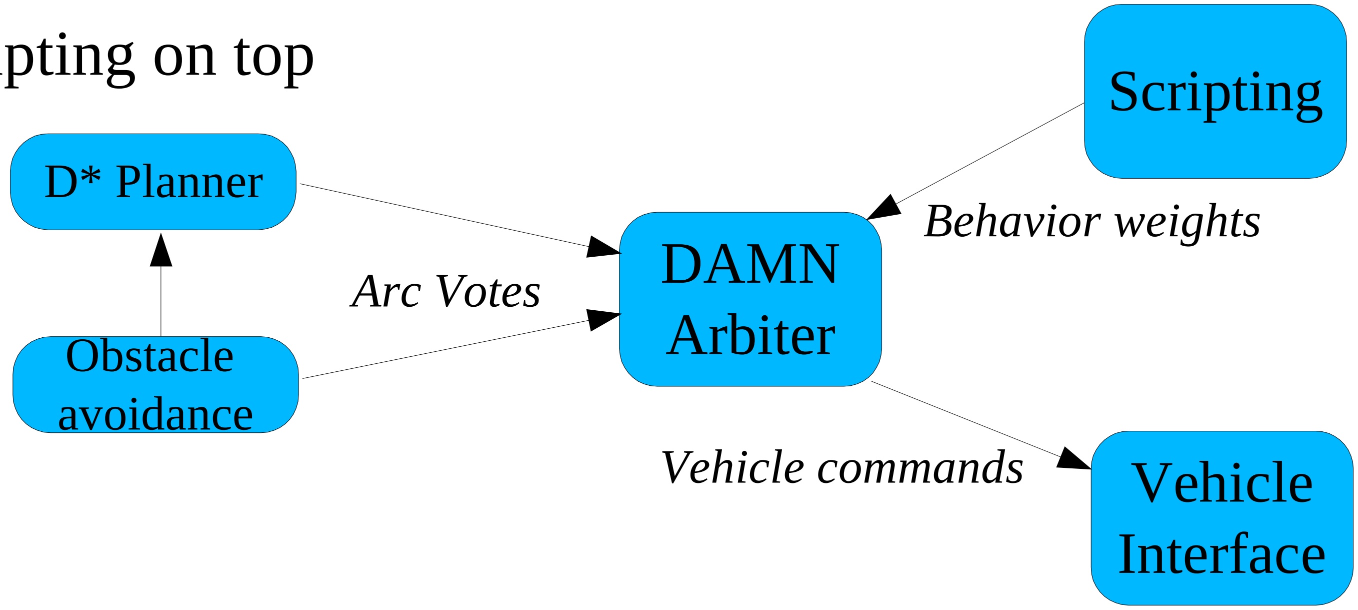
Navlab 1: Minimal Architecture

- 1989-1990
- Usually one perception algorithm and a controller
- Eventually developed hand built architecture
- Simple, point to point TCP/IP messaging
  - TCX -> IPT
- Annotated maps for "cold beverage" delivery



D**N Architecture

- 1990-1994, UGV Demo I, II
- DAMN Architecture
  - Combine multiple perception algorithms
  - Sensorimotor behaviors
  - Arc voting
- Minimal scripting on top

