Applied Industrial DevOps

Practical Guidance for the Enterprise

IT REVOLUTION
DEVOPS ENTERPRISE FORUM
2019

25 NW 23rd Pl
Suite 6314
Portland, OR 97210

Applied Industrial DevOps

Produced in the United States of America

Cover design and interior by Devon Smith

For further information about IT Revolution, these and other publications, special discounts for bulk book purchases, or for information on booking authors for an event, please visit our website at ITRevolution.com.

The 2019 DevOps Enterprise Forum was sponsored by XebiaLabs.

Approved for Public Release; Distribution is Unlimited; #19-1755; Dated 09/11/19

Preface

In May of this year, the fifth annual DevOps Enterprise Forum was held in Portland, Oregon. As always, industry leaders and experts came together to discuss the issues at the forefront of the DevOps Enterprise community and to put together guidance to help us overcome and move through those obstacles.

This year, the group took a deeper dive into issues we had just begun to unpack in previous years, providing step-by-step guidance on how to implement a move from project to product and how to make DevOps work in large-scale, cyber-physical systems, and even a more detailed look at conducting Dojos in any organization. We also approached cultural and process changes like breaking through old change-management processes and debunking the myth of the full-stack engineer. And of course, we dived into the continuing question around security in automated pipelines.

As always, this year's topics strive to address the issues, concerns, and obstacles that are the most relevant to modern IT organizations across all industries. Afterall, every organization is a digital organization.

This year's Forum papers (along with our archive of papers from years past) are an essential asset for any organization's library, fostering the continual learning that is essential to the success of a DevOps transformation and winning in the marketplace.

A special thanks goes to Jeff Gallimore, our co-host and partner and co-founder at Excella, for helping create a structure for the two days and the weeks that followed to help everyone stay focused and productive. Additional thanks goes to this year's Forum sponsor, XebiaLabs. And most importantly a huge thank you to this year's Forum participants, who contribute their valuable time and expertise and always go above and beyond to put together these resources for the entire community to share and learn from.

Please read, share, and learn, and you will help guide yourself and your organization to success!

—Gene Kim
June 2019
Portland, Oregon

Collaborators

Josh Atwell
Senior Technology Advocate, Splunk

Ben Grinnell
Managing Director, North Highland

Dr. Suzette Johnson
Fellow and Agile Transformation Lead, Northrop Grumman Corporation

Harry Koehnemann
SAFe Fellow and Principal Contributor, Scaled Agile Inc.

Dean Leffingwell
Co-Founder and Chief Methodologist, Scaled Agile Inc.

Vincent Lussenburg
Director of DevOps Strategy, XebiaLabs

Hasan Yasar
Interim Director, Software Engineering Institute

Robin Yeman
Lockheed Martin Fellow, Lockheed Martin Corporation

Reviewers

Dr. Jeff Boleng
Special Assistant for Software Acquisition to the Undersecretary of Defense for Acquisition and Sustainment, Software Engineering Institute

Eileen Wrubel
Co-Lead of the SEI's Agile/DevOps Transformation, Software Engineering Institute

Introduction

As DevOps continues to challenge the status quo and improve business outcomes for software systems, many of the world's larger enterprises are working to identify how to scale these practices across large, complex systems composed of hardware, firmware, and software. It is important that companies have the ability to iterate and deploy faster, allowing them to adapt to changing needs, to reduce cycle time for delivery, to increase value for money, to improve transparency, and to leverage innovations.

In a previously published DevOps Enterprise Forum paper from 2018, *Industrial DevOps: Applying DevOps and Continuous Delivery to Significant Cyber-Physical Systems*, we described a set of principles that development organizations can use to bring the power of DevOps to the build and maintenance of large-scale, cyber-physical systems such as vehicles, robots, complex medical devices, defense weaponry, and others. We introduced the term "Industrial DevOps" to expand the definition of DevOps in order to enable significant, cyber-physical system development programs to be more responsive to changing needs while also reducing lead times. This guidance has helped to establish the feasibility of using DevOps practices to more efficiently build, deploy, and maintain some of the world's most important—and most complex—systems.

In this paper, we take the original guidance concept of Industrial DevOps, eight supporting principles, and the subsequent definitions one step further, by applying the principles in the context of a hypothetical example using autonomous cars and then relating it back to those governing principles. Our intent is to help readers better understand the applicability and need for Industrial DevOps in different solutions, beyond those that are strictly software.

To illustrate how the practices work in a lifelike example, we have utilized a fictitious company known as Alset Transport, who produces

vehicles to support assisted and autonomous driving in both the commercial and consumer markets. Alset Transport needs to manage challenges including multiple competitors, rapidly changing technology, regulatory compliance, and safety concerns.

This example will focus on the safety technology for the autonomous vehicle, specifically the collision-avoidance capability that uses obstacle detection, vehicle control, and sensor management. The collision-avoidance capability is key to Alset Transport's continued success in the market. It uses sensors to detect an impending impact from any object that comes too close to the vehicle, based on a previously proven set of algorithmic parameters. Successful collision avoidance is directly tied to safety, regulatory compliance, and consumer trust in the vehicle. Any changes to collision avoidance will directly impact Alset Transport's future sales.

Use Case Overview

Business Objective

Alset Transport produces vehicles to support assisted and autonomous driving in both the commercial and consumer markets. Their success in the commercial space has quickly increased interest in their consumer models. As a result, Alset is eager to implement enhancements to their new vehicles, as well as to vehicles currently in active operation.

The first significant enhancement is targeted toward improving the proactive and reactive collision-avoidance capabilities of their vehicles. This improvement is driven by a desire to increase their vehicle safety rating and to address consumer feedback on braking behavior. Successful deployment of this improvement will increase customer satisfaction and consumer demand, as well as provide a more preferred platform for autonomous, commercial fleet vehicles. It is anticipated that the improvements will increase revenue by 5–8% annually over the next three years.

The Approach

Alset Transport's goal is 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

The Alset teams are going to articulate requirements through the definition of two core epics in the product backlog:

- Enhance obstacle-detection software or firmware in existing vehicles.
- Update camera and associated technology in future vehicles.

