17th International Conference on Space Operations, Dubai, United Arab Emirates, 6 - 10 March 2023.

SpaceOps-2023, ID # 291

SatDevOps: A Novel Automated Satellite Operations Methodology

Brunston Poon[a]*, Lucas Brémond[b], Caleb MacLachlan[c], Leon Stepan[d]

[a] *Loft Orbital Solutions, 321 11th Street, San Francisco, California 94103, USA,* brunston@loftorbital.com
[b] *Loft Orbital Solutions, 321 11th Street, San Francisco, California 94103, USA,* lucas@loftorbital.com
[c] *Loft Orbital Solutions, 321 11th Street, San Francisco, California 94103, USA,* caleb@loftorbital.com
[d] *Loft Orbital Solutions, 321 11th Street, San Francisco, California 94103, USA,* leon@loftorbital.com

* Corresponding Author

Abstract

We present SatDevOps, a satellite operations methodology which incorporates human knowledge into automated satellite operation systems. SatDevOps delivers consistent performance across a heterogeneous space infrastructure at a lower operating cost and faster development cycle than alternate traditional satellite operating philosophies by leveraging the respective strengths of human decision making and software automation.

There are six core competencies which comprise SatDevOps: build processes, then automate them; abstracted automation which can be reused across heterogeneous customer missions; infrastructure that enables an on-call paradigm rather than an on-shift paradigm; developing new capabilities under a test-and-develop-as-you- fly model; building tools that are usable by both software and human operators, and; a training program which encourages the developer-operator mindset by teaching software developers to be satellite operators.

Building manual processes with the same systems used for eventual automation increases development velocity. Abstracted automation allows an organization to maintain a single set of infrastructure for the concept of operations which are similar across otherwise heterogeneous missions. The on-call paradigm reduces the operator resource constraint. Maintaining the same type of infrastructure for test, development, and flight means that we can harmonize our satellite operations with our development cycle, avoiding friction in transitioning between environments. Building multifunctional tooling (for command and control, telemetry, alerting, data delivery, and customer interaction) that is designed from inception to be interacted with by both people and software allows us to maintain synchronization between automated concepts of operation and manual intervention. The training programs encourage every developer to learn the fundamentals of satellite operations in order to understand how to develop software for operational needs.

This paper will present the implementation of the SatDevOps satellite operation model using these core competencies and how we developed and use the SatDevOps operations model at Loft Orbital. Its benefits will be contrasted with traditional satellite operations models.

Keywords: satellite operation architecture constellation autonomy infrastructure

Nomenclature

Conops: Concept of Operations
DevOps: Portmanteau of *development* and *operations* in the context of software development.
Satcon: Satellite Controller
SatDevOps: Portmanteau of *satellite*, *development*, and *operations*, in the context of satellite operations.

Acronyms/Abbreviations

AIT: Assembly, Integration, and Test
API: Application Programming Interface
COE: Customer Operations Engineering
GSN: Ground Station Network
GNC: Guidance, Navigation, and Control
GUI: Graphical User Interface
LEO: Low Earth Orbit
LEOP: Launch and Early Operations Period
MCS: Mission Control System
MVP: Minimum Viable Product
NRE: Non-Recurring Engineering
OSI: Open Systems Interconnect
SME: Subject Matter Expert
SRE: Site Reliability Engineering
TLE: Two-Line Element
TM/TC: Telemetry/Telecommanding

1. Introduction

The commodification of space assets—satellites, payloads, ground station networks, etc.—has created a new frontier which is increasingly dominated by microsatellites. These satellite buses, more capable than cubesats but less expensive than their larger counterparts, offer a great balance between capability and cost. For Loft Orbital to be a space infrastructure service provider, where a customer flying a physical or software-defined payload should not be concerned with the underlying "bare metal" hardware, we must create abstraction layers which separate the interface from the implementation. Of course, this must be done with the software and hardware interfaces, and that can be seen in our Hub product as well as Cockpit, our mission control system [1][2].

To operate this infrastructure, we need to create an operational structure that evolved beyond the traditional satellite operations model, which scales the satellite operations team proportionally to the number of satellites in the constellation [3]. Drawing inspiration from the software deployment model of DevOps and site reliability engineering (SRE), pioneered by Google [4], we adapted the DevOps and SRE paradigms to satellite operations, naming this new satellite operations methodology SatDevOps [5].

The DevOps model comprises the tools, practices, and culture around operations and development [6]. It seeks to break down the cultural separation within an organization between the engineers responsible for building a system and those responsible for operating it. Sometimes it combines what were traditionally separate roles: an operations engineering team, responsible for the maintenance and deployment of the software written by a software engineering team. Sharing the responsibility for the overall success of a software deployment across all engineering efforts aligns the incentives.

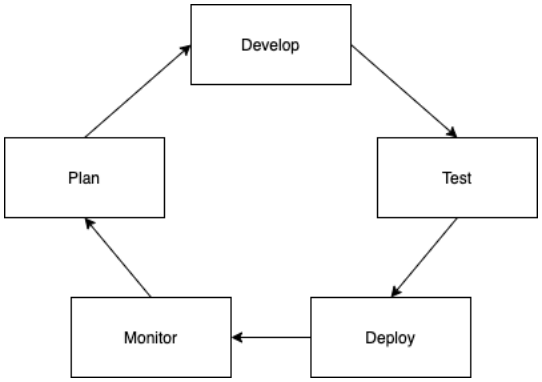

Fig. 1. DevOps development and feedback loop

Site reliability engineering treats operations as a software problem, believing that software approaches should be applied in processes beyond software product development [4]. SRE practitioners use service level objectives and measurement to make intelligent decisions on how best to manage availability (100% uptime on every service is unattainable and not considered an appropriate target). Site reliability engineers "work to minimize toil", understanding that performing operational tasks manually means less time doing product development work to make services more reliable [4]. At the same time, these manual tasks can provide insight into how best to approach the automation of those tasks and informs the product development roadmap, bringing reality to a process which is often overly theoretical. SRE aims to reduce the cost of failure—rather than focusing on never having issues on deployed systems, there is an emphasis on improving the speed of product development output, which allows fixes to occur earlier in the product lifecycle.