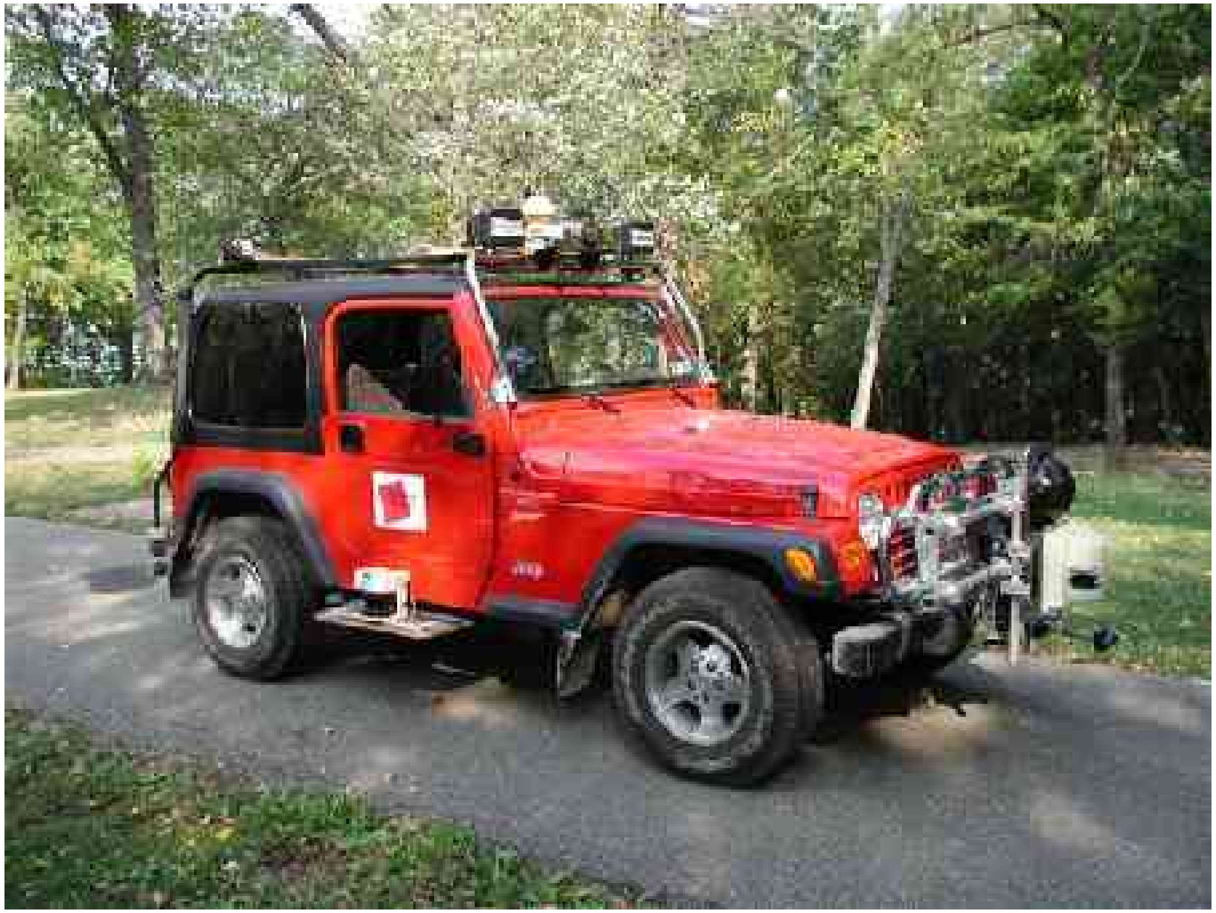


Highway Navigation

- No Hands Across America, AHS, 1994-1997
- Single purpose architecture
  - Servoing for lateral control
  - Servoing for longitudinal control
  - Reparameterize as necessary
  - Single purpose, single process
- Drove at highway speeds

Today...

- NavLab 11



What is an architecture?

- What does “architecture” mean anyway?
  - A blue print?
  - A grand philosophy?
- The hope:
  - Blueprint architectures standardizing over life of a project
  - Grand unified theory architectures standardizing over tasks (or all of robotics)
- Permanent, pervasive standards will lead to massive code reuse
  - Has this ever been demonstrated?

Modules, not Architectures

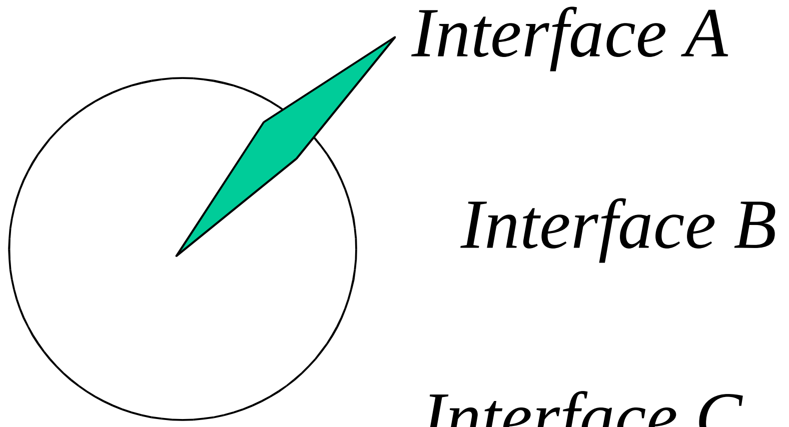
- The Navlab approach
  - Tasks drive algorithms, algorithms drive architectures
- Concentrate on
  - Support for algorithmic module development
  - Support for seamless transition of modules from "architecture" to "architecture".
- Provide an architectural toolkit to build systems of loosely coupled modules
- Using the "architectural" support does not trump the "module development" support

