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

SpaceOps-2023, ID #615

Applying DevOps strategies to Monitoring and Control IV&V activities – practical results from Meteosat Third Generation

J. Feiteirinha[a]*, P. Servan[b], T. Pfeiffer, Iskren Georgiev[c]

[a] *Vision Space Technologies GmbH*, jose.feiteirinha@visionspace.com
[b] *European Organization for the Exploitation of Meteorological Satellites*, pierre.servan@eumetsat.int
[c] *CGI Deutschland GmbH*, tilman.pfeiffer@cgi.com, iskren.georgiev@cgi.com
* Corresponding Author

Abstract

DevOps is a set of philosophies and practices aimed to improve the software development cycle. While it is still considered a novelty in mission-critical activities such as space mission operations, there are situations where DevOps-inspired approaches can offer considerable advantages. In the case of the Meteosat Third Generation (MTG) mission, a DevOps approach was used to configure and validate the Generic File Transfer System (GFTS), a distributed system responsible for managing the copying of file products across the MTG Mission Operations Facilities (MOF). The challenge was to ensure consistency and reliability of the GFTS configuration across all environments, while supporting the tight schedule. With no standard tools available, a custom solution was developed using the Gitlab CI/CD suite. This involved the use of an Excel file to contain and compare all the GFTS configuration, stored in a Git repository. Each commit then triggered DevOps pipelines which could be used to generate XML configuration files, consistency checks, comparison with previous commits and even documentation. The impact of this DevOps based solution was to greatly speed up existing processes while simultaneously increasing visibility and clarity. It also enabled automatic checks and documents that were previously unavailable. This solution was critical for the successful operational readiness of the MTG mission, and can be replicated in other mission-critical activities. To evaluate the usage of DevOps solutions other phases and areas of Space missions, we then considered several ECSS standards. These presented particular challenges, as they refer to very specific technical processes. Nevertheless, it is possible to use DevOps in these mission-critical contexts, by adapting the philosophies and practices to specific requirements. The benefits of flexibility, automation and speed can thus be leveraged, while upholding high levels of safety and reliability. In conclusion, this paper presents a successful case of DevOps usage in a mission-critical activity and demonstrates that, when adapted to the specific requirements of the domain, DevOps solutions can offer significant advantages to mission operations.

Keywords: DevOps, ECSS, GFTS, Meteosat, MTG

Nomenclature

IVV and *IV&V* – IVV typically stands for *Independent Verification and Validation*, however at EUMETSAT, the acronym has long been used for *Integration, Verification and Validation*. In what is relevant for this paper, the *Integration, Verification and Validation* for MTG is done by an *independent* nonpartisan team from the users or developers, this is called the IVV team. One could potentially use the term IIVV standing for *Independent Integration, Verification and Validation*, however for clarity this paper simply uses IVV.

DevOps & CI/CD (continuous delivery and/or continuous deployment) - DevOps and CI/CD are closely related, but they are not the same thing. DevOps is a cultural and philosophical approach to software development that emphasizes collaboration, communication, and integration between software developers and IT professionals. CI/CD, on the other hand, refers to the use of automation tools to continuously build, test, and deploy software. The overlap between DevOps and CI/CD lies in their shared focus on collaboration, communication, and automation. Both DevOps and CI/CD aim to improve the efficiency and effectiveness of the software development process by breaking down silos between teams, enabling better communication and collaboration, and using automation to reduce the time and effort required to build, test, and deploy software.