The final two principles that are core to SRE which transfer particularly well to satellite operations are a shared sense of ownership, and using the same tools regardless of function. By using the product on an operational basis, developers become keenly aware of the shortcomings of new features and the effects of architectural decisions. In using the same tools regardless of job function, it enables engineers from different teams to "speak the same language" and reduce "translation time" between the different phases of building a new product capability.

These approaches to developing and operating software systems share fundamental structures: they combine the product development and operations lifecycles with a collaborative management philosophy, through a feedback loop incorporating measurement and observation of the deployed production system, and with the aim of increasing efficiency and throughput in both lifecycles.

The major principles of SatDevOps are based on our synthesis of the keystones of DevOps and site reliability engineering with satellite operations. SatDevOps is like its inspirations in that it comprises both methodology and a culture. It requires a system that can support the implementation of the methodology, and it requires an engineering team that embodies the culture.

The paper is divided into two primary sections: an explanation of the methodology and philosophy of SatDevOps, and a case study of its application to two Loft Orbital satellites and the development of Cockpit, Loft Orbital's mission control and enablement software. It will conclude with a discussion of the advantages and pre-requisites for successful adoption of this paradigm.

2. The SatDevOps Satellite Operations Methodology

To operate a heterogeneous constellation of microsatellites in low-earth orbit (LEO), Loft Orbital designed a mission control system (MCS), Cockpit, and elected to operate its constellation with trained satellite operators and flight directors drawn from the set of engineers involved in design, implementation, and integration of the satellites and the MCS.

In order to accomplish this, SatDevOps requires following six core competencies: build processes, then automate them; abstracted automation which can be reused across heterogeneous customer missions; infrastructure that enables an on-call paradigm rather than an on-shift paradigm; developing new capabilities under a test-and-develop-as-you-fly model; building tools that are usable by both software and human operators, and; a training program which encourages the developer-operator mindset by teaching software developers to be satellite operators.

SatDevOps and satellite operations more broadly cannot be fully separated from the systems which are used to run those operations, the engineers who staff those operations, or the culture cultivated by the operating organization.

2.1 A Note on Systems and Culture

SatDevOps requires that operations systems have a certain agility, flexibility, extensibility, and scope. The systems used must be designed for rapid product development and iteration.

Loft Orbital uses its in-house MCS, Cockpit, built around the concepts of flexibility and scalability. The system enables the successful execution of SatDevOps as an operations methodology.

Cockpit takes the place of what in traditional low-earth orbit (LEO) satellite operations might require multiple systems [3]. In a single system, it encapsulates telemetry/telecommand (TM/TC) capability (both manual and automated operations), payload requesting, operation scheduling, simulation, flight dynamics, data processing, data delivery, operations alerting, ground segment management, among other capabilities. It has two interfaces: an application programming interface (API), and a web-based graphical user interface (GUI), both heavily used in operations.

The design philosophy of Cockpit is based on satellite and payload agnosticism—namely, that its architecture is designed to fly multiple satellites from different vendors and control multiple payloads from distinct customers without requiring major non-recurring engineering (NRE). Because Loft Orbital is a space infrastructure service provider, granting customers a variety of payload control and tasking authority based on configurations ranging from dedicated missions to rideshare to constellation management, Cockpit serves both internal uses and external customers. By centralizing control, management, planning, and data delivery in a single system, it provides a cohesive experience which accelerates internal product development and customer payload operational success.

The underlying implication of the development of SatDevOps in parallel with that of Cockpit at Loft Orbital is that the success of SatDevOps is largely made possible by the architecture, development, and deployment of Cockpit within the organization. For example, the translation between manual and automated processes, a key tenet of SatDevOps, is trivial to implement in Cockpit. The system responsible for automatic satellite operations, Autopilot, can run a formerly manual process directly, with no code changes required.

For the SatDevOps methodology to be successful, the mission control software used to operate the satellite or constellation must meet certain requirements to be able to support this method of operation. Illustrations of the six core competencies are enabled and implemented by the system use Cockpit as the reference system, as both SatDevOps and Cockpit, used for all satellite operations at Loft Orbital, were developed together. The architecture of Cockpit encompasses more than what traditional mission control software is capable of. It combines the core mission control capability of TM/TC with client requesting, ground station management, mission planning and optimization, automated and manual satellite operations, alerting, and client data delivery all within one integrated system. We support multiple satellite bus and payload architectures by leveraging abstractions of satellite and payload interaction, communication protocols, ground station providers, and many other systems.

SatDevOps cannot exist solely as a methodology without an associated organizational culture. In broad strokes, that culture is one of collaboration, constant improvement, failure as a learning opportunity, and process-building towards eventual automation. SatDevOps does not work with “reluctant operators”—rather, it demands engineers who are passionate about combining satellite operations with their primary responsibilities. Even engineers who are not in the operations on-call rotation must always consider the operational impacts of their product development and seek to improve operations in all that they do.

2.2 Nominal Operations

On any given day, there is a satellite controller (satcon) and flight director on-call. They are responsible for handling any anomalies which are surfaced by the alerting capabilities of Cockpit, or any issue which is called to their attention by any member of the development team. Issues raised by clients are filtered through the customer operations engineering (COE) team. The vast majority of nominal operations are automated, including the requesting, planning, scheduling, and execution of customer activities. Occasionally a manual pass must be executed, which can be performed by either the flight director and satellite controller on-call or any other qualified flight director / satellite controller pair. Development and test of future space infrastructure capabilities is conducted based on direct feedback from SatDevOps-trained engineers, and new features are often written by those same engineers.

There is no dedicated operations team. Satellite controllers and flight directors are trained through a robust training program (section 2.8) but are engineers whose primary responsibilities are the development and implementation of Loft Orbital products and services. The six competencies which follow in the next sections explain how we have accomplished this.

