Expanding DevOps Principles to Large Complex Solutions

October 2019

Robin Yeman

Lockheed Martin Corporation

Robin.Yeman@lmco.com

Dr. Suzette S. Johnson

Northrop Grumman Corporation

Suzette.Johnson@ngc.com

Introductions

Robin Yeman **Suzette Johnson**

Intent of Industrial DevOps

- How to scale DevOps practices across large complex systems composed of hardware, firmware, and software
- Address the misconception is that this form of rapid iteration and flow is only for software, or small applications or systems
- Provide an extended definition for DevOps, and provide recommendations

Why?

Industrial DevOps Applied

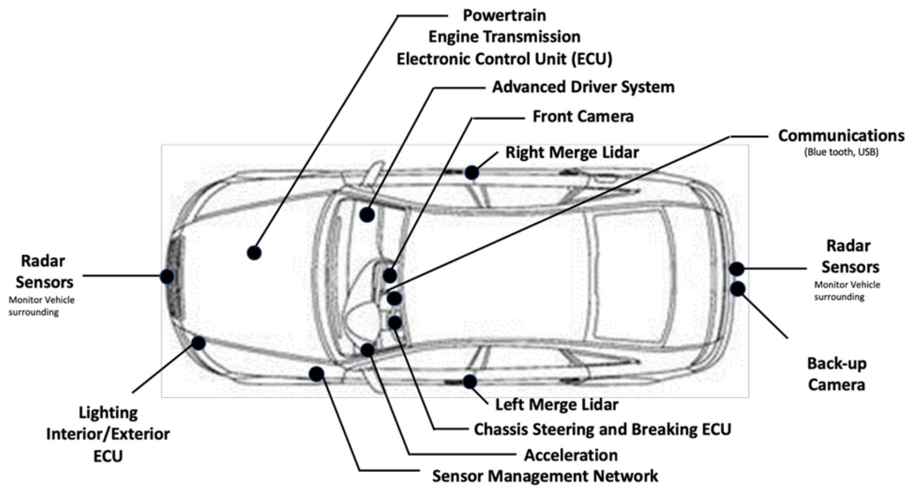
Companies like Lockheed Martin and Northrop Grumman build life human safety critical like F-35 and the B-2

F-35

B-2

Example: **Autonomous Vehicle**

Autonomous vehicles have similar complexity and human safety details as many of the products that Lockheed Martin and Northrop Grumman currently do.



Scenario: Alset Transport Goal

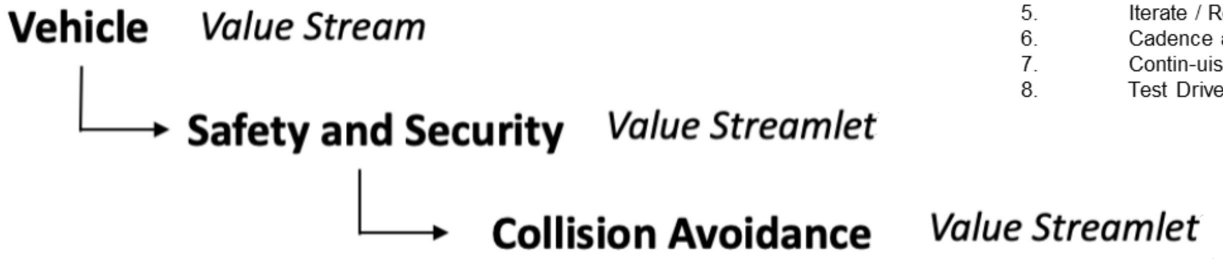
- To improve the collision-avoidance capability by increasing the obstacle-detection system’s actionable closing distance by 50% of the current operating distance.
- The enhancement of the collision-avoidance capability will be handled through a combination of incremental updates made to:
  - sensor firmware
  - control-system software
  - user-interface software
- Two core epics in the product backlog:
  - Enhance obstacle-detection software or firmware in existing vehicles.
  - Update camera and associated technology in future vehicles.

