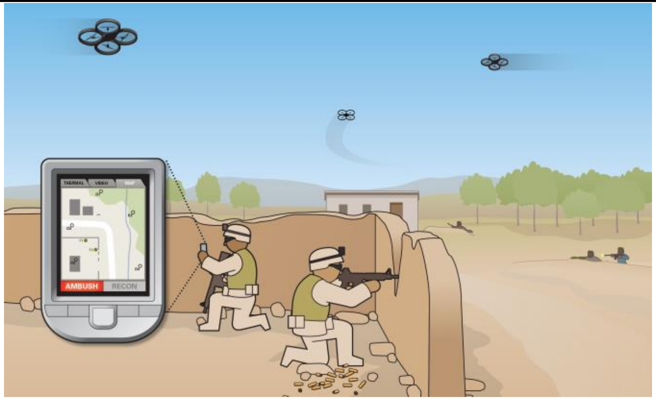
Collaborative Autonomy with Group Autonomy for Mobile Systems (GAMS)

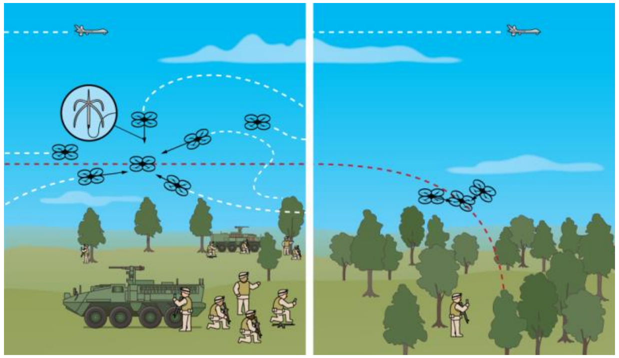
Presenter: James Edmondson (jredmondson@sei.cmu.edu)

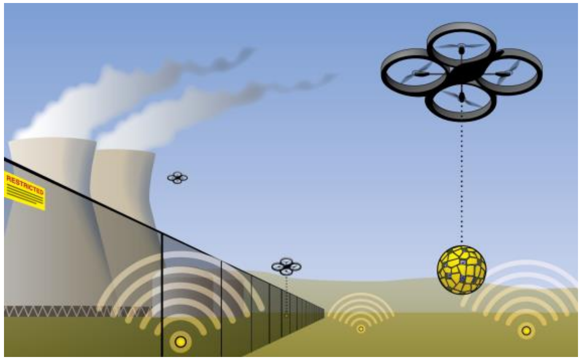
Date: August 19, 2014

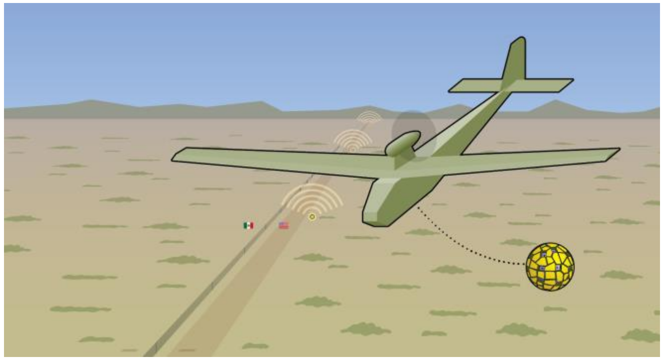
Problems facing group autonomy

- Autonomy focus is on single unit control
- Focus is on centralized controllers (prone to failure/attack)
- Autonomy frameworks tend to be targeted at homogeneous platforms and algorithms
- Blocking communications are prone to faults/attacks/outages/loss-of-control
- GPS is highly inaccurate for precise maneuvers
- Lack of standardization for autonomous collaboration