2.3 Build processes, then automate them

It is commonplace to write flight operations procedures [3] which describe satellite operations. Not everything can (or should) be automated from the very start. The benefit of SatDevOps is that operators can do this with an eye towards eventual automation. We align the incentive of traditional operations with the incentives of software development; namely, we want to reduce manual operations to the minimum subset possible. At the same time, the process of building that process can be *more* iterative when done in a "traditional" manner. We avoid the paralysis of waiting until a software system has a full capability by using an iterative approach. If a process (for example, downloading a debug log from a customer payload, or performing maintenance on the onboard filesystem) can be accomplished manually, and the source of truth for that process is version tracked and stored in the same manner as code, then it becomes trivial (and therefore more efficient) to automate that process.

We have designed Cockpit, our mission control system, to use the same storage and definition mechanisms for both manual and automated procedures. This is one of many architectural decisions which integrate the SatDevOps methodology into our systems from the ground up.

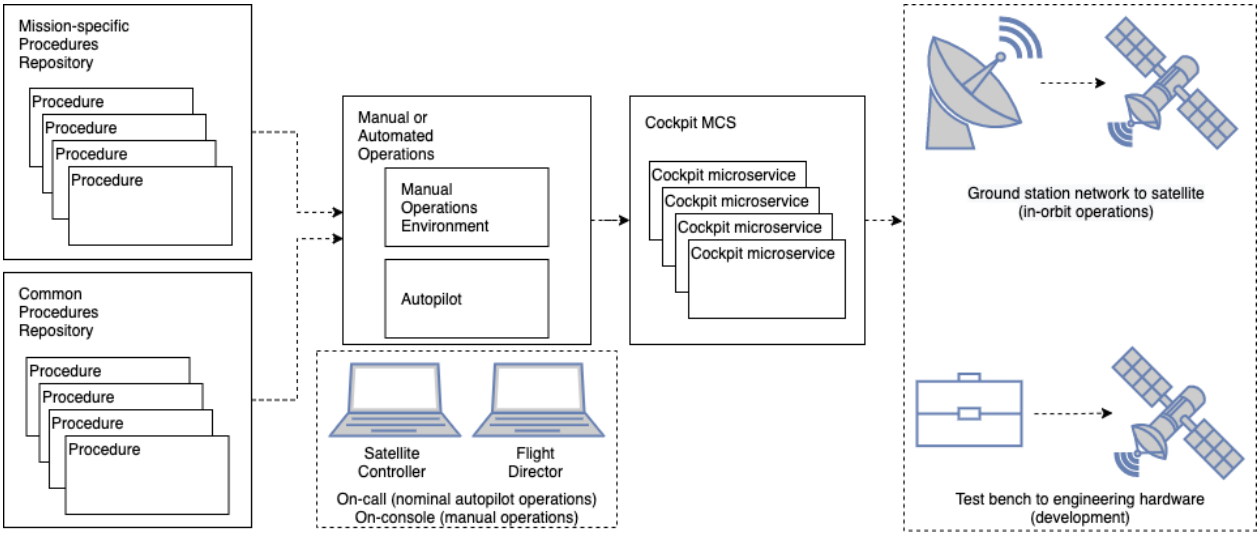

Fig. 2. Illustration of how Cockpit enables both automated and manual operations in-orbit and for development.

Culturally, the focus on building process means a reduction in single-points-of-failure, where only one individual has the information or context required to perform an action. This is not meant to make engineers disposable or replaceable, but rather to free them up from direct operational actions so that they can spend more time on core product development. By encouraging documentation and version-control, it is easier to maintain the source of truth between the development, test, integration, and operation phases of a new feature rollout or operational objective.

2.4 Abstracted automation

With the right set of abstractions, flying a heterogeneous constellation—whether in terms of satellite bus or of customer payloads—is no different than flying a homogenous one. The simplest analogy for this is that of a cassette tape player. The player itself makes minimal assumptions about the cassettes it accepts—there is a standard format for the audio which is encoded on the cassette, and a standard form factor, but beyond that, the player makes no further assumptions. Designing a mission control system around this type of standardization and set of limited assumptions creates an environment where new capability—usage of new satellite buses, payloads, ground station networks (GSN), or other hardware/software—can be incrementally added without constant redesigning or refactoring of the old system. Linking these abstractions as objects within the data models of a mission control system means that higher-level manipulation of these abstractions can occur regardless of their underlying implementation.

Implementing this abstracted automation in a mission control system on its own does not inherently create a more efficient means to operating a satellite. Yet, systems enable operations. Building the right framework improves flexibility both for the operator and in the developer—perfect for a model like SatDevOps where these two roles are shared by the same people. From an operator's perspective, automated abstraction allows an operator to maintain a single mental model shared between manual operations and automated operations. Reusing the same abstractions

across heterogeneous payloads allows for the bigger picture of a heterogeneous constellation to be built. It reduces the uniqueness aspect of different customer operations. From a developer's perspective, abstracted automation makes it easy to build new features that make use of these abstractions. The Autopilot of Cockpit is a great example of how abstracted automation in the *system* improves the *operator* experience.

In software, it is common to consider the highest-level interface—the application—as a composition of abstractions on top of abstractions (see, for example, the Open Systems Interconnect (OSI) networking model). The SatDevOps model of operations relies on the abstraction of automation described above—it demands that the system-level automation used in mission control software be standardized as much as possible, reusing the same architecture and concepts to accomplish different customer concept of operations (conops). Everything within the system composes on top of lower-level abstractions. More precisely, individual telecommands are abstracted into sets which accomplish certain tasks, like downloading backorbit telemetry. These task sets can be ensembled into more complex pass plans and payload activities. The underlying architecture, therefore, is of paramount importance. It is linked to every possible composition of abstraction.

In "Lessons Learned from Automating Heterogenous Spacecraft Systems" [7], we describe the way we have chosen to model these abstractions and architecture within Cockpit.