The epics and associated hypotheses are defined in Figure 1. The teams will implement epics by applying principles of Industrial DevOps throughout execution, deployment, and operations. We further define the epics that will be implemented in Table 1.

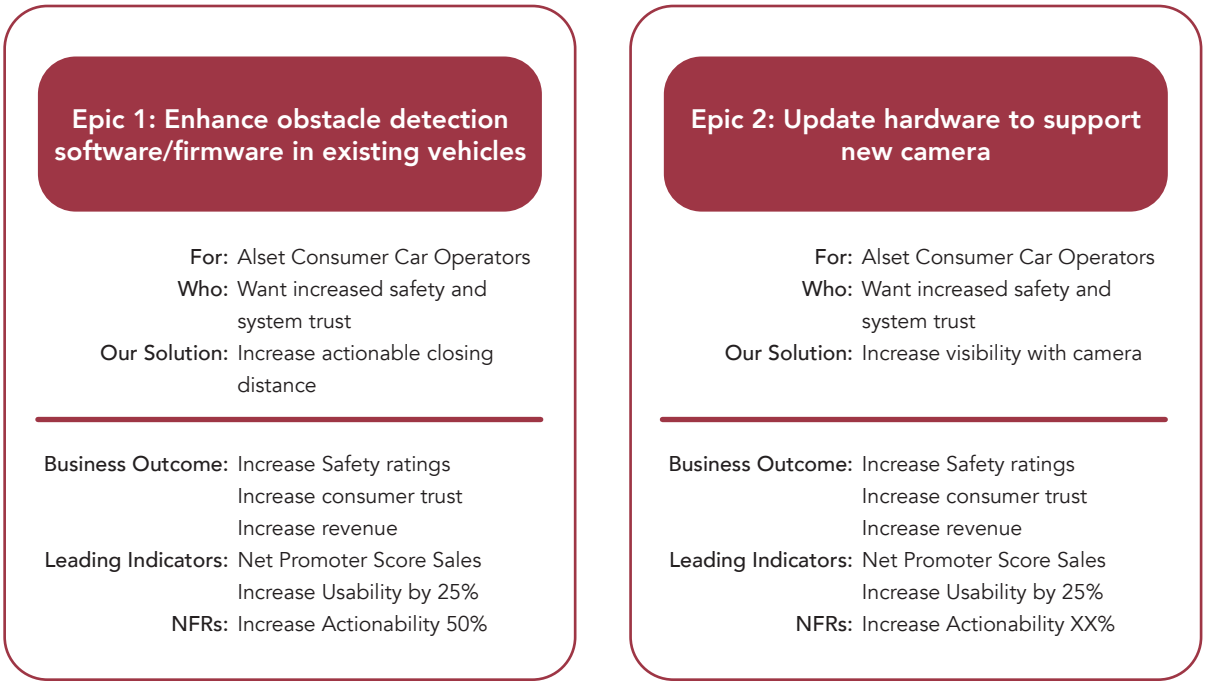

Figure 1: Epic Hypothesis

	Epic 1	Epic 2
Value Streamlet	Collision avoidance	Collision avoidance
Title	Software-only updates to existing fleet vehicles	New camera
Systems Impacted	Obstacle detection, vehicle control, sensor management	Chassis, vehicle control, camera
Activities	• Update sensor types • Refactor architecture to increase modularity • Testing/integration • IT security updates • Regulatory evaluation	• Update sensor mount • Hardware interoperability • Materials evaluation • Testing/integration
Change Type	Software, firmware	Software, hardware
Downstream Features Impacted	Cruise control, lane detection, parking assist	Lane detection, parking assist

Table 1: Epic Descriptions

How Does the Use Case Impact the Principles in Industrial DevOps?

The following is the list of Industrial DevOps principles generated from our first paper. We will walk through each of these principles to discuss in greater detail how this applies to an autonomous vehicle, which includes hardware and software components.

- Visualize and organize around the value stream.
- Use multiple horizons of planning.
- Base decisions on objective evidence of system state and performance.
- Architect for scale, modularity, and serviceability.
- Iterate and reduce batch size.
- Establish cadence and synchronization.
- Employ "continu-ish" integration.
- Be test-driven.

Visualize and Organize around the Value Stream

A "value stream" can be defined as a sequence of activities required to architect, design, build, test, and deploy a product or service that delivers value to a consumer. For large, complex cyber-physical systems, we further decompose value streams into value streamlets. A value streamlet is defined as a smaller value stream that feeds into a larger one. The next step is to organize a cross-functional team around the value streamlet. This team should have all the skills needed to implement changes from definition to operations in order to independently deliver user value. The benefits of cross-functional teams organized around the value streamlet are shorter lead times and higher quality levels. In many cases, some teams will be more hardware-centric while others will be more software-centric.

In reference to our example of Alset Transportation building an autonomous vehicle, we employ the value stream concepts. The steps to provide an autonomous car to a customer are a value stream. An autonomous car is a complex system; therefore, we can further decompose into the interoperable value streamlets, such as the body

and frame, chassis, power train, infotainment, and safety and security. The safety and security value streamlet can be broken down into multiple streamlets of their own, including collision avoidance, electronic stability control, active head constraints, etc. Our collision-avoidance streamlet impacts multiple vehicle systems, including obstacle detection, vehicle control, and sensor management.

Alset Transport has structured their agile teams around value streamlets to plan and organize their work in order to have independent, deployable components with synchronized integration points. (See Figure 2.)

Figure 2: Synchronized Integration Points

In the case of our example, both Epic 1 and 2 are delivering value through the same streamlet: collision avoidance.

Epic 1: Software-Only Updates to Existing Fleet Vehicles

The first collision-avoidance team is focused on making software changes to the existing fleet. Teams will refactor the architecture to increase modularity, incrementally update sensor types, improve sensor management refresh rate, and more. Given the nature of the changes, we selected a more software-focused collision-avoidance team to deliver Epic 1's capabilities.

Epic 2: New Camera

The next collision-avoidance team is focused on the new camera and the corresponding required vehicle updates. Physical models of the system are shown in Figures 3 and 4. The team needs to make updates to both the forward and backup cameras, which impact the hardware, firmware, and software in the collision-avoidance stream-

let. Risk will be reduced by making and evaluating changes in a digital twin: a full digital model of the vehicle.