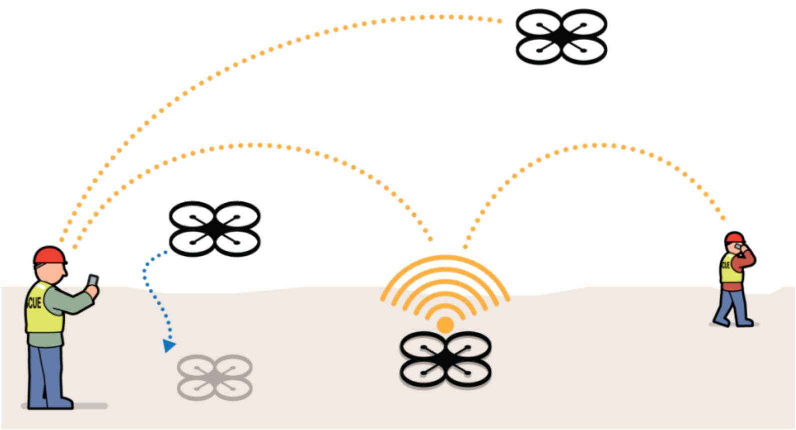






Our Approach to Group Autonomy

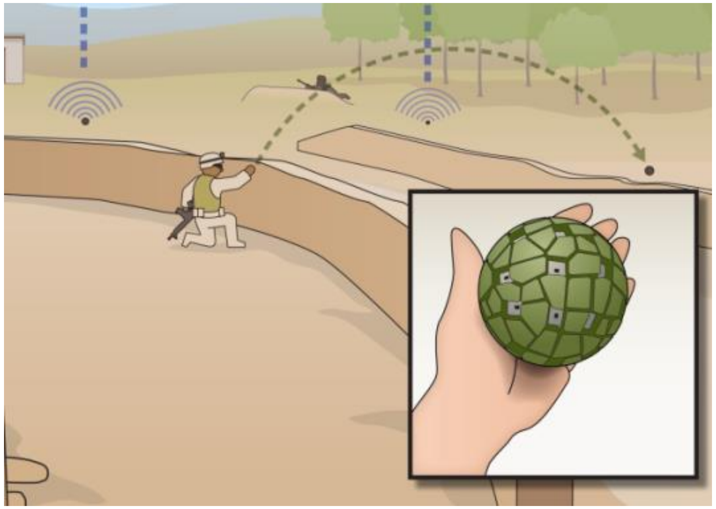
1. **Create a portable, open-sourced, decentralized operating environment for autonomous control and feedback. Focus on scalability, performance and extensibility**
2. **Integrate the operating environment into unmanned autonomous systems (UAS), platforms, smartphones, tablets, and other devices. Focus on portability.**
3. **Design algorithms and tools to perform mission-oriented tasks like area coverage and network bridging between squads**
4. **Design user interfaces to help single human operators control and understand a swarm of UAS, devices, and sensors (human-on-the-loop autonomy)**

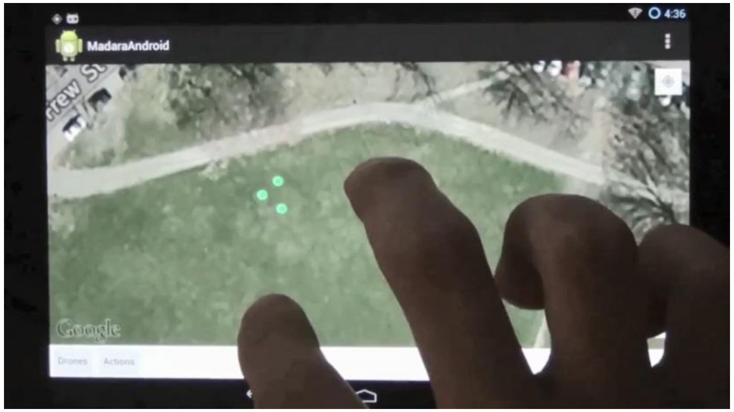


FY 2015 Technologies/Platforms

We are investigating several platforms and collaborations this year, including:

- **UAVs (Parrot and 3D Robotics)**
- **Autonomous Boats (Paul Scerri—CMU)**
- **Micro-UAVs (Vijay Kumar—UPenn GRASP)**
- **Flood sensors (Anthony Rowe, Huntingdon County EMS)**
- **Throwables (Bounce Imaging), Smartphones, Tablets (Android)**
- **High precision and gps-denied positioning**





Principles of our open-sourced middleware (MADARA and GAMS)

