Verification of Collaborative Autonomous Systems for Multi Domain Operations Using a DevSecOps Model

Apr 1 2020

Rob Murphey, Ph.D., Principal Research Engineer, Georgia Tech Research Institute (GTRI)
Outline

• Distributed Multi-Domain Warfare

• Digital Advanced Battle Management System (ABMS) and the Electromagnetic Spectrum (ES) “Domain”

• Autonomy As a Means to Gain ES Dominance

• Concepts of DevSecOps

• Autonomy Software Factory

• Continuous Testing
Terms of Multi-Domain

• **Domain.¹**

  “Maneuver in a domain is often a unique, defining feature that separates domains from one another.”

  Domain: “A critical maneuver space whose access or control is vital to the freedom of action and superiority required by the mission.” Traditionally DoD domains are air, space, land and sea. Cyber has recently been added to the list and sometimes generalized to Electromagnetic Spectrum.

• **Multi-Domain Operations (MDO).²**

  MDO is more than systems in one domain supporting operations in another domain (necessary but not sufficient). MDO are high velocity, operationally agile operations that present multiple dilemmas for an adversary at an operational tempo they cannot match.

  **Multi-Domain Command and Control (MDC2).** Seamless, dynamic and continuous integration of capabilities generating effects in and from all domains

---

2. Brig Gen Chance “Salty” Saltzman, MDC2 Overview, 2018 C2 Summit,
2019 Dr. Roper (SAF/AQ) appointed Preston Dunlap to be AF Chief Architect. First order of business was to recast ABMS as Digital Solution for MDC2. Approach: Build capability through 28 “thin” builds that complement one another.

How do we validate this system?
Expanded Ideas of ‘Domain’

**Electromagnetic Spectrum (ES) Domain.**

- In 2011, the DoD recognized the strategic significance of managing the disparate missions of communication and control, strike, surveillance, navigation, and electronic warfare as a single, integrated ES “maneuver space”.

  - New strategic domain spans the traditional warfighting domains of air, space, land and sea, enabling Multi Domain Operations (MDO) and Command and Control (MDC2).
  - Managed across many platforms (manned and unmanned) with many unique ES enabled payloads across vast distances in land, sea, air and space.

- **Dominance in ES domain:** Extending ES-enabled platforms some degree of autonomy and providing them with the ability to collaborate.

Autonomy Play Calling Concept

Provide a collaborative autonomous system of agents with flexibility to adapt to the environment, threats or changes in priorities but ensure they obey certain rules and are focused on achieving objectives.

- **Play calling** – A supervisory state machine evaluates information from the agent’s sensors, avionics and network and based upon that information will call a play to be executed. The play is simply some predefined (parametric) behavior.

- **Collaborative Task Negotiation.** If a play requires multiple agents with prescribed roles, then the agent may negotiate with other agents to adopt the play and assume roles in the play. Often this is accomplished using a market based auction but could be any method including a state machine.

**Benefits of a Playbook:**
- Information used to trigger a play can be from anywhere and take any form.
- Plays can be restricted using constraints described in time, space, or state space.
- Plays can be called with humans (Man-Machine Teaming) or without (Machine-Machine Teaming).
- Task negotiation can be decentralized or hierarchical or a hybrid.
The DevSecOps Software Factory

Autonomy services are, in essence, *Software*. So we should be looking for ideas that promote efficiencies in building and testing Software: **DevSecOps factories.**

- 2019 Nicolas Chaillan appointed to be AF Chief Software Officer by Dr. Roper (SAF/AQ)

- 2019 Dan Deasy appointed to be Chief Information Officer of OSD.
  - In cooperation, AF and OSD immediately issued guidance on moving to a DevSecOps approach.  
  - AF created and staffed LevelUp and PlatformOne to assist programs in creating DevSecOps Software Factories
  - Software Factory methods now integrated with the Digital ABMS approach.


---

**What is DevSecOps?**

A compound of development (Dev) security (Sec) and operations (Ops), DevSecOps is the union of people, process, and technology to continually provide secure software services.

**3 key ideas for autonomy software:**

- Breaking down monolithic autonomous software services into multiple loosely coupled “microservices”
- Containerizing microservices for portability and baked in security
- Performing continuous integration, continuous testing, continuous delivery

---

A Software Factory Approach for Multi-Domain Autonomy

- DevSecOps pipeline model provides products that naturally span multiple missions and then integrate in deployment
  1. Produce novel and innovative algorithms.
  2. Transforms these into production software.
- Services become domain (air, land, sea, space) agnostic thus naturally enabling MDC2

6 months - 3 years: S&T to create new behaviors

Hours to weeks: tune behaviors & triggers
Microservices API Architecture

What are Microservices?