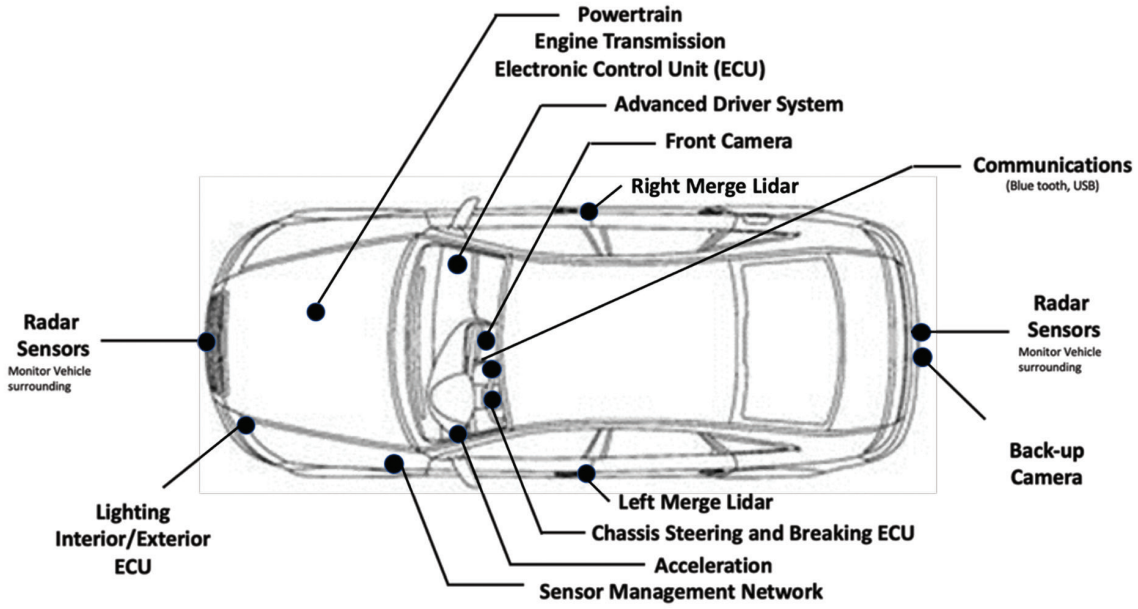

Figure 3: Overview of Vehicle

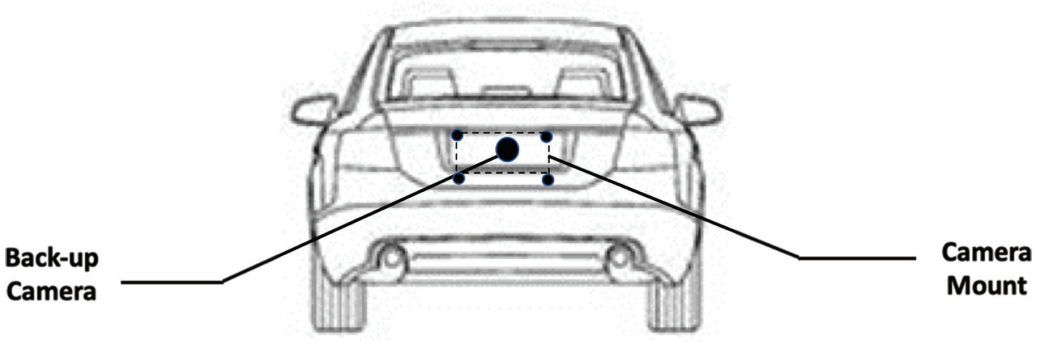

Figure 4: Rear Camera

Multiple Horizons of Planning

The Agile influence in DevOps emphasizes multiple horizons of planning, typically containing a high-level vision and program plan, which can be many years for large, complex cyber-physical systems. The program plan is further broken down into an

annual, quarterly, sprint, and daily plan, as outlined in Figure 5. Each horizon of planning informs the next, allowing teams to adjust and adapt based on what they have learned as well as supporting forecasting of capability delivery.

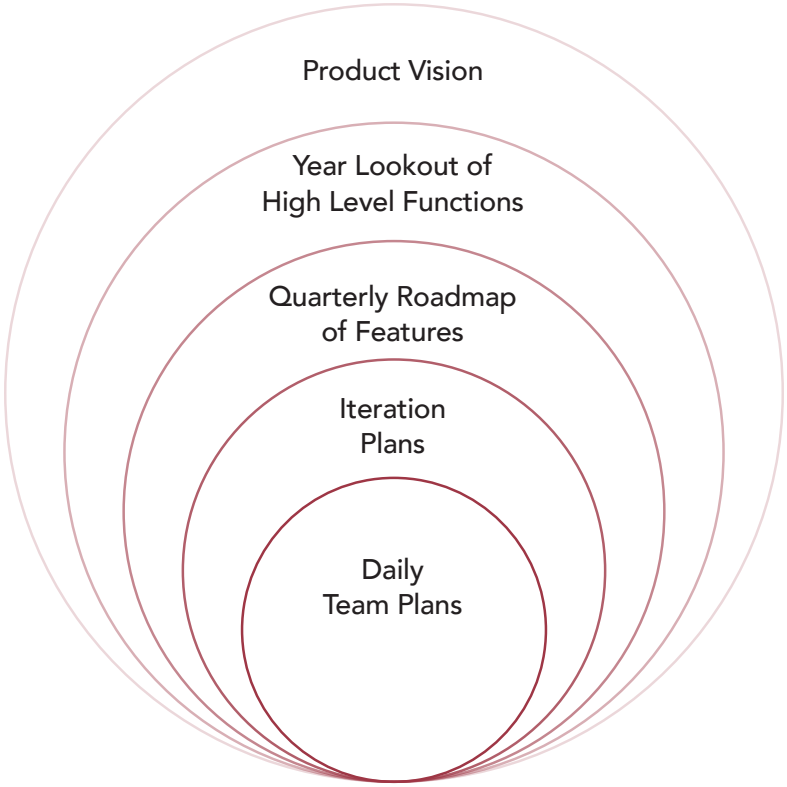

Figure 5: Multiple Planning Horizons

Through each of the planning horizons, teams focus on the principles of DevOps and how to create flow, gain feedback from stakeholders and users, and take advantage of the learning from each iteration across a massive value stream. But, what does it mean to deliver frequently with products that can take years to build? It means breaking down the products into smaller components, engaging stakeholders to determine priorities, and delivering those broken-down parts over multiple horizons of time. Our approach supports companies like Alset Transport who need longer planning horizons to communicate with stakeholders and to account for hardware lead times while enabling their teams to develop components through smaller batch sizes with frequent integration for rapid learning.