ModUtils

- A meta-architectural C++ toolkit for developing and integrating robot software modules
- Provides a common Module framework which developers fill in with algorithm and display code
- All input and output to the “system” is done through reconfigurable interfaces
- Modules are unaware of what lies on the other side of the interfaces
- Modules are isolated from the architecture in which they reside

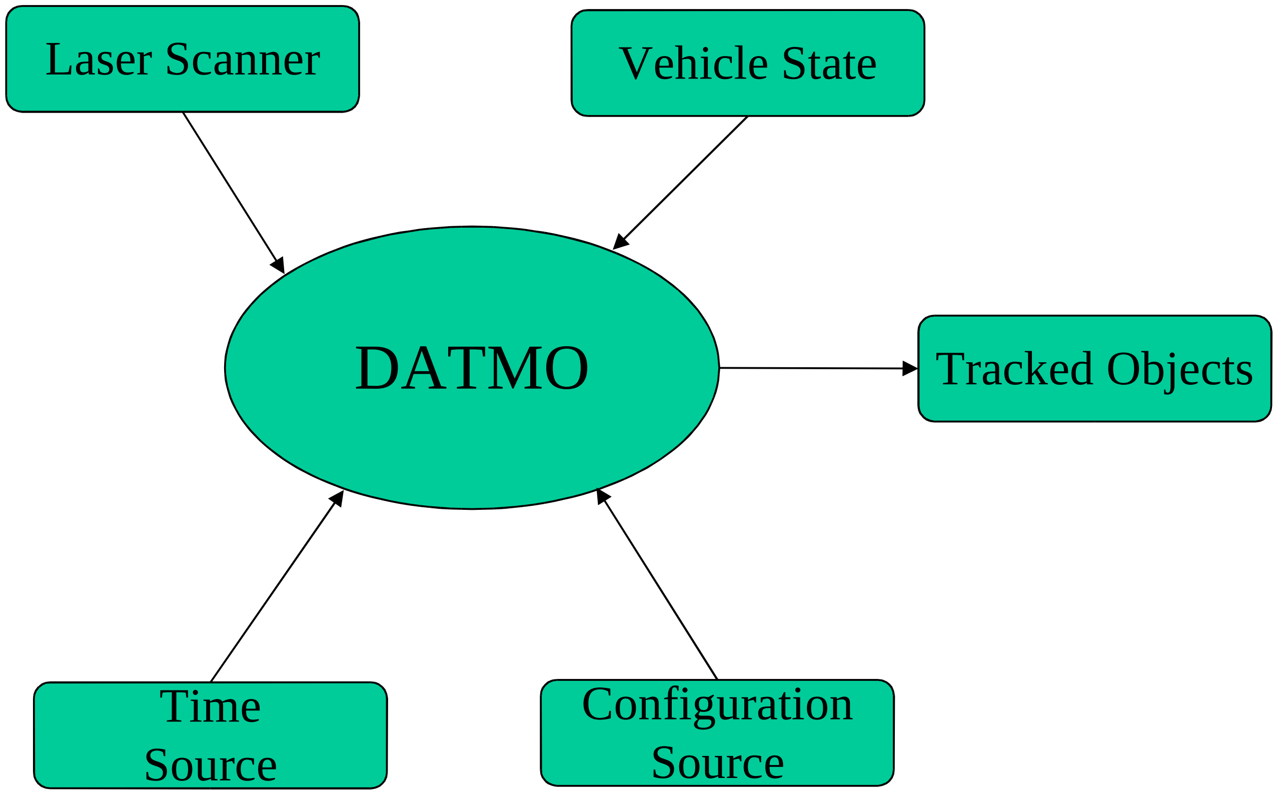
Reconfigurable Interfaces

- Modules view the system through reconfigurable interfaces built using ModUtils
  - Abstract API
  - Suite of instances available at runtime
  - Configured through specification strings read from configuration source
- Consistent means of isolating modules from the architecture in which they reside