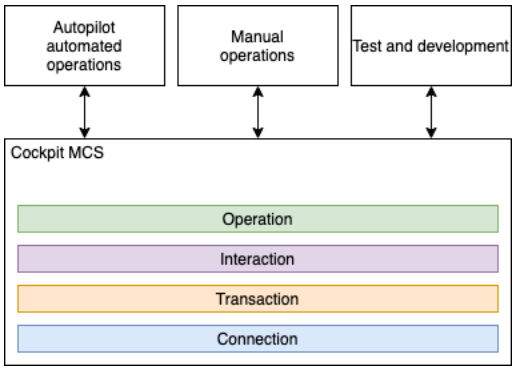

Fig. 3. Simplified abstraction layer architecture of the Cockpit mission control system.

The consequence of these abstractions on operations is monumental. It simplifies the process of managing nominal automated operations and occasional manual operations by wrapping the complexity of these different payload systems within a framework re-used across the infrastructure, allowing operators to maintain a single mental model when writing new automated procedures or diagnosing an in-orbit issue.

Similarly, from a system perspective automated operations and manual operations use the same version-controlled procedure code. The same operations which are performed on test benches or representative simulators on the ground must be directly translatable to operations in-orbit. Properly implemented, it is a key part of what enables the flexibility of operations. See section 2.6.

2.5 Infrastructure that enables an on-call paradigm

In merging satellite operations with DevOps and SRE principles, SatDevOps requires an infrastructure that is resilient and observable, and that supports multiple alerting systems to raise an alarm if off-nominal conditions are detected. Part of the influence of DevOps is that the software systems and infrastructure which are used for satellite operations are being developed with high velocity while maintaining strong levels of reliability, security, and quality. As a result of training developers in lieu of a dedicated operations team, the impact of every new feature, software version, and infrastructure deployment gives direct feedback, encouraging the constant elimination of pain points. Therefore, we stress the *on-call paradigm* in opposition to the *on-console* or *on-shift* paradigm (which implies constant monitoring and constant availability).

In this sense, infrastructure is an intentionally overloaded term. Software, process, and human infrastructure work together to enable the on-call paradigm.

2.5.1 Software infrastructure

Software services designed to be fault-tolerant or are capable of automatically restarting themselves if they crash and deploying these services using industry best-practices from Silicon Valley are a critical part of SatDevOps. The use of Kubernetes-deployed micro-services allows for incremental upgrades to different parts of the system without

downtime. Designing Cockpit as a cloud-based MCS from the beginning [7] allowed us to leverage the substantial amount of software available to facilitate these deployments. Monitoring, logging, observability, and availability are all core principles that must be used when designing and deploying the software infrastructure. This is a tangible influence of the DevOps heritage of this new satellite operations methodology.

2.5.2 Process infrastructure

Within SatDevOps is a process for triaging and responding to satellite anomalies. This is a direct implementation of the first principle of SatDevOps (*Build processes, then automate them*).

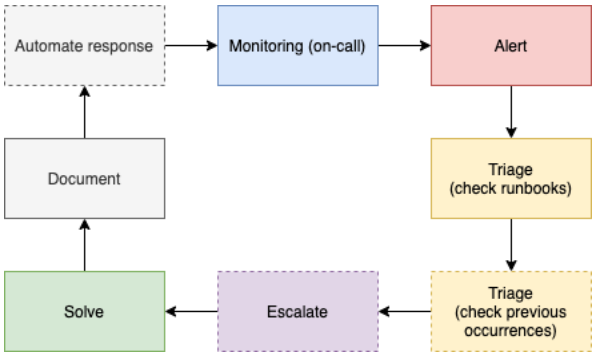

Fig. 4. Satellite anomaly triage resolution and response automation feedback loop.

During an on-call shift, if an error which the system is not designed to autonomously handle is found (based on alerting values established during the development of a particular program), the on-call satellite controller is paged. They begin by checking runbooks, documents detailing manual processes to respond to particular issues which we have not yet automated but understand how to properly address.

If no runbook is available, they check the error against a list of compiled error tickets which detail previous errors and any corrective actions (documented manual actions), if available, to rectify the issue. Finally, if a satellite controller does not find an established corrective process, they escalate to the on-call flight director, who has additional experience and training which they can use to identify the issue and resolve it. If a new issue is discovered, it is documented and any corrective action taken to fix it is also noted, building a manual process which can later be automated.

2.5.3 Human infrastructure

There are certain limitations that should be imposed on satellite controllers and flight directors who are on-call. Following Federal Aviation Administration rules for pilots, intoxication is forbidden for operators in the eight hours prior to, or during, their shift. Additionally, operators must remain reachable by phone or internet and be able to connect to the mission control system within a certain amount of time.

SatDevOps is not a “free lunch”—care must be taken to achieve balance in all aspects. Satellite operations, even performed using the SatDevOps methodology, requires a certain amount of bandwidth from the operating engineer. As a result, the other responsibilities of an on-call engineer must be scaled appropriately to consider the amount of time spent on operations. The time spent is also variable; it may be that an entire shift occurs without any anomalies, or the spacecraft may go into safe mode, requiring substantial hands-on operations to restore nominal operations. On-call satellite controllers and flight directors should be compensated for the shifts that they take, as it represents an additional responsibility that they bear on top of their product development roles (in exchange, the organization does not need to staff a separate operations team, reducing required headcount and therefore overall operational expenditure).

2.6 Test-as-you-fly

A major requirement of SatDevOps is a test-as-you-fly approach to both pre- and post-launch activities. This is the equivalent of using the integration testing paradigm in as many places as possible during the lifecycle of a mission. There are a few components to test-as-you-fly which are worth highlighting.

2.6.1 Use the same systems everywhere

SatDevOps works better as an operations methodology the more places the operation system is used. If payload conops development occurs using the same mission control system as operations, there is one fewer translation layer in the system. One must remember that getting development and operations in as tight of a feedback loop as possible

improves operational efficiency and throughput, which is the advantage that SatDevOps provides over other satellite operations methodologies.