Product Vision and High-Level Horizon Plan

All products begin with a vision and a high-level plan to deliver value. The vision and high-level plan for Alset Transport includes improving the existing fleet of vehicles already in production and providing enhancements for the next fleet of vehicles in the idea and early concept phases.

Annual Plan

Large, complex cyber-physical systems can take many years to complete. Teams decompose the high-level plans into annual plans in order to simplify problems and provide more focus on what they are going to build first.

Alset Transport creates their annual plan by taking a holistic systems view and evaluating its various components and supporting suppliers. They have selected two epics that will focus on refactoring the system architecture for modularity, increasing sensor types, and improving the recognition and reaction behaviors associated with braking by incorporating a new camera. The hypothesis associated with the epics states that these changes will improve customer satisfaction, safety ratings for their vehicles, and the platform for supporting autonomous, commercial fleet vehicles.

Quarterly/Increment Plan (Or Slightly Smaller Increments)

Annual plans can still be quite complex and difficult to manage. Teams can break down annual plans into quarterly, or slightly smaller, program increment plans. The plans are visualized through a road map identifying the features to be delivered over the next few quarters. The road map is constructed at a level high enough to provide sufficient detail for stakeholders to have an idea of the path forward while allowing for an ease of change and reprioritization.

Alset Transport further breaks down their annual plan into a quarterly plan that includes features such as enhancing the Lidar sensor color profile and improving the sensor management refresh rate.

Iteration/Sprint-Level Plans (and Communications across Agile Teams)

At the iteration or sprint level, the teams break down work and time into detailed plans and user stories. The detailed plan for each story or work item for a given iteration is further defined into step-by-step plans by the team completing the work. Their detailed plans include designs, tests using prototypes, digital twins, simulators, emulators, and acceptance criteria that they will demonstrate when the iteration ends. The team plans the work in a manner that, whether the backlog items are software or hardware, can demonstrate completeness to some functionality of an integrated system for the product owner and other stakeholders at the end of the iteration.

Alset Transport has decomposed some of their team features into user stories that include splitting Lidar by component value to obtain color saturation and explore camera interoperability.

Daily Plan

The team collaborates daily to understand what work they completed yesterday and what they are going to do today in order to complete the iteration goals.

Alset Transport teams hold daily stand-ups to provide situational awareness to the capabilities being worked on and to identify any support needed. This plan provides a short feedback loop to improve the flow of delivery.

Multiple horizons of planning impact our example in several ways. The product managers and owners refined their initial features from the quarterly road map. As plans were defined, Alset Transport's teams identified multiple dependencies and the associated risks. It's important that all team members, including those from both hardware and software, physically participate in planning events. Teams define iteration-level items that can be demonstrated via some specific acceptance criteria. As their plans are created, they identify the earliest integration points possible using the existing hardware with an intent to integrate several times per iteration.

Epic 1: Software-Only Updates to Existing Fleet Vehicles

An example of how Alset Transport's collision-avoidance team may break down work across the quarter can be seen in Table 2.

Epic	Q1 Features	Q2 Features	Q3 Features	Q4 Features
Enhance obstacle-detection software through multiple sensor types and refactor architecture for modularity	Enhance Lidar sensor color profile	Improve sensor management refresh rate Obstacle-identification accuracy	Enhanced frontal distance detection	Tamperproof vehicle's networked communication system

Table 2: Epic 1 Example: Work across the Quarter

Epic 2: New Camera

An example of how Alset Transport's collision-avoidance team may break down work across the quarter can be seen in Table 3.

Epic	Q1 Features	Q2 Features	Q3 Features	Q4 Features
Add new camera, associated technology, and hardware updates	Interoperability camera (spike) New sensor Lidar enhancement Recording playback	Procure camera's ongoing traffic surveillance Obstacle-detection system enhancement	Enhanced communications with braking-system sensors Seat belt–sensor coordination Camera prototype on vehicle	Camera instantiation Full regulatory compliance

Table 3: Epic 2 Example: Work across the Quarter

Base Decisions on Objective Evidence of System State and Performance

Each of the time horizons support the ability to base decisions on objective evidence. The evidence supplied will vary based on time, number of teams, and technology available. Teams will focus on what can be built in each two-week iteration and how it can be demonstrated in a way that allows stakeholders to interpret what they're seeing

and judge whether it meets their needs. Evidence for software, firmware, and hardware are not always demonstrated in the same manner. However, due to increasing technology improvements, many of the changes for complex, cyber-physical systems can be evaluated in a digital environment using a computer model or digital twin of the whole vehicle, emulators, and simulators.

As the teams have access to physical components, or hardware, they can begin prototyping on a sample of production cars assigned for this purpose. The ability for the team to test portions of the epic on a physical production machine supports the understanding of nuances in the user experience, the interaction between the vehicle's responsive system, and the impact to the passenger. Through regular testing of the features, teams will continue to make adjustments to ensure a safe user experience.

Each time an enhancement is needed, the change becomes part of the team's iteration plans in the form of a user story. The teams will also observe several ways that their live tests and observations differ from those on the digital twin when it is configured with the existing camera's input. This data is passed on to the team who owns the digital twin, so it can be enhanced to better reflect reality.

In the case of our example at Alset Transportation, testing the frequency and vibration of the camera may not be effective in a digital environment. Complex systems require multiple types of testing in a variety of environments. Alset Transport performs lab testing on the new camera and software while the car housing is being designed. This test uncovers a sensitivity of the new faster focus camera at a certain frequency of vibration. This early test identified the problem early, allowing this requirement to be incorporated into the design rather than being discovered at system integration when the housing is completed, thus greatly reducing cost, timescales, and rework.