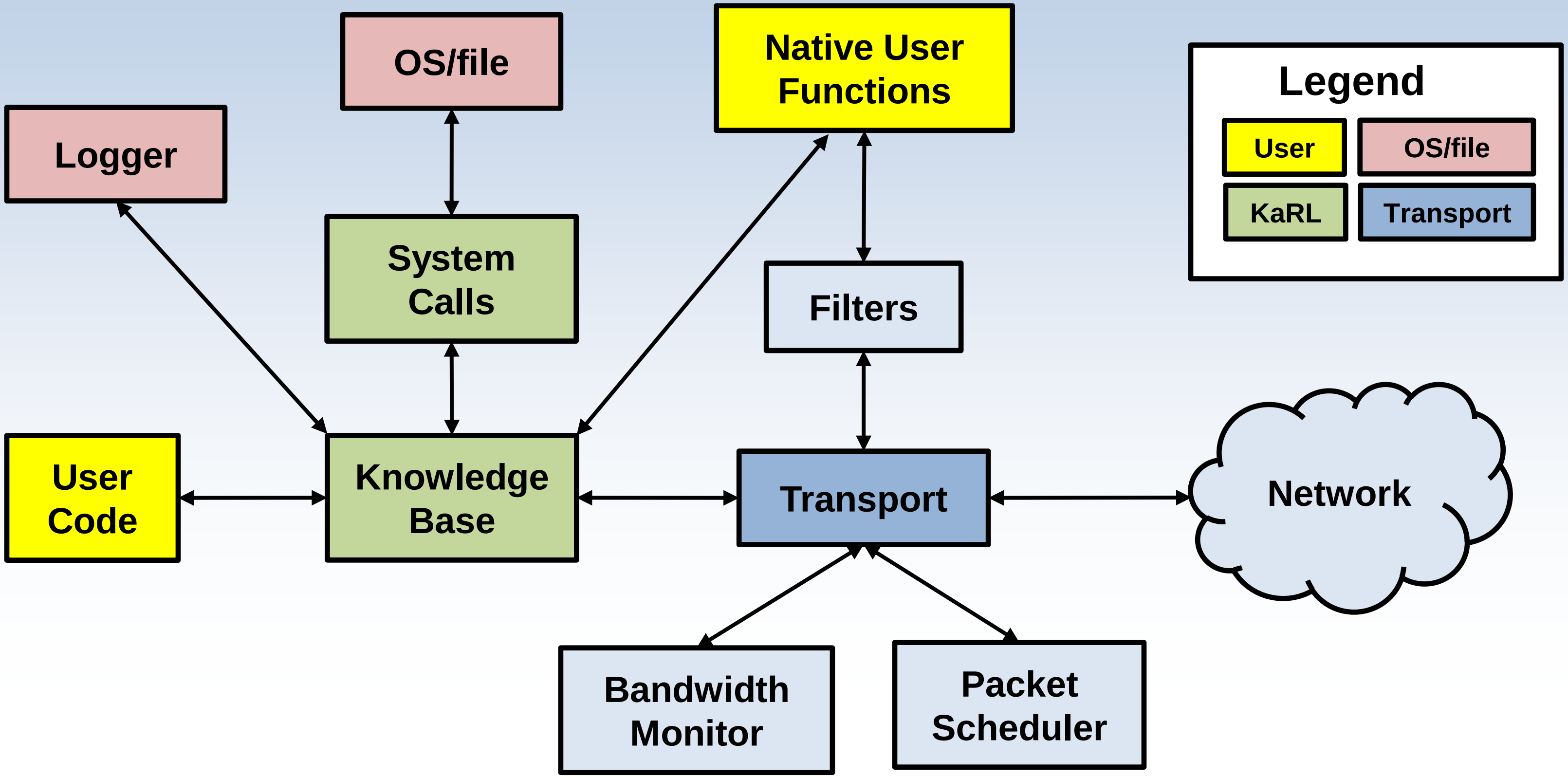
1. Be useful to application developers
2. Enable distributed, decentralized artificial intelligence
3. Be fast, small, and capable
4. Be portable to as many platforms relevant to UAS as possible
5. Be extensible to facilitate new transports, linking with external libraries, security, assurance, and consistency
6. Provide extensive documentation

Key MADARA Features (2009-present)