“An approach to structuring systems as a collection of loosely-coupled services. Goes hand-in-hand with small, autonomous teams that develop, deploy, and scale their services independently.”*

Decomposes an application into single-function modules with well-defined application programming interfaces (APIs).

*https://www.atlassian.com/software-development/practices/microservices

Well designed microservices and APIs are key to enabling low latency applications and fast, efficient testing.
Deploying Autonomy Microservices to Hardware

Each microservice runs in its own container

Autonomy code is built in microservice pipelines

- Multi Vehicle Control (MVC)
- Electronic Attack Effects (EAE)
- Non-Electronic Effects (NEE)
- Electronic Support Perception (ESP)
- Surveillance & Reconnaissance Perception (SRP)
- Seeker-Target Perception (STP)
- Discover, Access, Communicate (DAC)
- Continuous ATO Planning (CAP)
- Blended Mission (BM)
- Metrics and Logging (ML)
- Deliver Dispense Payload (DDP)

Multiple containers can run on the same OS

Containers are mesh deployed and coupled & secured with sidecars or a service mesh

Adapters to platform resources
Integrate “Digital ATO Enterprise” into autonomy services factory

- Data collection for machine learning of new autonomy microservices
  - ML requires massive data sets – cannot rely exclusively on simulated data
  - How do we “backhaul” operational data?
    - Must be sparse – challenged comms
    - Inverse reinforcement learning, transfer learning
    - Digital twin, “IoT on a weapon” linked to edge service

- Enhanced delivery pipelines to support continuous ATO
  - How do we push new microservices to the field at anytime and anywhere?
    - Algorithms to Playbook Production – CI/CD Dev Pipelines
    - Mission data, playbooks – CI/CD Ops Cloud & Edge
What is Continuous Verification Testing?

Testing Throughout Development. Learning throughout Operations.

- **Unit Build Testing.** Prioritized by risk. Mostly automated.
- **Functional Validation.** Ensure the functionality expressed in user stories works as expected.
- **Integration Testing.** Microservice integration is through APIs. So API testing becomes critical.
- **End-to-End Regression Testing.** Last step prior to fielding (Unit Acceptance Test).

<table>
<thead>
<tr>
<th>CT Practices</th>
<th>Why it Matters</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Chunk” change requests</td>
<td>Smaller units of code are easier and faster to build and verify</td>
</tr>
<tr>
<td>Shift left</td>
<td>Pulling production environments into early testing heads-off errors when cheaper to fix</td>
</tr>
<tr>
<td>Use “stubs” to accelerate testing</td>
<td>Simulated stand-ins for missing (hard) pieces help define the missing piece</td>
</tr>
<tr>
<td>Automate testing process</td>
<td>Automate pull requests to build, log defects, ... speeds up time to fielding</td>
</tr>
<tr>
<td>Automate testing</td>
<td>Computer built scripts &amp; test execution have fewer errors, more efficient use of test resources</td>
</tr>
<tr>
<td>“Canary” deployments</td>
<td>Deploying to small parts of the fielded system allow in situ eval without system-wide risk</td>
</tr>
<tr>
<td>Monitor fielded system</td>
<td>Automated digital engineering reliably captures user stories, bug reports, analytics, ML</td>
</tr>
</tbody>
</table>
A Note on Continuous Validation Testing (not yet solved)

• **Validation is strongly linked to the user**
  - Does it do what the user needs it to do (or not do)?

• **Validation is typically performed at the end of the testing cycle.**

• **DevSecOps methods: an opportunity to start validation much earlier.**
  - We could use “user stories” at functional level for incremental validation.
  - Short development iterations on small code changes with continual feedback from users: In the limit this is “continuous validation.”
Challenges Remain.
Multi Domain Autonomy as a Cyber Physical Deployment Problem.

- Devices (like IoT) exist on cloud edges.
- Edge devices create and consume data.
- Idea of Edge Computing is to compute at the edge. **This is autonomy.**
- However, we still want the benefit of the cloud
  - Analytics
  - Learning
  - Centralized development

Expect the demand for autonomy services to scale just as IoT and 5G will likewise need to scale. How will we test this?
Questions?

Thank you for your attention!

Robert Murphey, Ph.D.,
Principal Research Engineer,
Aerospace, Transportation and Advanced Systems Laboratory (ATAS)
Georgia Tech Research Institute (GTRI)
1270 N. Eglin Pkwy, Ste A-15
Shalimar, FL 32579
Robert.murphey@gtri.gatech.edu
Office: (404) 407-7196
Cell (470) 462-8987