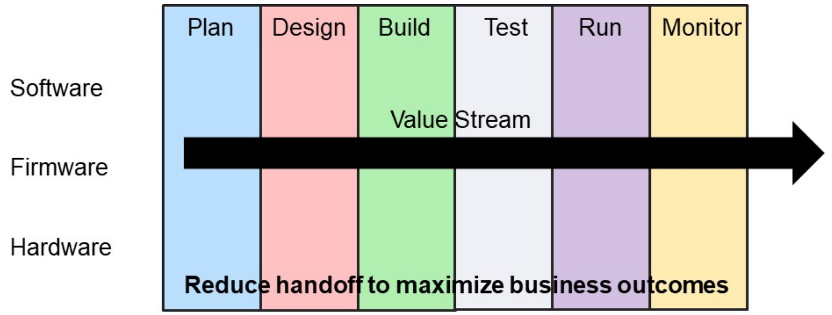
Industrial DevOps Principles

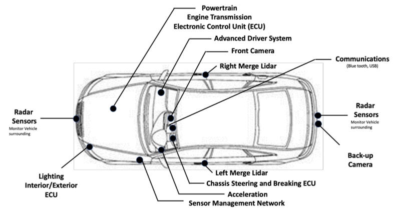
1. Visualize and organize around the value stream
2. Multiple Horizons of Planning
3. Base decisions on objective evidence of system state and performance
4. Architect for Scale, Modularity, and Serviceability
5. Iterate / Reduce batch size / Get fast feedback
6. Cadence and Synchronization
7. Continuish Integration
8. Test Driven Development

Organize Around the Value Stream

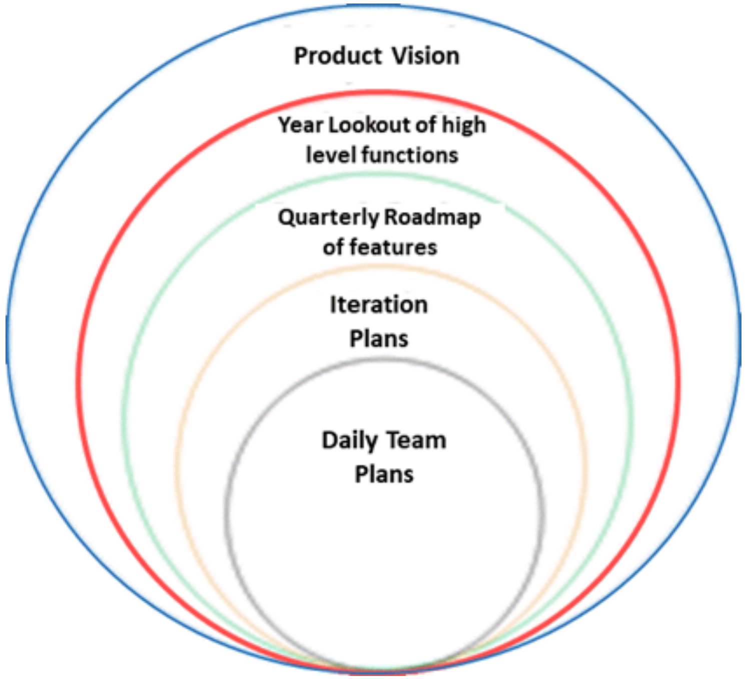
1. Visualize and organize around the value stream
2. Multiple Horizons of Planning
3. Base decisions on objective evidence of system state and performance
4. Architect for Scale, Modularity, and Serviceability
5. Iterate / Reduce batch size / Get fast feedback
6. Cadence and Synchronization
7. Contin-uish Integration
8. Test Driven Development







Multiple Horizons of Planning


Product Vision
Year Lookout of high level functions
Quarterly Roadmap of features
Iteration Plans
Daily Team Plans

1. Visualize and organize around the value stream
2. Multiple Horizons of Planning
3. Base decisions on objective evidence of system state and performance
4. Architect for Scale, Modularity, and Serviceability
5. Iterate / Reduce batch size / Get fast feedback
6. Cadence and Synchronization
7. Contin-uish Integration
8. Test Driven Development