At Loft Orbital, we use Cockpit throughout the test process, which allows us to build test procedures which eventually become operational procedures without requiring transfer to another system (see section 2.3 and figure 2). It also provides a more accurate simulation environment for training (see sections 2.6.3 and 2.8).

2.6.2 Automation

Setting up SatDevOps-based satellite operations for success prior to launch (or in-orbit, when developing new features) requires automated testing, easily reconfigurable test benches to support multiple heterogeneous payload architectures and satellite buses, a staging environment that mimics the production deployment, and engineers who understand the value in catching errors through automated testing before encountering them in-orbit.

To this end, Loft Orbital has developed an internal product called Cutest which allows developers to track the results of tests under given configurations, enabling granular test-driven-development when integration tests can span a wide range of interconnected software systems. Much of this integration testing includes deployments of Cockpit, using the same procedure code used to operate the system in-orbit, reducing the likelihood of software regressions. Other automated systems allow deploying a specific set of software versions to physical hardware test benches, facilitating on-target flight software testing that also interacts with the MCS directly, avoiding unnecessary translation between test and flight procedures.

2.6.3 Fly-as-you-train

Test-as-you-fly mostly refers to the system, ensuring that the actions it is performing match those of your production system in-orbit. To train engineers as competent operators, it is imperative that the training environment they use match the flight environment as closely as possible. This can be done with software deployments (on-target or in emulated environments) of satellite and payload flight computer software, high-fidelity mocking of key components which cannot be included in the simulation environment, and a training environment which encourages experimentation and mistakes, to train against operational stresses during real operations.

2.6.4 DevOps in SatDevOps

Though SatDevOps claims descendance in part from DevOps, it also uses DevOps principles itself in the development and test process. The use of continuous integration and deployment reduces the bugs which manifest in flight. No code is deployed which hasn't "gone green", namely, passed comprehensive test suites (from unit tests to integration tests).

For ground software, to include changes to the mission control software, this includes testing at the local deployment level (think developer workstations or test benches), at the staging environment level (mimicking the production deployment), and finally being released to the production environment.

For flight software, this might include a deployment to an emulated target (a Linux image, for example), then a deployment to the target (an engineering flight computer deployed to a physical test bench), and finally uploaded and flashed to the production target in-orbit.

2.6.5 Operations-as-a-company

SatDevOps as a practice and as a culture needs to be practiced company-wide, to include management support and support from engineers not directly involved in operations. As a cross-disciplinary effort, satellite operations necessitates close collaboration between flight software and ground software teams. Other subject matter experts, payload engineers, embedded software engineers, system engineers, and assembly, integration, and test (AIT) engineers must contribute to truly qualify a satellite through test-as-you-fly. This includes integrated day-in-the-life testing that incorporates as many flight-like software and hardware products as possible prior to launch. The SatDevOps mindset and culture can truly flourish if it is shared between all teams (which must also be supported by team management working to break down silos and encourage the free flow of ideas, information, and solutions).

2.7 Tools usable by both software and humans

Deriving from the SRE conception that "operations is a software problem" [4], the system that supports the SatDevOps methodology must be built of tools usable by both software and humans.

This concept ties directly to why using the same tooling across test, manual operations, automated operations (to include the servicing of customer requests), and internal maintenance has already been mentioned in this paper (see sections 2.3, 2.6). It is easier to keep one system in sync than to maintain multiple at once. If the interfaces designed

for human operators also treat their software APIs as first-class citizens, it intrinsically improves testability without requiring additional work.

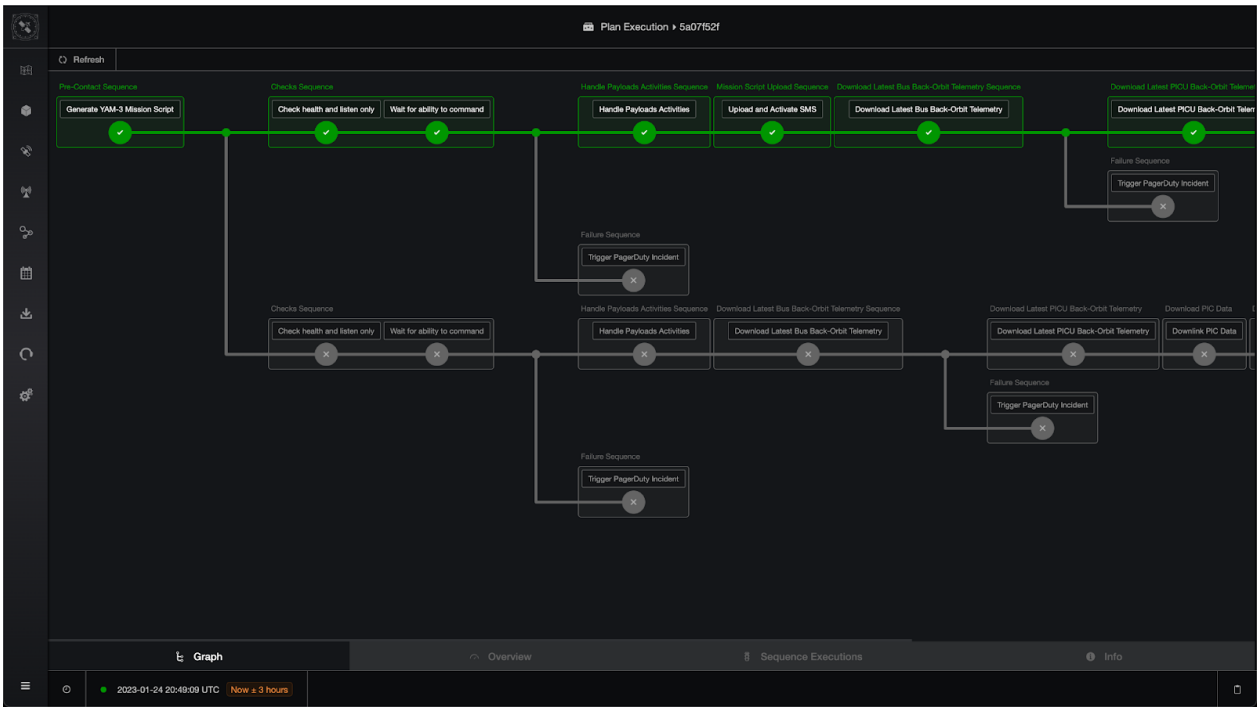

Fig. 5. Autopilot plan execution in Cockpit.