- Allows developers to write both state-based and event-based programs (or combinations of both) for distributed artificial intelligence
  - Programs can react to receive, send, or rebroadcast events
  - Programs can have deadline-enforced periodic executions, wait for certain state-based conditions to come true, or execute efficient, dynamic actions in KaRL (Knowledge and Reasoning Language)
- Allows developers to script (KaRL) or utilize object-oriented programming to codify their algorithms and applications
- Supports C++, Java, Python, ARM, Intel, Windows, Linux, Android, iOS
- Supports IP multicast, broadcast, unicast, OMG DDS transports
- Enforces consistency of updates through Lamport clocks, priorities
- Extensible transport layer, filtering system, and callbacks
- Extensive documentation (guides, tutorials, doxygen)

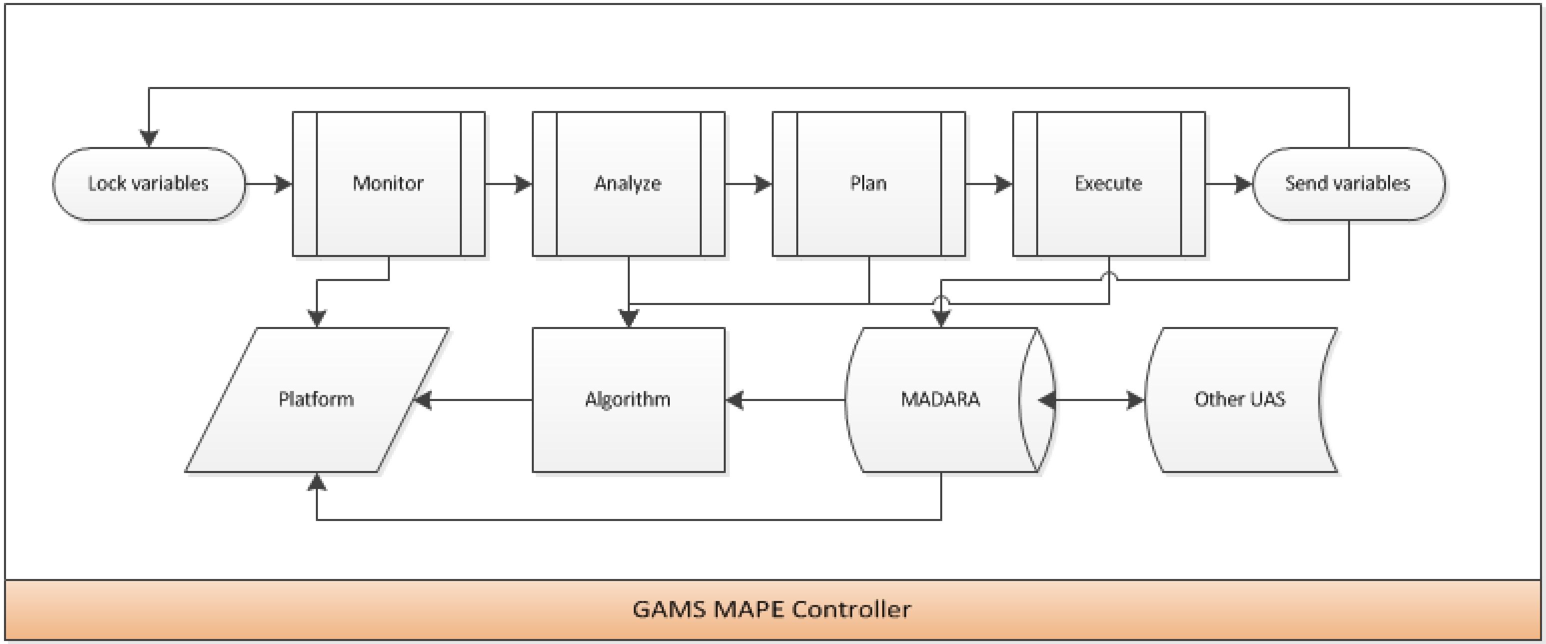
MADARA Architecture

More information, tutorials, and documentation at http://madara.googlecode.com

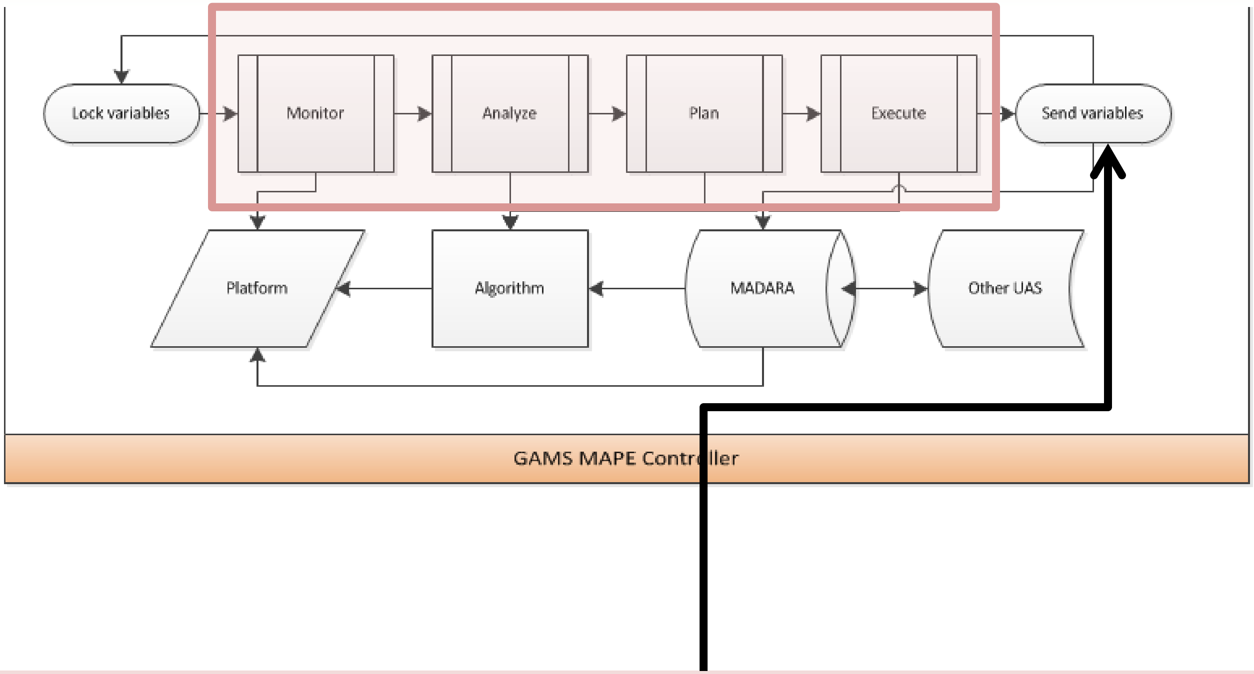


GAMS Architecture (FY 2014)

1. Built directly on top of MADARA
2. Utilizes MAPE loop (IBM autonomy construct)
3. Provides extensible platform, sensor, and algorithm support
4. Uses new MADARA feature called containers, which support object-oriented programming of the Knowledge Base



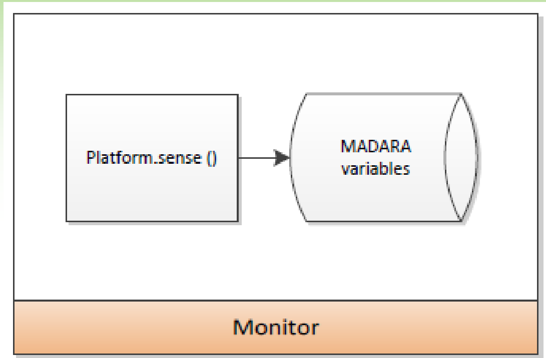
GAMS Architecture (FY 2014)