Short Term MVP's
To
Long Lead Items

Apply Decisions Based on Objective Evidence

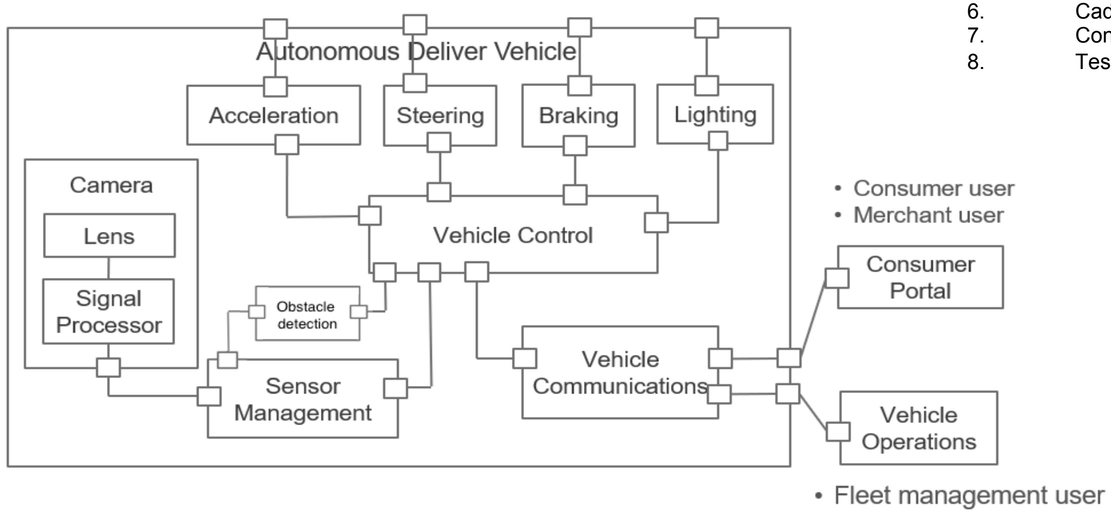
	Time Horizon	Capability	Evidence
epic	Annual	Enhance obstacle detection through updates to sensor types; refactoring architecture	Drive vehicle through multiple scenarios to validate sensor types; Evaluate deployment rate for new updates
feature	Quarterly	Enhanced Lidar sensor color profile	View colors in simulator to verify improvement
User story	Iteration	Split Lidar by component value	Validate demonstration of Lidar split by through test of
task	Day	Update cloud point extents in ESRI	CI/CD Pipeline has identified no errors with change

1. Visualize and organize around the value stream
2. Multiple Horizons of Planning
3. Base decisions on objective evidence of system state and performance
4. Architect for Scale, Modularity, and Serviceability
5. Iterate / Reduce batch size / Get fast feedback
6. Cadence and Synchronization
7. Contin-uish Integration
8. Test Driven Development

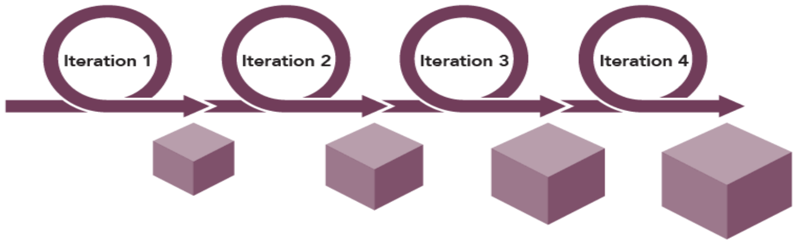
Evidence at each level

Applying Architecture for Modularity

1. Visualize and organize around the value stream
2. Multiple Horizons of Planning
3. Base decisions on objective evidence of system state and performance
4. Architect for Scale, Modularity, and Serviceability
5. Iterate / Reduce batch size / Get fast feedback
6. Cadence and Synchronization
7. Contin-uish Integration
8. Test Driven Development



Iterate and reduce batch size



1. Visualize and organize around the value stream
2. Multiple Horizons of Planning
3. Base decisions on objective evidence of system state and performance
4. Architect for Scale, Modularity, and Serviceability
5. Iterate / Reduce batch size / Get fast feedback
6. Cadence and Synchronization
7. Contin-uish Integration
8. Test Driven Development

Epic 1: Software-Only Updates to Existing Fleet Vehicles

- Sensor system with an available test environment
- Simulated environment and small, code-based sensors, such as a forward and backup camera.