Acronyms/Abbreviations

Acceptance Review (AR)
Backup [Satellite Control Centre] Environment (BSCC)
Consultative Committee for Space Data Systems (CCSDS)
Critical Design Review (CDR)

Commissioning Result Review (CRR)
End-of-Life Review (ELR)
Flight Dynamics System (FDS)
Flight Readiness Review (FRR)
Generic File Transfer System (GFTS)
Integration environment (IV)
Interface Control Document (ICD)
Independent Integration, Validation and Verification (IVV)*
Local Control System (LMC)
Launch Readiness Review (LRR)
Mission Close-out Review (MCR)
Mission Control System (MCS)
Mission Definition Review (MDR)
Mission Operations Facility (MOF)
Operational Environment (OPE)
Operations Preparation Environment (OPRE)
Operational Readiness Review (ORR)
Payload Data Acquisition and Processing (PDAP)
Preliminary Design Review (PDR)
Preliminary Requirements Review (PRR)
Qualification Review (QR)
Simulator (SIM)
System Requirements Review (SRR)
Validation environment (VAL)

1. Introduction

DevOps is a proven and well-known set of philosophies and practices aimed to improve the software development cycle. DevOps is typically associated with Agile methodologies [2]. These are antagonistic of the Waterfall and similar models used in almost any critical space activity [3], such as satellite on-board software or mission control systems. Nevertheless, over the years DevOps has creeped into the space domain, even if indirectly. Evermore small and nanosats use strategies, tools and techniques that are inspired if not copied directly from DevOps workbooks.

In more traditional, robust missions however, DevOps is still considered a novelty or risk and is not part of the plan. Due to its origins with Agile development, it is often considered as unreliable and incapable of providing the level of warranties necessary for larger projects [4]. This is not an incorrect assumption, Agile methodologies make the most use of DevOps and typically rely on changing requirements and adapting the needs. These “advantages” are irrelevant to any large mission, which has a large and clearly defined set of goals, sub-goals and requirements that must be achieved for the mission to succeed.

That being said, defenders of Agile methodologies will argue, without being wrong, that schedules are rarely met and that more interactions (such as those in the DevOps approach) are always required. In fact, it has been our experience that this level of interaction between developers, testers and end-users consistently increase as schedules get tighter.

One critical step in these Waterfall inspired processes is the Integration, Validation and Verification (IVV). The IVV team has the responsibility to independently test and validate the systems under development. On one side, more errors found by the IVV team, may impact the schedule ahead. On the other hand, if IVV is accelerated and errors are missed, then it may have a worse impact later on. Nevertheless, the allocated time for IVV is not infinite and as schedules get tighter, speeding-up IVV may look like a tempting gamble. This temptation works on managers

* See *Nomenclature*.

(that must ensure the schedule is met), and end-users that want to use the latest functionalities as fast as possible. However, it naturally carries some risks.

The standard methods used by IVV to ensure coverage with shortening schedules, are work preparation and parallelisation. Often test procedures (automated or manual) are tested, validated, repeated and fine-tuned to ensure the smooth working of the IVV team when the final deliveries arrive. Furthermore, teams can be split or increased to allow parallel work, so that more can be achieved in shorter time periods. Clearly there are limits to this kind of optimization. Furthermore, it's clear that the faster the pace of software development, the more critical and relevant is the value of IVV. The worst scenario would be that more and more software is delivered without being properly tested, this is the "growing monster" scenario, which must be prevented if the mission is to succeed.

In Meteosat Third Generation (MTG), the Monitoring and Control IVV team is responsible for independently integrating, validating and verifying the Mission Control System and all related interfaces, as well as ancillary systems. This large and complex set of systems is part of the MTG Mission Operations Facilities (MOF) which together with the Payload Data Acquisition and Processing (PDAP) facilities comprises all environments, software and hardware available for MTG spacecraft operations.

An "environment" is the name given by MTG to any particular instance of the hardware and software required to perform monitor and control operations. Multiple instances exist, including the IV environment – for initial delivery, integration and initial verification; VAL used by the IVV team; OPE and BSCC for nominal and contingency operations (note that this is coarsely presented for clarity purposes). While the environments are very similar to each other, they are not exactly the same. For example, some context files need to be periodically copied from the operational (OPE) environment to the backup (BSCC), additionally some resources are necessarily shared such as simulators or ground-stations. Also the IV environment has limited access to other parts of the EUMETSAT infrastructure and parts of MTG and in some cases only interfaces with stub machines. These differences increase the necessary complexity of some systems.

