Towards a Common Controller Software Architecture

April 8, 2015

Kurt Bruck
1. Unmanned systems mission space is becoming more advanced (air/ground/sea collaboration)

2. Unmanned systems vehicles are becoming more autonomous and have a decreasing need for unique and dedicated controllers to command them.

3. The opportunity for a reduction in cost, logistics, training, and greater situational awareness with a CCA implementation grows larger as 1 & 2 trend onward.

Why a common control software architecture
Core Team

- Douglas Gregory: Neya Systems
- David Barnhard: Kutta Technologies
- Kurt Bruck: QinetiQ North America
- Bill Huff: JHNA

Team members representing the NAMC - not their individual companies.
Problems with existing common software

• **Common to a subset of departmental needs**
  • Each government organization has unique objectives for a CCA.
  • Strategy between departments vary: long-term versus short-term
  • Air domain focused departments generally leave out ground domain:
    • (E-Stop, multi-path radios, IOP/JAUS, teleop control).
  • Ground domain focused departments generally leave out the air domain
    • (Stanag 4586, complete mission planning features, general lightweight expectation).

• **Maintenance required**
  • Common control software requires a high level of maintenance in order to not become obsolete in one or more domains.

• **SDK maturity**
  • A solid SDK is critical to take advantage of innovation from industry
Scope & Application

Hardware scope – hardware independence

• Objective – All hardware
  • 8 bit microcontrollers to up 32 core server systems

• Threshold – Tactical hardware
  • ~smartphone to ~x86 quad-core

Heterogeneous Robotics and Autonomous Systems (RAS)

• Tactical air platforms (Group 1 UAS)
• Ground vehicles/systems

Representative Operational Environments

• Tactical dismounted operations (e.g. CLARK)
• Support and logistics operations (e.g. runway clearing)
The Approach

1. Top-Down
   - Evaluate High level architectures provide a solid basis to inform enterprise level decision
     - UAS Control Segment (UCS)
     - Future Airborne Capability Environment (FACE)
     - STANAG 4586 (Unmanned Aerial System messaging)
     - Joint Architecture for Unmanned Systems (JAUS – ground and maritime robotics)
   - Determine ideal, concept architecture: “blue-sky”

2. Bottom-up
   - Investigate Specific programs and prototypes provides a rapid path to “real” development
     - Navy: SPAWAR’s Multi-robot Operator Control Unit (MOCU)
     - Army: Tactical Open Government Architecture (TOGA), WMI, Nett Warrior
     - Marines: Tactical Robot Controller

3. Develop solutions for how the two meet
Top-down (ideal concept architecture)

- **AS4-JAUS** is ubiquitous amongst UGV’s
  - Strengthened by IOP standards
  - Gradual adoption by UUV’s

- **STANAG 4586** is ubiquitous amongst the UAV community
  - Soon to be a required messaging standard

- **UCS** is a well-established, Mature, well-funded
  - UCS is migrating under the AS4 umbrella
  - UCS extensions can be created to allow for interoperability between STANAG 4586 and AS4-JAUS
Bottom-Up Potential Candidate being explored: MOCU

Development of a Common Controller Architecture could be expedited using a product like MOCU

- **Strengths:** Mature, government owned product that has been used across many different robotic systems.
  - MOCU is being utilized in many different programs, including the Army’s TOGA program.

- **Weakness:** The way MOCU is implemented works well for the Navy, but is not modular enough to suit the needs of industry and the larger robotics enterprise

- **Conclusion:** MOCU would require industry and government support to enhance its architecture to more directly align with the UCS architecture and business model

- **Impact:** MOCU, in its present form, is not a ready made solution for the CCA, but is very close

- **Risk Mitigation:** Revise MOCU architecture to fundamentally align with UCS architecture and create transition package including data use rights, etc. for industry.
• Implement UCS service interfaces with MOCU component architecture
Option 1 Advanced implementations

- MOCU/UCS “thin-client” resides as a component within Nettwarrior software framework
- Host MOCU/UCS on MCWL Tactical Robot Controller
- Host MOCU/UCS on TOGA controller
- Implement TARDEC’s WMI presentation layer as a UCS component within MOCU
Next Steps

• Decide on 2-3 key architectural approaches
• Develop a prototype
• Test component elements on existing hardware