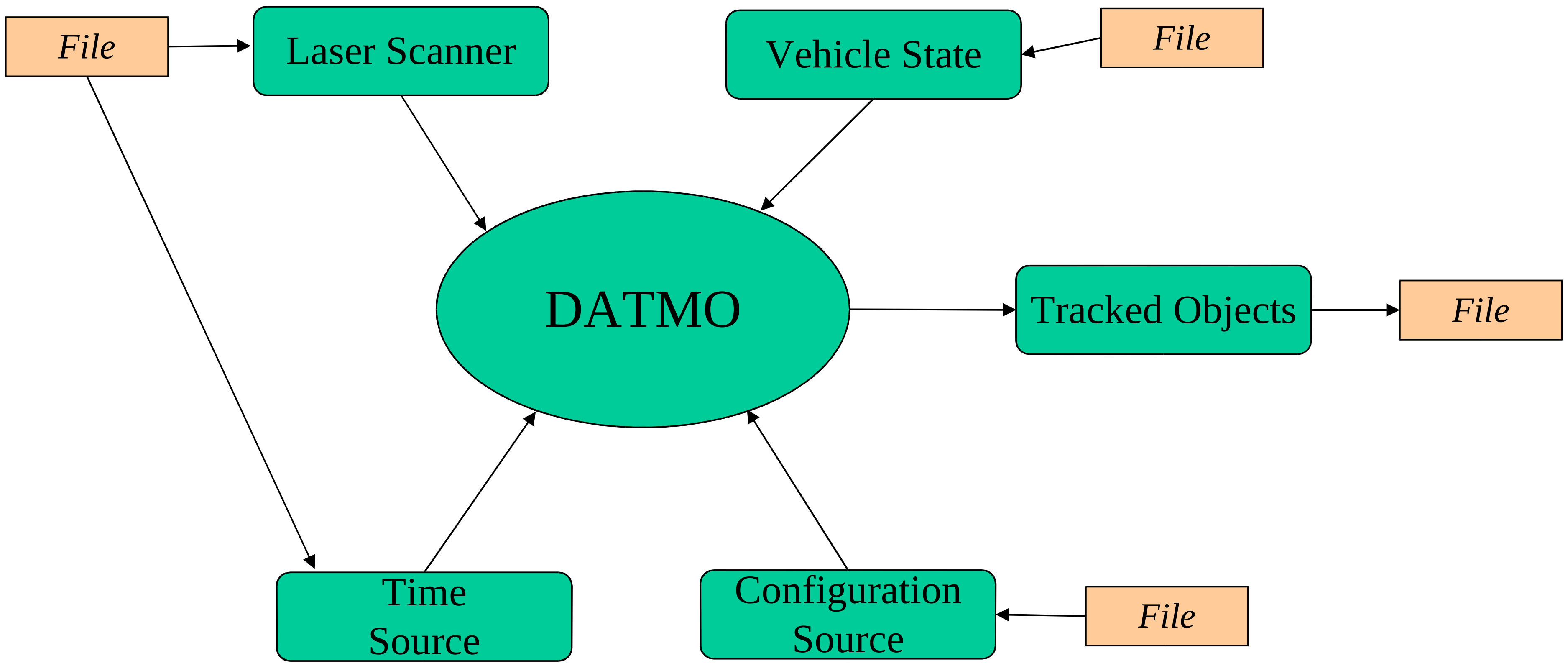
Example: DATMO

- Detection and Tracking of Moving Objects
- Abstract data sources, destinations



Module Development Architecture

- Stand-alone development: Single proces
- All data read from time tagged files
- Debug with local GUI, output to file for post analysis