One such system, is the Generic File Transfer System (GFTS), a product developed by the European Space Agency [5] and used as an element in the MTG ground segment. GFTS is used to orchestrate the copying of file products across the MOF and to/from externals. It consists of a distributed architecture with different nodes responsible for managing different file types. Each environment has its own independent GFTS configuration. For MTG this currently consists of a considerable amount of configuration files per environment, then multiplied by the number of environments. This number will further increase as more satellites are added in the scope of the MTG programme. The GFTS configuration is delivered with each MOF version and pre-customized for the IV environment, stub machines are used to simulate external interfaces.

This implies that while the IV environment can work out-of-the-box, necessary changes are required for each other environment. Furthermore, occasionally tests with external entities are necessary which may have become unaligned with the MOF deliveries. This is not uncommon as different parties may each have its own schedule, but it means that specific changes may need to be done on the GFTS configuration of any specific environment in order to accommodate test and validation activities.

If not handled carefully, this could easily spin out of control, with each GFTS environment evolving independently from the others. To make the process further complicated, changing the GFTS configuration requires using a user interface which means separately connecting to each machine of each environment.

It should now be clear, that each MOF delivery would bring new GFTS changes which in turn require careful comparison of XML files across 4 different environments. To further complicate the task, GFTS XML files are large, complex and unsorted which renders file comparison impractical. This bottleneck with each delivery was handled by the IVV team, as they needed to migrate the GFTS configuration from IV to the VAL (and later to OPE and BSCC) environments.

A standard way to process GFTS files was thus necessary, but no such tool was available. In MTG it was decided to use the powerful Gitlab CI/CD suite (Continuous Integration and Continuous Delivery). Our process involves using an Excel file to contain and seamlessly compare all the GFTS configuration in a clear, user friendly and manageable way. This file contains all necessary information across the multiple environments, and is stored in a Git repository. Each commit then triggers DevOps pipelines which can be used - depending on the branch – to trigger the generation of the XML configuration files, consistency checks, comparison with previous commits and even documentation. Gitlab Issues functionality is used to track the changes.

In this paper we present in detail the technical, logistical and processual challenges that were overcome by the use of our solution. We detail the impact of these DevOps based solutions in supporting the MTG launch and operational readiness.

Finally, we map this activity to the larger picture and conclude that this and similar approaches can greatly speed-up existing processes while simultaneously adding clarity and visibility by enabling automatic checks and documents that were previously unavailable.

1.1. DevOps

Although DevOps is becoming a common word in technical environments, there are still several too frequent misconceptions. Because of these all, we find best to first clarify exactly what we mean and what do not mean by DevOps.

1.1.1. DevOps is not a technology

It is an approach to software development that emphasizes *collaboration* between development and operations. By automating processes, organizations can reduce the time it takes to develop and deploy software, improve the quality of the software being deployed, and improve the overall availability of their services.

1.1.2. DevOps is not a position

It is a *culture* that emphasizes the importance of automation, continuous integration and delivery, and the use of DevOps tools to manage the process. DevOps is about creating a unified culture and shared understanding of the business objectives and goals.

1.1.3. DevOps is not a tool

It is a *set of practices* and tools that support collaboration, communication, and automation of the software delivery process. Common DevOps tools are used to automate the process of software delivery and infrastructure changes. But these practices are intended to allow organizations to rapidly develop, deploy, and manage their applications. Tools are the way to achieve DevOps, not DevOps itself.

1.1.4. DevOps is

DevOps is then, rather a set of principles, practices and – yes – tools that enable organizations to improve the speed and quality of their software delivery process, while also improving their overall availability and security. DevOps can help organizations improve communication and collaboration between departments, reduce costs, and increase the speed of software delivery. For development teams, DevOps encourages continuous integration, which means code is constantly tested and deployed to production. This allows developers to make faster changes and receive feedback quickly. It also allows them to release software faster and more reliably. For operations teams, DevOps promotes automation and monitoring of applications, which allows them to quickly detect and address issues. This helps to improve the availability, scalability, and security of applications. For managers, DevOps gives them visibility into the entire software delivery process and helps them make more informed decisions. It also helps them to track progress, identify areas for improvement, and make sure their teams are working efficiently. For costumers, DevOps helps to ensure that software is delivered quickly and reliably. The fast turnaround of new features and bug fixes leads to a better user experience and improved customer satisfaction. With more work getting done faster and for lower costs.