The mindset here is that of the complementary nature of human intelligence and software capability.

Because the operators in this methodology are also the developers of the system, the incentives to build the operations tooling in this manner are aligned. In an ideal system developed in parallel with SatDevOps, there is a unification of user experience, user interface, developer experience, APIs, and customer experience, because they are all are be linked and part of the same system. It reduces maintenance burden and eases the mental load of manual operations when the framework inherently exists for eventual automation. Reducing friction here improves efficiency, which leads to faster, better outcomes.

The same mindset can be applied to the customer-facing side of the system, if applicable. The purpose of SatDevOps is to improve execution of the mission. For Loft, those missions are performed on behalf of external customers.

Since Cockpit supports more than just the traditional TM/TC responsibility of a MCS, also being responsible for planning and scheduling [8], those services within Cockpit are also architected to be interactable through the APIs as well as visualized with the Cockpit Web GUI for operator awareness and customer insight.

2.8 Training program

Developing a strong training program is paramount to the success of SatDevOps. Since operators in this paradigm do not have the primary job responsibility to perform satellite operations, nor are they necessarily selected for their past operations experience, the training regime must be structured to provide a baseline of satellite operations, spacecraft systems, and astrodynamics knowledge so that a satellite controller or flight director can provide a guaranteed level of support regardless of background.

SatDevOps operates on a tiered model for training. The process is below:

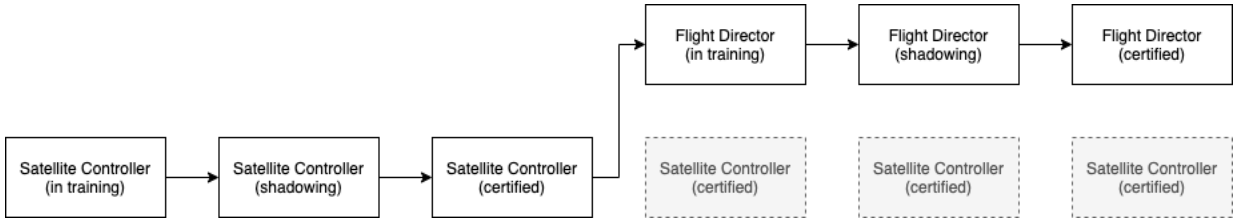

Fig. 6. SatDevOps training sequence.

2.8.1 Satellite controller training

The training material is developed collaboratively between subject matter experts (SME), flight directors, and satellite controllers (satcon) to share knowledge gained through interfacing with satellite bus manufacturers, suppliers, nominal operations, and commissioning.

Satellite controller training begins with familiarization with spacecraft systems, understanding what subsystems exist on a satellite and how they are used, followed by familiarization with ground software and flight software systems. This learning can be tailored somewhat to the background of the engineer, since the pool of in-training operators is comprised of engineers working on one part of the system or another.

After learning theoretical processes, a satellite simulation environment is used to develop familiarity with manual satellite operations. Trainees learn about the execution environment for manual procedures, as well as how to use Cockpit, the mission control system, to manage both nominal and off-nominal operations tracking and planning.

While final operational control rests with the flight director, the satellite controller is authorized to follow resolution guides for known errors that surface occasionally and do not pose a risk to the satellite. As a result, we train satellite controllers to the degree that they can perform these corrective actions without assistance.

The last phase of training before satellite controllers are certified within this methodology is the “shadow” phase, where they are placed on-call with the role of shadow satellite controller and will be the first person to receive alerts. Prior to escalation to the certified satellite controller, the shadow satcon practices dispositioning alerts themselves, but has the safety net of the certified on-call satellite controller (and, of course, the on-call flight director) if they are unable to find a resolution.

2.8.2 Flight director training

Flight directors are not expected to know the disposition of every anomaly encountered in-orbit, but rather know the process for gathering information on an anomaly, finding relevant information, consulting relevant subject matter experts, and ultimately choosing a path forward towards eventual resolution given the constraints. Consequently, the training for the flight directors is an additive process which incorporates the subject matter expertise of flight director trainees.

Since Cockpit (our mission control system) and its associated tooling is highly extensible, it is also important for our flight directors to be able to extend that tooling during anomaly investigations. By learning how to build new telemetry dashboards and run complex queries on spacecraft logs and downlinked data, it’s in this phase of training where flight directors in training build problem-solving and anomaly root-causing skills.

One of the direct benefits of the SatDevOps methodology is that flight directors have deep knowledge in different areas.

Fig. 7. Cockpit Grafana dashboard displaying certain spacecraft telemetry.

Part of the transition between the satcon role and the flight director role requires that a flight director trainee has spent enough time as a satcon to build a strong foundation of understanding YAM operations, in addition to the same "shadow" phase before becoming certified. As they work on more anomalies / activities, they build their skills to be effective flight directors by following correct processes, learning sound decision making, and building a mental catalog of how to respond to certain events.

3. Case Study: YAM-3, YAM-5, and Beyond

3.1 Product development with YAM-2 and YAM-3

In the software industry it is common to use the concept of a "minimum viable product" (MVP) as a milestone for launching an initial release [9]. This version of the product contains the minimum subset of features required for the product to perform the desired mission, with the understanding that the "wisdom of production" [4] and the ability to rapidly develop the product using DevOps principles will guide and enable the roadmap of future feature development.