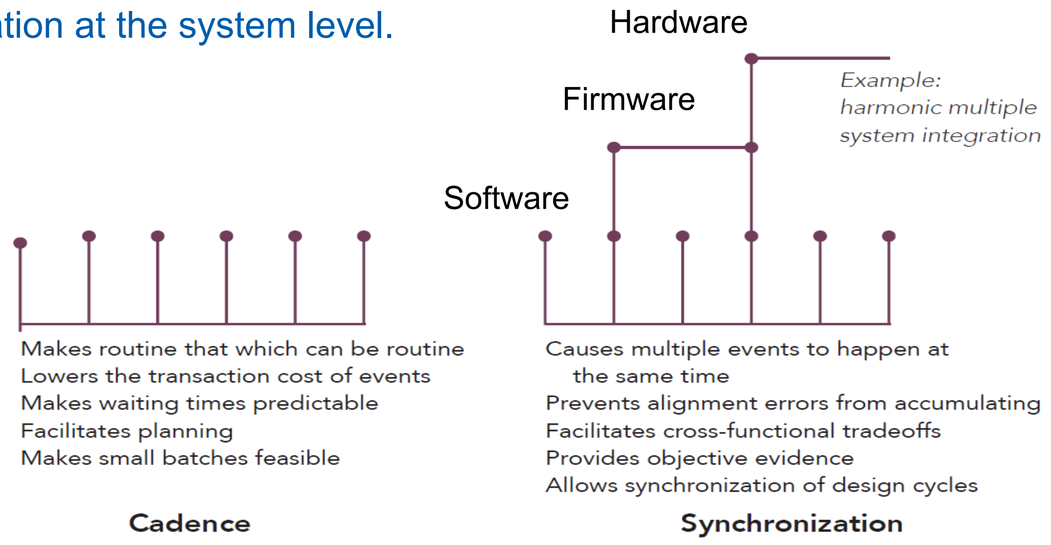
Epic 2: New Camera

- New functionality for radar sensors and using previously generated tests on a radar simulator to check the new code deployment.
- Code integration with the camera and radar system and resolve any integration errors.
- Upon successful demonstrations in the simulated environment, the team tests the firmware and software updates on a couple production cars.

Applying Cadence and Synchronization

Both teams established a cadence of regular quarterly planning and iterations at a two-week period.
Regular synchronization occurs by conducting demonstrations at the end of each two-week iteration at the system level.

1. Visualize and organize around the value stream
2. Multiple Horizons of Planning
3. Base decisions on objective evidence of system state and performance
4. Architect for Scale, Modularity, and Serviceability
5. Iterate / Reduce batch size / Get fast feedback
6. Cadence and Synchronization
7. Contin-uish Integration
8. Test Driven Development


Hardware
Firmware
Software
Example: harmonic multiple system integration
Makes routine that which can be routine
Lowers the transaction cost of events
Makes waiting times predictable
Facilitates planning
Makes small batches feasible
Cadence
Causes multiple events to happen at the same time
Prevents alignment errors from accumulating
Facilitates cross-functional tradeoffs
Provides objective evidence
Allows synchronization of design cycles
Synchronization

Apply Continu-ish Integration

Epic 1: Software-Only Updates to Existing Fleet Vehicles

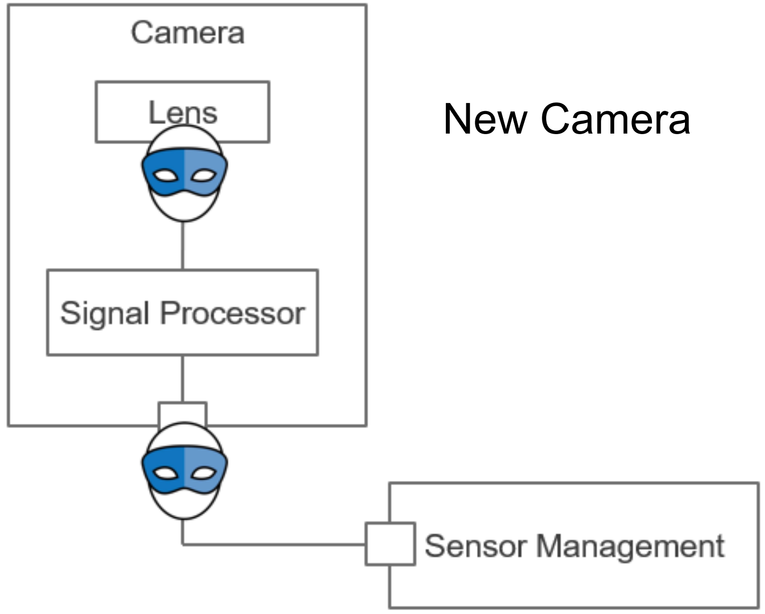
- Digital twin and test environment make developer-level testing cheaper and faster; code-level integrations can happen routinely, daily, or even hourly
- Routine DevOps practices of source code control, automated builds, and automated build verification tests apply well in this case

1. Visualize and organize around the value stream
2. Multiple Horizons of Planning
3. Base decisions on objective evidence of system state and performance
4. Architect for Scale, Modularity, and Serviceability
5. Iterate / Reduce batch size / Get fast feedback
6. Cadence and Synchronization
7. Contin-uish Integration
8. Test Driven Development

Epic 2: New Camera

- Specs provided
- Build a camera-emulator software device which feeds simulated camera data in via the predetermined API and protocols. Deploy the device into the test bed and on a test vehicle to allow new algorithm development for enhanced obstacle detection without the physical device. May also employ field-testing mocks on an actual test vehicle.
- The mechanical design is tested with mechanical mock-ups, which are consistent with the intended physical properties of the camera. The supplier provides pilot-camera hardware in the test environment.