Epic 1: Software-Only Updates to Existing Fleet Vehicles

An example of how Alset Transport's collision-avoidance team may evaluate work at each level is shown in Table 4.

	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 through testing
Task	Day	Update cloud-point extents in Esri	CI/CD pipeline has identified no errors with change

Table 4: Epic 1 Example: Work Level Evaluation

Epic 2: New Camera

An example of how Alset Transport's collision-avoidance team may evaluate work at each level is shown in Table 7.

	Time Horizon	Capability	Evidence
Epic	Annual	Add new camera, associated technology, and hardware	Drive vehicle through multiple scenarios with new backup camera
Feature	Quarterly	Interoperable camera-mounting components	View hardware change within digital twin across multiple cameras
User Story	Iteration	Prototype camera mount options on a model	View camera mount on model
Task	Day	Define camera models	Review list of camera models

Table 7: Epic 2 Example: Work Level Evaluation

Architect for Scale, Modularity, and Serviceability

Architecture modularity significantly impacts DevOps goals to continuously develop, integrate, deploy, and release value. Modular, component-based architectures communicate in a consistent way through well-defined interfaces and thereby reduce dependencies between components. (See Figure 6). Such components can be independently tested, released, or upgraded. As teams work to create the product road map, they concentrate on several areas using the current system architecture.

- Review the architecture for constraints.
- Identify dependencies and determine if there are enhancements that can be made to the architecture to reduce dependencies.
- Identify interfaces and data formats.
- Determine what updates need to be made in the architecture to reduce work.
- Make adjustments or changes to the concept of operations, including the supply chain, field services, operations staff, interaction models, etc.
- Capture potential vulnerabilities and security risks.

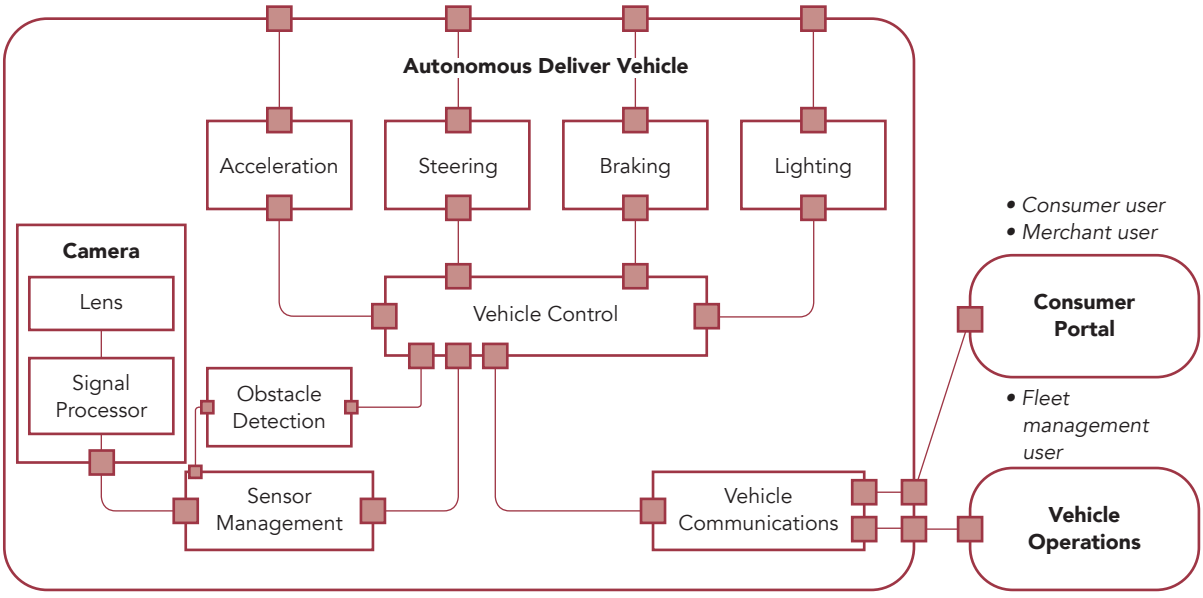

Figure 6: Components Communicate through Well-Defined Interfaces

So, how would the principle of architecting for modularity apply to the Alset Transport teams who are working on these epics?

Epic 1: Software-Only Updates to Existing Fleet Vehicles

As the team looks at the impact of Epic 1, they review the architecture of the vehicle-control software for constraints, dependencies, and risks. When the team works to improve obstacle detection, they focus on how the event is handled. This means understanding the architecture and, later, ensuring the testing of vehicle control to the reaction of the braking system and communications. The team also looks at how changes to the software impacts the obstacle-detection system and how to deliver the enhanced functionality without having to do a complete recertification of many subsystems.

Because this is a software-only update, they do not perceive any limitations implementing the improved obstacle-detection feature in the system, as it is nicely contained within the vehicle-control component. Although the algorithm updates are complex, no extraordinary constraints are encountered, and the risk of being able to deploy is minimized.

Epic 2: New Camera

As the team moves toward Epic 2, the scope of the change extends far beyond vehicle control. The new camera not only has a higher resolution but a faster focus time as well. This leads to changes in the sensor-management software and the interface with the vehicle-control module. The amount of data vastly increases, affecting the communication bus of the vehicle-control module, as well as impacting the resource usage. Both memory and CPU significantly increase. It becomes obvious that obstacle detection needs to be as near real time (NRT) as possible.

Another constraint that is identified is the impact on testing. In the future, it can be expected that the variance in cameras will increase. The processing characteristics for each camera will vary.

As the team works through reviewing the architecture, a constraint is clearly identified, and the team decides to introduce a new module between sensor management

and vehicle control as identified in Figure 6. This shifts responsibility for identifying an obstacle event to this new module: obstacle detection. During iterations, this component starts out as a programmable development board as a proof of concept, evolves into a field-programmable gate array (FPGA), and is finally integrated into the vehicle as an application-specific integrated circuit (ASIC) to vastly improve performance and power efficiency. The modularity and adaptability of the system architecture enabled the ease of additional sensors to be integrated without impacting the vehicle-control module.