Cockpit was developed using this MVP approach. Developing from the very beginning with a micro-constellation of heterogeneous satellites in mind—different satellite buses with different onboard payloads for a diverse set of customers operating under several varied concepts of operation—and a payload business model which favors building infrastructure over non-recurring engineering forced the design of abstraction layers within Cockpit to ensure that as much of the system as possible could be used by both satellites without modification. Every piece of the Cockpit architecture was designed in concert with the six principles of SatDevOps.

3.2 Product and satellite operations evolution from YAM-3 to YAM-5

In *The Site Reliability Engineering Workbook*, Google engineers espouse the "wisdom of production" [4]. The lessons we learned from automating countless workflows and processes developed during the payload commissioning and nominal customer operations for YAM-3 informed the development of future features and procedures for YAM-5.

The influence of production experience on product development is not to be understated. The immediate pain felt in satellite operations, by the developers, allowed ruthless prioritization of the key elements of the system needing improvement.

Throughout the commissioning period of YAM-3 and its operational period post-commissioning prior to the launch of YAM-5, the SatDevOps methodology continued being refined. We continually refined and improved the customer experience by making their payload operations requests easier to manage and delivered more reliably, while improving the alerting and response infrastructure to require less manual work by the satcon or flight director to return to nominal operations. Additional tooling was built to improve the ability to test operations procedures exactly on test benches

like they would be operated in-orbit (see section 2.6.1). The training program (section 2.8) was developed for the next cohort of satcons and flight directors—the original group were drawn from engineers with prior satellite operations experience or with extensive knowledge of the satellite bus and payloads developed during the integration and test phases of the satellite development.

As the developers also responsible for developing the mission control software, flight software, and other products, the immediate feedback from being involved in satellite operations contributed to a rapid improvement and focus on improving product reliability, error handling, observability, and automation.

In preparation for the launch of YAM-5, we took the lessons learned from the commissioning of YAM-3 and prepared extensive tooling for both nominal and off-nominal scenarios, relying on the suite of automation and software features that had been developed in the year and a half between launches.

3.3 Operational testing for YAM-5

With a mature and constantly improving product, operational testing for YAM-5 was able to be completed using a representative version of the Cockpit production deployment eventually used to command YAM-5 in-orbit. This meant that LEOP procedures testing and training could occur by test-as-you-fly (section 2.6) and allow us to build processes, automate them (section 2.1), and prepare for off-nominal recovery operations in a simulated environment.

The test campaign for YAM-5 was operated on a test-as-you-fly basis. By operating the integrated spacecraft using Cockpit we were able to validate the whole system from end-to-end, including operation of the satellite from the production deployment of the MCS. This level of integration testing enabled a high degree of confidence and risk buy-down for the program. Running functional test campaigns through Cockpit also built efficiencies. Test procedures used to validate payloads on the test bench or integrated on the spacecraft could be used to validate the same functionality in-orbit with a minimal amount of change.

3.4 Operational training for YAM-5

Crews were trained for LEOP by flight directors and engineers with prior experience with LEOP on YAM-3 and other satellite missions. They were provided with both nominal and off-nominal commissioning scenarios under realistic conditions to simulate the launch.

Operational training for YAM-5 used the simulation environments and lessons learned from YAM-3 LEOP and commissioning. Documentation was improved for response to off-nominal operations as various scenarios were practiced. A "red team" responsible for orchestrating the off-nominal scenarios consisted of flight directors with extensive experience operating YAM-3 training the flight directors, satcons, and other ground support personnel who would be on-console during LEOP. In the spirit of test-as-you-fly, we made the simulation environment as representative as possible.

3.5 LEOP and commissioning

During LEOP, SatDevOps operates similar to traditional satellite operations, with flight director and satcon roles being augmented by a guidance, navigation & control (GNC) engineer, a ground system engineer, a ground segment liaison engineer, and the satellite commissioning lead. This is the only time where shifts are on-console, to provide the maximum amount of support. As SatDevOps and the systems supporting it matures, fewer and fewer aspects of LEOP will require manual processes at all, improving the cadence and repeatability of the commissioning process.

What distinguished the LEOP and commissioning phases of both YAM-3 and YAM-5 from other satellite operations methodologies was the rapid development and expansion of capability. Once Cockpit was flying our satellites in-orbit, we were able to leverage a fast cycle of feedback, building new features, using them, and iterating to create a cohesive suite of tools which enabled better commissioning.

For example, in the commissioning period of YAM-5, we had tasked our guidance, navigation, and control (GNC) engineers to update two-line element (TLE) orbital parameters based on telemetry from the satellite. We were able to create an automated process for generating new TLEs from the received telemetry which could send them to our ground station provider on an automatic basis. This freed up our GNC engineers to perform other tasks.

This illustrates the power of the feedback cycle: within two days of understanding a pain point—having to manually update TLEs during LEOP—an engineer was able to develop an automated solution that used the same tooling which was originally being used on a manual basis, test it, and deploy it to our production system. This velocity was only made possible by the architecture of Cockpit, and our use of SatDevOps.

3.6 YAM-6, YAM-7 Longbow, and beyond

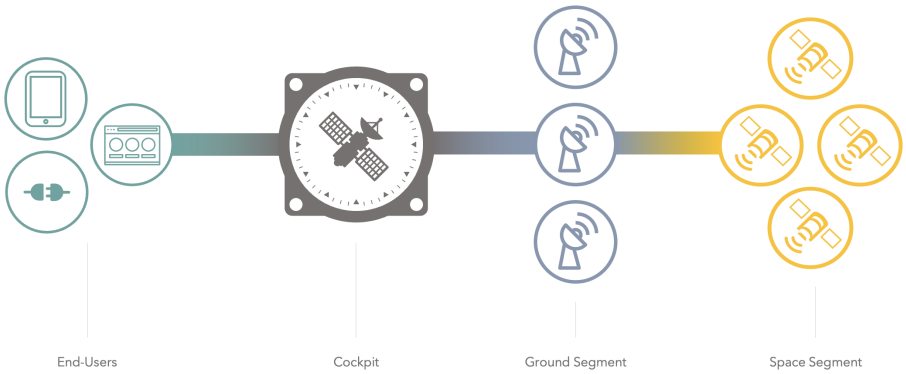

Fig. 8. Simplified overview of Loft Orbital space infrastructure.