Apply Test Driven



New Camera

1. Visualize and organize around the value stream
2. Multiple Horizons of Planning
3. Base decisions on objective evidence of system state and performance
4. Architect for Scale, Modularity, and Serviceability
5. Iterate / Reduce batch size / Get fast feedback
6. Cadence and Synchronization
7. Contin-uish Integration
8. Test Driven Development

- Model-based systems engineering (MBSE) and computer-aided design (CAD) environments for electrical and mechanical development provide rich support for testing and validating designs.
- Mocks and test doubles isolate hardware changes for early evaluation and testing.

Test-driven development applies to software and hardware development.

Contributors

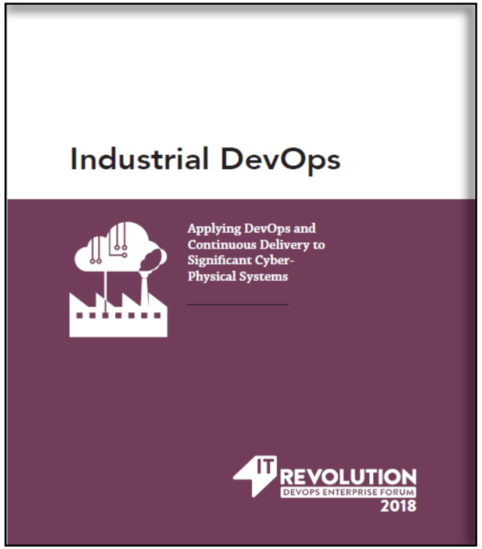
- Josh Atwell jatwell@splunk.com
- Ben Grinnell ben.grinnell@northhighland.com
- Dean Leffingwell, Co-founder and Chief Methodologist, Scaled Agile Inc., @deanleffingwell
- Dr. Suzette Johnson, Fellow and Agile Transformation Lead, Northrop Grumman Corporation
- Harry Koehnemann, SAFe Fellow and Principal Consultant, Scaled Agile Inc.
- Vincent Lussenburg vlussenburg@xebialabs.com
- Dr. Steve Mayner, SAFe Fellow and Principal Consultant, Scaled Agile Inc., @stevemayner
- Robin Yeman, Lockheed Martin Fellow, Lockheed Martin Corporation, @robinyeman

Help We Are Looking For

- Each year we build on this scenario….how should we continue to elaborate the story?
  - Metrics and how the product team uses their metrics for improvement and decision-making
  - Connection between the development of these two scenarios to manufacturing and handling of long lead items
  - A story that connects the people, collaboration, planning, engineering, flow, and delivery together… *Once upon a time, there was a product team*….
  - Or…what?

Industrial DevOps Publication

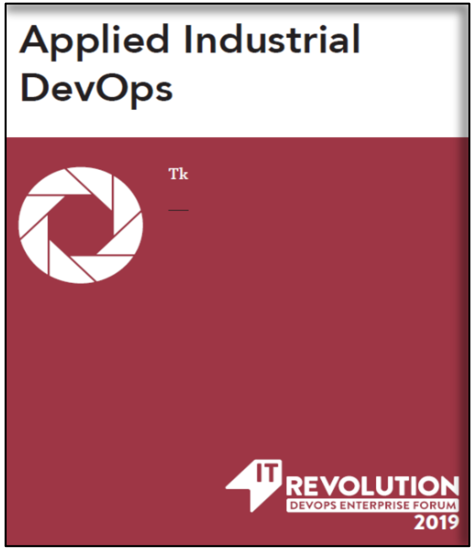
- Industrial DevOps expands the definition of DevOps outside of software to enable significant cyber-physical systems development programs to be more responsive to changing needs while reducing lead times.
- It is the application of continuous delivery and DevOps principles to the development, manufacturing, deployment, and serviceability of significant cyber-physical systems.


Industrial DevOps

Applying DevOps and Continuous Delivery to Significant Cyber-Physical Systems

IT REVOLUTION DEVOPS ENTERPRISE FORUM 2018

Defining the Principles


Applied Industrial DevOps

Tk

IT REVOLUTION DEVOPS ENTERPRISE FORUM 2019

Applying the Principles

https://itrevolution.com/book/industrial-devops//
https://itrevolution.com/book/applied-industrial-devops/

Summary

Leveraging the power of Industrial DevOps for large complex systems is an industry step change and the companies that solution this problem first will increase transparency, reduce cycle time, increase value for money, and innovate faster.

QUESTIONS?