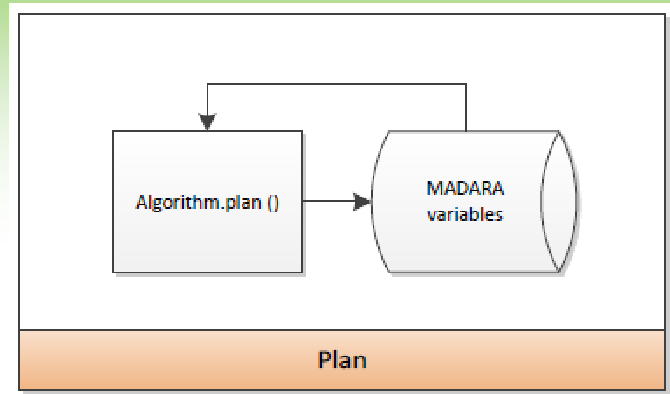


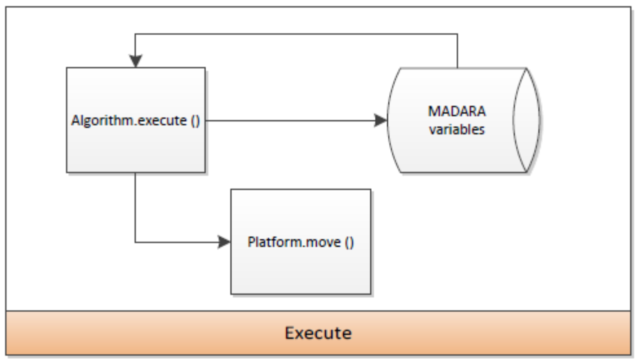
Key points:

- **During the MAPE loop, the context is locked from external updates**
- **At the end of the MAPE loop, all global variable changes are aggregated together and sent to other UAS participating in the mission**

GAMS Platform and Algorithm Interactions

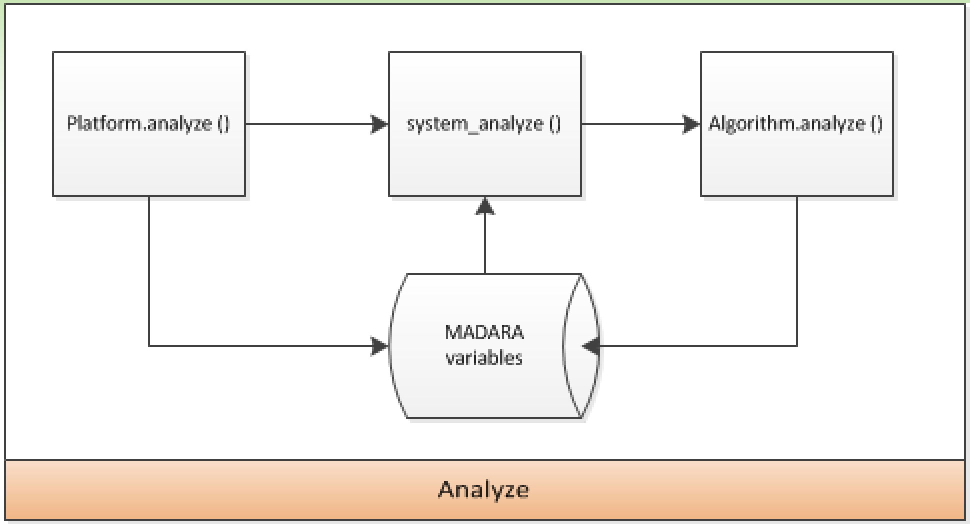






The Monitor, Plan, and Execute phases are pretty straight-forward

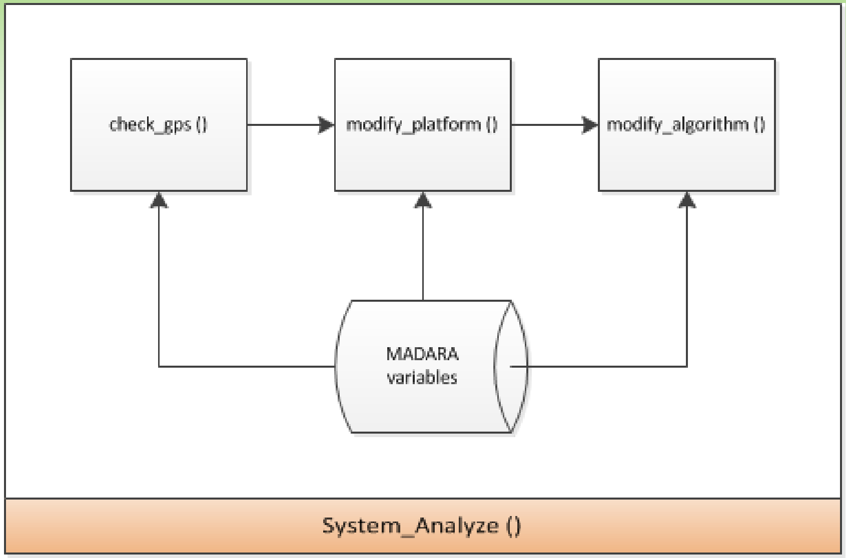
GAMS Platform and Algorithm Interactions