During system testing, we have the ability to test the streamlets in a modular fashion. This means we can test events within the obstacle detection such as processing camera data and seeing results quickly without impacting the entire system. Larger solution tests are conducted on a regular quarterly cadence, or more frequently if needed, to test the end-to-end flow of the capability across the entire streamlet or value stream depending on the impact of the enhancement. Because teams are able to have access to a digital-twin system, and can build and test in a modular fashion, fewer issues are found during large solution testing.

The modularity of the system also allows for the teams to work independently throughout the iteration with limited impact to other teams. For example, the evolution from the development board to ASIC did not impact the iteration plans of the team working on innovative ways to handle obstacle events with improved safety features.

Iterate and Reduce Batch Size

As discussed earlier, each epic is broken down into multiple levels: from epic, feature, user story, to task. The break down of work allows Alset Transport teams to reduce their batch size and iterate through capability development, which enables flow through the system while offering fast feedback and continuous learning opportunities. During each iteration, teams implement user stories, which are small pieces of functionality or system enablers that can be built, tested, and proven to show results. As the team seeks to learn more about the system they are building, they perform regular user engagement.

The Alset Transport teams were building and testing the stories in two-week iteration cycles. Over time they recognized that their user stories were too large and taking

the entire iteration to complete. This created a large bottleneck with all the testing being pushed to the end of the iteration, and too often, the testing was slipping into the next iteration. These large batches resulted in increased time and effort as they struggled to integrate the new functionality. The team noticed an increase of defects and longer delays as integration had become more complex. A team member with a Lean development background suggested they try limiting the work in process (WIP) and create smaller batch sizes. First they focused on writing smaller stories that could result in stories being accepted throughout the iteration versus all stories being tested and accepted at the end of the iteration. This also meant that instead of starting all stories on day one of the iteration, they limited how many they worked at one time. The team would swarm around the minimal set of stories and get them completed as quickly as possible.

Short iterations provide an opportunity for teams to plan the earliest integration points possible with regularly validated requirements and interfaces. Early iterations of hardware development focus on the creation of an emulator, while later iterations focus on the creation of the first prototype with refinements and enhancements guided by the results of each completed iteration. Working in small batch sizes provides regular visibility of progress and makes course changes less impactful with a quicker recovery time than past experiences working in larger development cycles. Additionally, working in short iterations promotes the concepts of building for flow, providing feedback, and a continuous learning cycle.

Epic 1: Software-Only Updates to Existing Fleet Vehicles

During the quarterly planning event, the hardware- and software-engineering teams analyzed the first epic focused on the software-only system updates for the existing fleet vehicles. They selected a sensor system with an available test environment, such as a simulated environment, and small, code-based sensors, such as a forward and backup camera.

Epic 2: New Camera

The hardware team focusing on Epic 2 participated in the planning of their sprints and identified the constraints, interface requirements, and module behavior of each unit.

The team developed new functionality for radar sensors and was able to use previously generated tests on a radar simulator to check the new code deployment. They focused on frequent code integration with the new camera and radar system and resolved any integration errors. Upon successful demonstrations and test results in the simulated environment, the team moved to testing the firmware and software updates on a couple of production cars. This provided end-to-end functional and safety testing to ensure a positive user experience. Keeping to their commitment of working in small batch sizes and limiting WIP allowed them to test more frequently in the production environment with faster feedback as they validated the new cameras.

Establish Cadence and Synchronization

Cadence and synchronization are critical for the planning and development of the solution. Cadence provides predictable time boxes and rhythmic patterns for planning. Synchronization offers the team an opportunity to align their efforts and get regular feedback on how the integrated system is working.

For Alset Transport, the hardware and software teams agreed to adopt the same quarterly program increment and iteration cadences. Originally, the teams discussed different iteration lengths, whether they should be two-, three-, or four-week cycles, and what would work best for them. During the discussion, the teams balanced their concerns regarding being able to break down work at a granular-enough level to fit within a two-week iteration with the concern of having iterations that were too long, resulting in delayed feedback and reprioritization mid-iteration.

After negotiating, the teams agreed to go with a two-week iteration length and to adjust it in the future if they find it isn't working for them. The hardware team still has concerns but is willing to give it a try based on their learning and recognition that a two-week iteration may not mean completed capability, but rather a point for them to receive feedback on their work products.

As part of the synchronization points, the teams agreed that at the end of each iteration they would demonstrate their progress and receive feedback. Both the software and hardware teams collaborated as Agile units in order to create their quarterly and iteration plans and to align their demonstrations.

Epic 1: Software-Only Updates to Existing Fleet Vehicles

Their quarterly program of increment planning enabled them to identify and plan the specific software enhancements needed to implement the first epic and areas where they needed to collaborate with the hardware-focused team supporting Epic 2.

Epic 2: New Camera

The hardware team focused on solution-specific camera updates, while engaging with the software team to address risks and dependencies. At the same time, the hardware team identifiesd and planned out the new camera features, models, designs, and long lead items.

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

As illustrated in Figure 7, cadence and synchronization together provide teams with the tools they need to help manage the complexity and variability of large-scale solution development.