SatDevOps will continue to iterate based on what we've learned from operating YAM-2, YAM-3, and YAM-5, and our continued dedication to creating space infrastructure which is payload, mission, and bus agnostic. The fundamental structures we've created (and presented here) provides us a framework with which to do so.

From a system perspective, Cockpit supports 4 satellite bus types already, including Loft Orbital's Longbow platform, derived from Airbus' Oneweb satellites, along with buses from LeoStella, Blue Canyon Technologies, and a cubesat provider, with more currently in development. The abstraction of the payload interfaces in Cockpit, combined with the Hub product developed by Loft Orbital for abstracting the physical and software interface of the satellite bus away from the payload onboard the satellite, allow Cockpit to provide an agnostic space infrastructure on top of which customers can operate their payloads. Because of Cockpit's extensibility and the SatDevOps space infrastructure-oriented approach to satellite operations, we are already prepared to support YAM-4, YAM-6, and YAM-7, along with our ongoing development of YAM-8 to YAM-19. As we grow our constellation, the planning and scheduling system will continue to evolve (see "Automated Planning and Scheduling System for a Heterogeneous Spacecraft Constellation" [8]).

4. Results

4.1 Limitations

SatDevOps is a philosophy tailored towards certain types of satellite operations and constellations. The principles were developed to fly a heterogeneous fleet of microsatellites.

SatDevOps is not, for example, meant for crewed spaceflight operations. Its principles are tailored toward the concept of flying a constellation of heterogeneous microsatellites (though it could be easily applied to a homogenous constellation) where the time-to-orbit and speed of execution is prioritized. Because it is dependent on organizational culture, organizational structure, and the MCS and other supporting software systems, it may be difficult to transition an existing organization to this methodology. We were fortunate in being able to instill SatDevOps as the satellite operations methodology of Loft Orbital from the beginning. The SatDevOps principles are also meant to evolve over time. As complexity builds, we will seek to simplify.

With YAM-3 and YAM-5 representing the operational portion of Loft Orbital's fleet of in-orbit satellites, we benefit from sharing a satellite bus architecture. As we launch our new Longbow-based satellites and ones built on other architectures, we will have to iterate on the best way to share subject matter expertise with the on-call rotation. We are experimenting with whether it makes more sense to train flight directors and satcons on single satellite platforms or on all supported buses. As we expand the fleet and constellations we support in-orbit, it will require careful management and sustained product development on further automation to ensure that we can continue to scale our satellite operations without impacting the quality of service for our customers.

5. Conclusion

As new business models and payload concepts of operations are made possible by offering a space infrastructure that is agnostic to payload or mission, the satellite operations methods must evolve alongside them. Cockpit's unique approach to mission control—enabling concurrent operation of multiple payloads across different satellite buses and ground station providers through manual/automated operations or customer requests—is expanding the scope of MCS from mission control to mission enablement [1][8][7][9].

For organizations seeking to have success as a space infrastructure service provider, they should consider whether their satellite operations might benefit from an approach like the one we have presented here.

In creating a satellite operations model which emphasizes agility and product development, SatDevOps is enabling Loft Orbital operations to run at the speed of software. It expands the DevOps practices used with massive success in the software industry to the paradigms of new space and serves as a foundational aspect of our role as a revolutionary space infrastructure service provider.

Acknowledgements

We would like to thank all our colleagues at Loft Orbital who created and continue to implement SatDevOps, in particular (at time of publication) our flight directors: Gauthier Damien, Rémy Derollez, Hélène Gourlaouen, Brandon Hing, Thomas Léonard, Caleb MacLachlan, Thomas Michel, Brunston Poon, Pablo Salas, Emma Vatine, and Etienne Vincent; and our satellite operators: Bastien Arata, Xavier Almenar Cerdan, Matthijs van Duijn, Mattia Fioraso, Laura García, Joshua Hurley, Sarah Lappin, Jaelin Mccreary, Thibaut Miquel, Laurent Rivière, Grant Schmaedick, and Alexandre Tant—all of whom are engineers building and running the space infrastructure that Loft Orbital is pioneering while also being the driving force of SatDevOps.

List of references

[1] L. Brémond, B. Poon, G. Damien. Future-Proof Mission Control Systems: Leveraging Agnostic Design for Autonomous and Event-Driven Satellite Operations, IAC-22-B6.5.7x68666, 73rd International Astronautical Congress, Paris, France, 2022, 18 – 22 September.

[2] P-D. Vaujour and L. Brémond. System and Method for Providing Spacecraft-Based Services. U.S. Patent No. 10,981,678. Washington, DC: U.S. Patent and Trademark Office, 2019.

[3] F. Sellmaier, T. Uhlig, M. Schmidhuber (Eds.), Spacecraft Operations, second ed., Springer Cham, New York 2022.

[4] B. Beyer, N.R. Murphy, D.K. Rensin, K. Kawahara, S. Thorne (Eds.), The Site Reliability Workbook, first ed., O'Reilly, Sebastopol, United States, 2018.

[5] Loft Orbital Solutions, Inc., SatDevOps™, 2023.

[6] Amazon, What is DevOps? https://aws.amazon.com/devops/what-is-devops/, accessed 10 January 2023.

[7] L. Brémond, E. Vincent, G. Damien, C. MacLachlan, B. Poon. Lessons Learned from Automating Heterogenous Spacecraft Systems, 17th International Conference on Space Operations, 2023, 6 – 10 March.

[8] R. Derollez, R. Petitdemange, L. Brémond. Automated Planning and Scheduling System for a Heterogeneous Spacecraft Constellation, 17th International Conference on Space Operations, 2023, 6 – 10 March.

[9] E. Ries. The Lean Startup, first ed., Currency, New York, 2011.