During the analyze phase:

1. **The platform analyzes its state and informs the rest of the GAMS system via MADARA variables**
2. **The system analyzes the platform and environment for algorithm changes**
3. **The algorithm then analyzes its state and sets appropriate MADARA variables.**

GAMS Platform and Algorithm Interactions



About system_analyze ():

1. **The platform can inform the control loop of gps-spoofing, if it has capabilities**
2. **Check_gps () is also intended to implement gps-spoof checking in software**
3. **Environmental or platform characteristics can result in changes to the platform (e.g., an arm is damaged) or algorithm (e.g., the UAS should return home)**

How to use GAMS with new platforms and algorithms

1. Extend the platform base class
   - Implement move (), land (), takeoff (), or other functions
   - Implement sense ()
   - Implement analyze ()
2. Extend the algorithm base class
   - Implement analyze ()
   - Implement plan ()
   - Implement execute ()
3. Extend the base controller class (optional)
   - Override MAPE methods
4. Use the parameterized Mape_Loop class (optional)
   - Use the define_monitor, define_analyze, etc. methods with MADARA functions

What exactly are we solving?

1. MADARA is a bit expansive in its capabilities and developers can find themselves pulled in many different directions when thinking of autonomy to implement. **GAMS provides an interface for algorithms and platforms to be added and utilized within a wireless environment**

2. **GAMS provides mechanisms for tracking platform and algorithm states and characteristics**, such as detection of GPS-spoofing, blocked/deadlocked conditions within algorithms, low battery, degraded sensors, etc.

3. While MADARA may support any type of distributed artificial intelligence paradigm, **GAMS provides a stable, consistent framework for group autonomous behaviors and may prove beneficial to standardization efforts for group autonomy**

Future Work

Though we have great partners and stakeholders, we do have hardware and other research that we would love to focus on in FY 2015 and 2016

- **Other gps-denied localization systems**
  - **Vision** (have discussed aspects of this with Dr. Davide Scaramuzza at University of Zurich and Dr. Marios Saviddes of CMU)
  - **Acoustic** (Anthony Rowe of CMU)
- **Accent/Augmentation algorithms**
  - Focus on control loops and structures that allow for secondary algorithms to run in parallel to accomplish autonomous functions without interfering with primary algorithms
- **Formal verification of complex, asynchronous distributed applications** and algorithms

Carnegie Mellon University

Software Engineering Institute
Carnegie Mellon

Closing remarks

In this talk, we've discussed

- A **distributed reasoning engine called MADARA** that provides portable, fast reasoning services for distributed artificial intelligence
- An **extensible framework called GAMS for distributed algorithms and platforms** that enables Monitor-Analyze-Plan-Execute-based distributed autonomous systems
- **A model-checked code compiler and prototype generator called MCDA for distributed applications**

FY 2014 Open Source Release

The algorithms, tools, and middleware created at SEI are released via BSD-style licenses through the following projects:

- Multi-Agent Distributed Adaptive Resource Allocation (MADARA) for the distributed OS layer: http://madara.googlecode.com
- Group Autonomy for Mobile Systems (GAMS) for the algorithms and UIs: http://gams-cmu.googlecode.com
- Model Checking for Distributed Applications (MCDA) http://mcda.googlecode.com
- Drone-RK for the UAV device drivers: http://www.drone-rk.org
- Contact: jredmondson@sei.cmu.edu

SEI Project Members

James Edmondson

Sagar Chaki

Sebastian Echeverria

Gene Cahill

CMU Project Members

Anthony Rowe

Oliver Shih

Luis Pinto

Vanderbilt Students

Anton Dukeman (CS)

Subhav Pradhan (ISIS)

Carnegie Mellon University

Software Engineering Institute
Carnegie Mellon