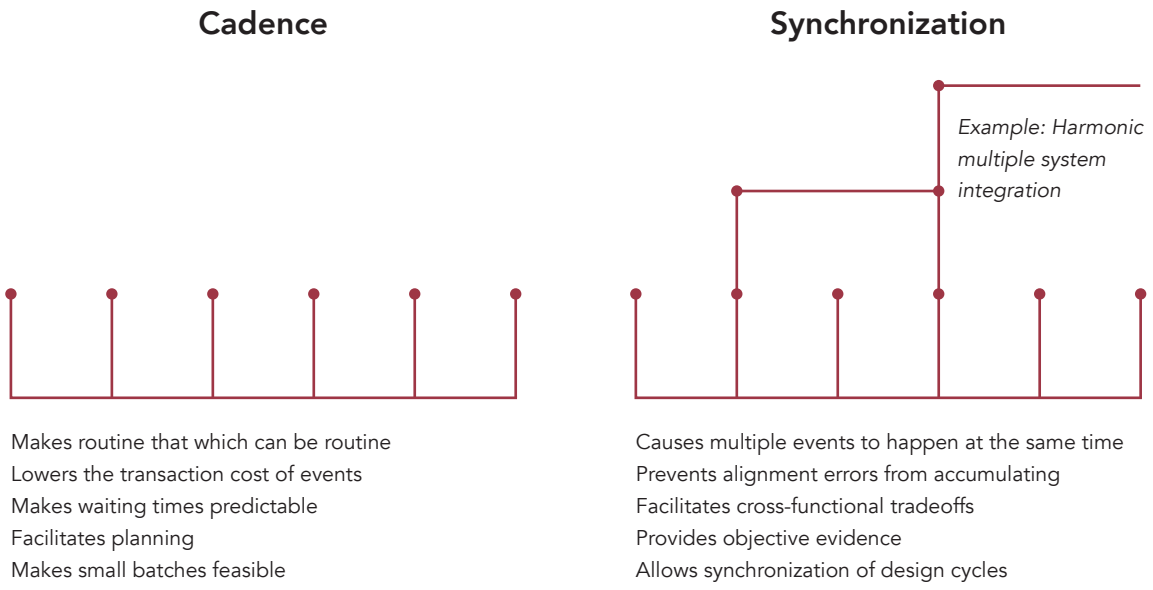

Figure 7: Cadence and Synchronization

Employ "Continu-ish" Integration

The principle "employ 'continu-ish' integration" describes how the goal of truly continuous integration faces economic and practical challenges when dealing with large-scale, cyber-physical systems. It isn't practical to integrate full end-to-end solutions as frequently as we can with pure software systems. The value streamlets discussed in section one may evolve at different rates. Rather, we use the economic and physical constraints to create a plan that integrates as much as possible as frequently as possible—with the larger goal being overall risk reduction for the program. Figure 8 illustrates this approach.

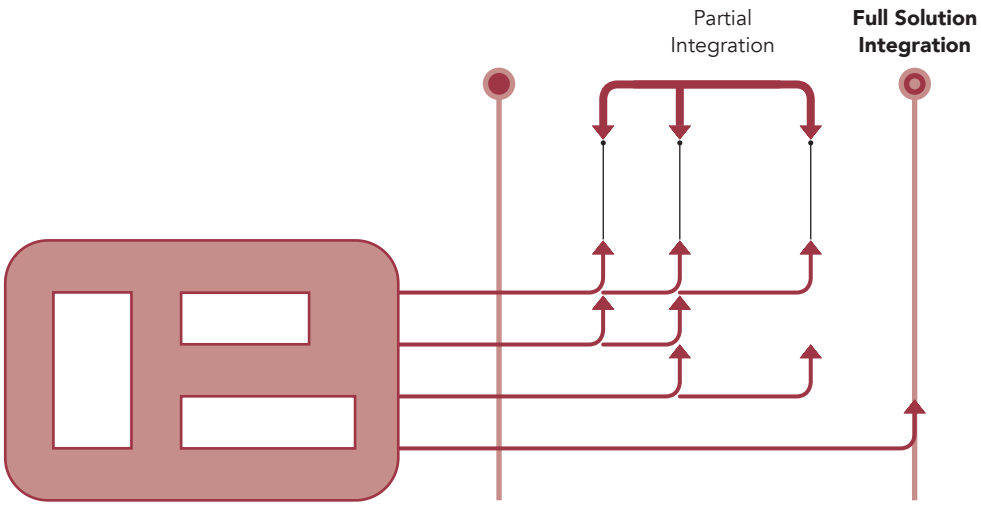

Figure 8: "Continu-ish" Integration

Each streamlet will mature their part of the solution independently by evolving the software and hardware. Streamlets evolve hardware using development kits, breadboards, brassboards, systems on chip (SoCs), FPGAs, hardware revs, and other prototypical solutions to frequently integrate their localized changes with the system's other streamlets. Balancing the costs and effort to create new hardware with the value of fast feedback and reduced risk helps determine the optimal frequency for revisions. Looking at ways to lower manufacturing time and costs, many find that frequency to be within weeks or a few months.

Per our vehicle exemplar, there are two different cases, each with different integration patterns.

Epic 1: Software-Only Updates to Existing Fleet Vehicles

The first epic is the most straightforward. The application of cadence and synchronization defines that work will naturally occur in short, predefined timeboxes—typically, sprints of two weeks aggregated into periodic program increments. The logical architecture illustrates the various software and firmware elements that need to be updated. In support of incremental development, the program has established a digital twin and test environment that makes developer-level testing cheaper and faster. Most new developments and code-level integrations can happen routinely, daily, or even hourly on that environment. Routine DevOps practices of source code control, automated builds, and automated build verification tests apply well in this case.

In terms of deployment, however, the situation becomes more interesting as eventually the new algorithms have to be field-tested in a real vehicle. For software changes, teams could apply a continuous delivery, the DevOps deployment strategy:

- Release to the controlled test environment and run validation protocols on vehicles on the test track.
- Deploy a feature toggle into a canary-release environment for vehicles in production, which allows for remote and server selective toggling of the capability, as well as limits exposure to the selected target vehicles. Validate the solution in the limited-scope but true production environment.
- Leaving the feature toggle in place, deploy the update to the entire fleet. Selectively control and enable the feature toggles until the entire fleet is enabled.
- Finally, move the change into the production process.

Epic 2: New Camera

The addition of the new camera presents a more significant challenge to implementing DevOps and assuring continuous flow. Camera specs have been established and provided by the supplier. The question becomes, how can the development teams

continuously develop and validate the mechanical and software aspects of a new camera—one with higher resolution, an enhanced depth of field, and a faster focus time—which does not yet exist? To address this, the teams employ the following strategy:

- Build a camera-emulator software device, or camera test double, which feeds simulated camera data in via the predetermined API and protocols. This device is deployed into the test bed and on a test vehicle to allow new algorithm development for enhanced obstacle detection without the physical device. The team also supports field-testing mocks on an actual test vehicle.
- In the meantime, the mechanical design has been tested with mechanical mock-ups, which are consistent with the intended physical properties of the camera.
- Beta devices replace the camera double as the supplier makes them available. Teams use the new camera revisions for continuous testing and evaluation, as well as to provide electrical, performance, and mechanical feedback to the supplier.
- During this time, hardware teams from the chassis group collaborate with the supplier on mounting positions and angles to optimize manufacturing and to retrofit existing vehicles. Feedback is also provided to the supplier as new camera revisions become available.
- Eventually, the supplier provides the final form factor, pilot-camera hardware devices, which are integrated into the test environment for continuous development and validation.
- When first production lots arrive, the team replaces all newly enhanced devices and continues testing.
- The team finalizes mechanical production specs and provides them for manufacturing.