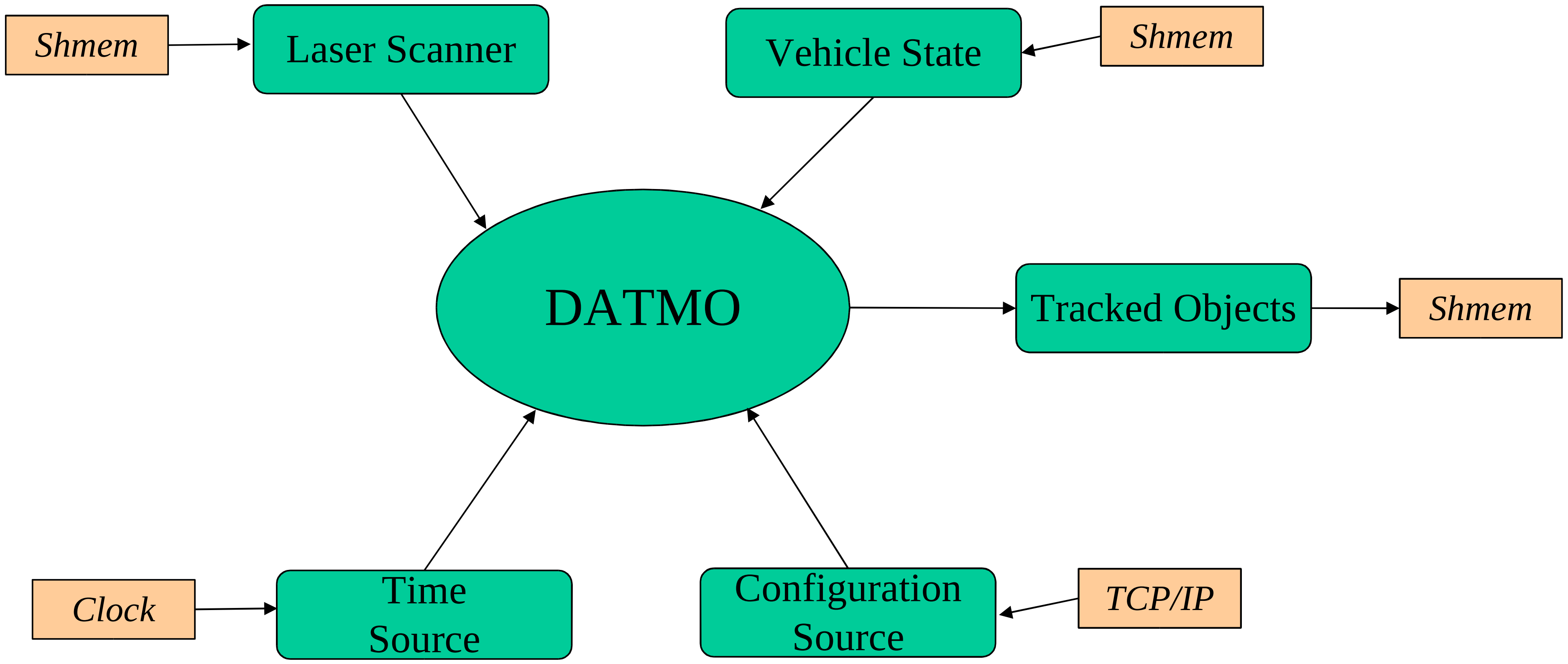


ModUtils Support

- Provides standard abilities for writing time-stamped data, and accessing by time.
- Provides an extremely flexible configuration file system, accessed through an abstract “Configuration Source”
- Provides an abstract interface to time
  - In module development, time can be driven by the data.
- Provides utilities for creating 2D, 3D, and image overlay debugging displays