Be Test-Driven

Test-driven simply means beginning with the end in mind. With test-driven development (TDD), teams write the tests for a change before they implement it, in both software and hardware. Over time, TDD creates a large set of automated and manual regression tests that help ensure quality software and hardware.

Modular architectures enable TDD by making testing functional behavior for individual system components simpler and faster. Components can be independently built and tested, and, in the spirit of DevOps, released with more confidence that the change does not break another part of the system. The above-mentioned integration strategies, coupled with the development cadence and synchronization, provide opportunities for higher-level tests to validate full system functionality.

When it comes to the firmware aspects of our vehicle, frameworks exist for applying TDD and mocks, or test doubles, to hardware description language in electronic design (VHDL) and Verilog for unit testing. Additionally, TDD can be applied to register transfer language (RTL) defining the digital portions of design blocks in ASIC circuit design. Engineers need to adopt the test-first mindset in order to apply TDD in this space.

Test-driven development also applies to hardware 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. In fact, some of these tools provide more mature and feature-rich testing capabilities than other software tools. The build of mocks and test doubles isolates hardware changes for early evaluation and testing.

Epic 1: Software-Only Updates to Existing Fleet Vehicles

Test-driven development in Epic 1 is relatively straightforward. These practices and tools have long existed for software. For every software modification, the developer adds or modifies one or more tests and then updates the software to pass them. This process repeats, with all tests being rerun continuously, until the change is completed and delivered.

The test-first challenge for cyber-physical systems resides in the fact that much of the software has been embedded. Existing test-driven infrastructure expects to launch the unit under these tests, drive its execution, and gather data to report results. Many embedded systems today target both a host and specific environment. Existing TDD infrastructure can be used for host-based testing but needs to be modified for target-based testing. Testing on the target requires additional infrastructure to launch, execute, and transfer test results.

Epic 2: New Camera

When it comes to hardware changes, the vehicle's modular architecture facilitates TDD. Test doubles allow development on parts of the system to proceed independently. In Figure 9, a camera test double simulates the updated message format and frequency to allow sensor management to evolve. As the camera design develops, any changes to the interface or its behavior are also made to the test double. Sensor management is done entirely in software, so common TDD practices can apply.

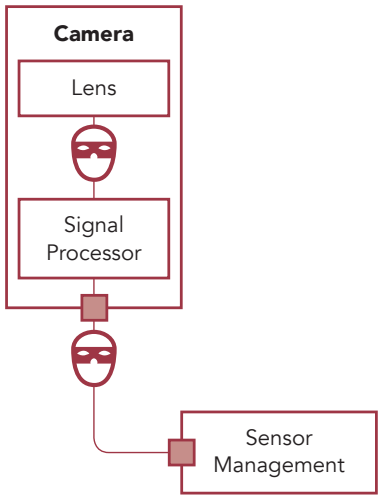

Figure 9: Test Doubles Facilitate Test-Driven Development for Hardware

Within the camera subsystem, a lens test double simulates optical signals that the processor can convert into formatted messages. In the electrical CAD environment, the designer starts by creating a small, new—or perhaps, modifying an existing—signal simulation for the lens. The designer sees it fail then modifies the design to process this new signal. The designer can create or modify another signal from the new lens and update the design. This small test, or change process, repeats until the signal-processor design is completed.

Conclusion

Alset Transport is a fictional company but demonstrates a real and recognizable opportunity for companies who build cyber-physical solutions. The application of DevOps principles permits Alset Transport's teams to work with more agility in smaller batch sizes. An implementation of digital twins permits automated testing and continuous integration. As the products move to physical systems, this work enables unique continu-ish integration, shortening cycles between software deployment to hardware. These combined capabilities allow Alset Transport's teams to develop against the multiple epics simultaneously and deliver value in shorter, iterative time spans.

Ultimately, Alset Transport's application of Industrial DevOps will deliver increased value to customers more quickly and, as a result, yield greater economic rewards—an outcome that should attract all manufacturers of cyber-physical systems. The companies that solve this problem early will improve transparency, reduce cycle time, increase value for money, and innovate faster. More simply, they will build better systems quicker, and they will become the ultimate economic and value-delivery winners in the marketplace.

References

"Guide to Car Safety Features." Consumer Reports website. Last modified June 2016. https://www.consumerreports.org/cro/2012/04/guide-to-safety-features/index.htm

Kim, Gene, Jez Humble, Patrick Debois, and John Willis. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. Portland, OR: IT Revolution, 2016.

Oosterwal, Dantar P. *The Lean Machine: How Harley-Davidson Drove Top-line Growth and Profitability with Revolutionary Lean Product Development*. New York: AMACOM, 2010.

Plumb, Steve. "Steering Towards Autonomy." *Automotive Design & Production*, December 2, 2016. https://www.adandp.media/columns/steering-toward-autonomy.

Reinertsen, Donald G. *Principles of Product Development Flow: Second Generation Lean Product Development*. Redondo Beach, CA: Celeritas Publishing, 2009.

"SAFe Principles." Scaled Agile Framework website, accessed April 29, 2018. https://www.scaledagileframework.com/safe-lean-agile-principles/.

Udaniz, Alex Udanis. "The Technology Behind Active Safety Systems in Cars." *All About Cars*, 2016. https://www.allaboutcircuits.com/news/the-technology-behind-active safety-systems-in-cars/

A Special Thank You to Our Sponsor

Our mission for the Forum is to bring together technology leaders across many industries and facilitate a dialogue that solves problems and overcomes obstacles in the DevOps movement. For three days at this private event, we gather 50 of the best thinkers and doers in the DevOps space to tackle the community's toughest challenges. We ask these thought leaders to collaborate and generate a piece of guidance with their best solutions to the challenges.

We would like to thank all of our attendees and our friends at XebiaLabs for helping to make this year's Forum a huge success.