Integrated Architecture

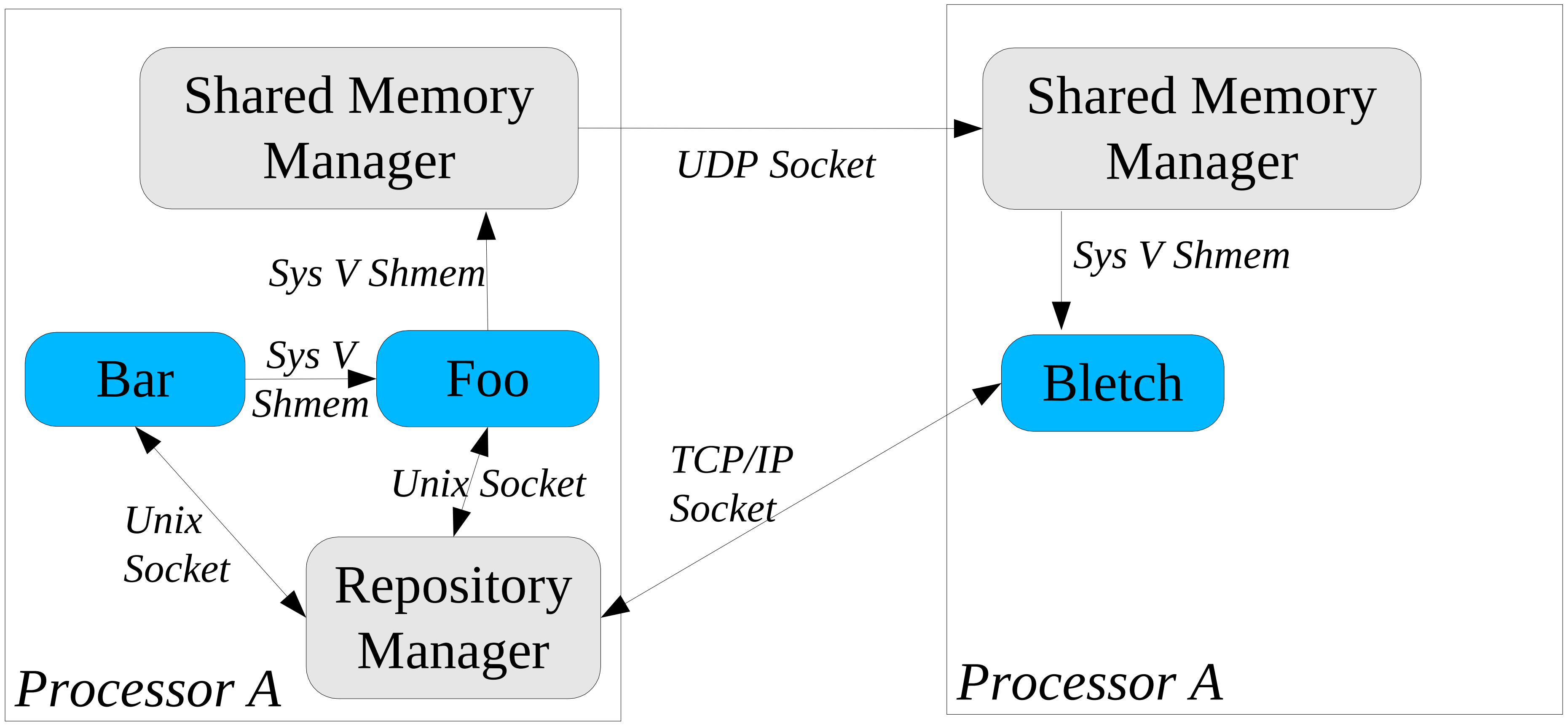
- Read, output data via shared memory/UDP
- Access central configuration server via TCP/IP
- Looks the same to the module algorithm



ModUtils Communications Support

- Interprocessor shared memory
  - For “signals”, Sys V shared memory or UDP
  - 99% of communications in our systems
- Simple messaging toolkit
  - For “symbols”, TCP/IP
  - Mainly used at startup and shutdown
- Central database of configuration information
  - For “information,” implemented with messaging.
  - Not just a passive repository, users can “attach” and “set” values
  - A general back door for communications

A Typical ModUtils “Architecture”



Repository Manager:

- Has all configuration information
- Acts as a blackboard
- Runs python scripts to manage the processes

Wrap Up

- Since DATMO is written with ModUtils
  - Able to move from a single Sick to multiple fused sicks for 360 degree awareness
  - Able to integrate it rapidly on the GD XUV
- ModUtils is best for me, is it good in general?
- ModUtils is open source and available
  - User manual: `http://geeveegie.msl.ri.cmu.edu/jayg/ModuleDoc`
  - Code snapshot: `http://geeveegie.msl.ri.cmu.edu/jayg/ModUtils.tgz`