1.2. Agile

Agile software development is an iterative development methodology which encourages collaboration between cross-functional teams. It uses short development cycles and emphasizes delivering working software quickly and frequently. Agile software development focuses on responding to change and delivering value to the customer. Common tools for Agile are user stories, sprints, retrospectives, backlogs, stand-up meetings, and continuous integration.

There are different Agile methods or frameworks that can be applied, such as Scrum, Kanban, Extreme Programming, or Feature Driven Development. Each of these methods typically involve iterative and incremental development, collaboration between cross-functional teams, and the use of adaptive planning and adaptive development. Each also has its own set of practices and tools that can be used to plan, track, and manage the project. Scrum is one of the most popular Agile methods and is based on the use of short cycles of work (sprints) that involve a consistent cadence of planning, development, review, and retrospective activities. Kanban is a visual system for

tracking progress and managing work, and is often used to improve workflow. Extreme Programming (XP) is an Agile methodology that focuses on delivering working software quickly while encouraging collaboration and embracing change. Feature Driven Development (FDD) is an Agile method that focuses on delivering small, incremental, and visible bits of functionality.

1.2.1. Agile and DevOps

The usage of DevOps solutions with Agile is common because it helps to speed up the delivery process and increase the visibility of the project. The DevOps team can automate the release process, test and deploy the software, and monitor the performance of the software in production. The DevOps team can also provide the development team with feedback on the performance of the software in production and assist with troubleshooting and debugging. This helps to speed up the development process and increase the quality of the software.

For this reasons Agile and DevOps are closely related and often used together. One could say DevOps provides the necessary tools and systems to make Agile projects successful. Using Agile without DevOps is possible but can lead to delays in the delivery of the software and lack of visibility. For large projects there are no known advantages in using Agile development without an enabling DevOps solution. Without an enabling DevOps solution, it can be difficult for teams to keep up with the rapid pace of development and delivery that is characteristic of Agile development. A DevOps solution can help teams automate the build, test, and deployment process, which can improve the efficiency and speed of delivery.

Overall, Agile and DevOps are two complementary approaches that can be used together to create an effective and efficient software development process. Agile methodologies help teams to quickly respond to customer feedback and make necessary changes in the product as required, while DevOps helps to ensure that the software product is of the highest quality and can be delivered to customers quickly and without any errors. The combination of Agile and DevOps has allowed teams to develop and deploy software applications faster and with fewer errors, resulting in greater customer satisfaction.

1.3. European Cooperation for Space Standardization (ECSS)

The European Cooperation for Space Standardization (ECSS) is a set of standards and guidelines developed to guide and ensure the quality and reliability of space products and systems. The aim of ECSS is to provide a consistent framework for the development, acquisition and use of space systems. The ECSS standards cover a wide range of topics, including software engineering, hardware engineering, systems engineering and operations. The standards are designed to ensure that products are safe, reliable and effective in any space mission.

ECSS is used by the European space community to guarantee that all space products and systems are designed and developed according to the same standards. This ensures that the space products are safe, reliable and effective for their intended purpose.

1.3.1. ECSS and Agile

“Agile Software Development Handbook” or ECSS-E-HB-40-01A, is the ECSS handbook for Agile software development. This handbook provides guidance on how to use Agile methods in the development of space-related software. The handbook provides guidance on the principles of Agile development, how to use Agile methods in the development of space-related software, and how to integrate Agile methods into the larger system development process. The handbook is intended to be a guide to Agile development for experienced software developers, and provides advice and guidance on how to use Agile methods in the development of space-related software.

The ECSS “Agile Software Development Handbook” however, does not mention DevOps. Reading document, specifically chapter 5 "Guidelines for Agile life cycle selection", does not provide information on how to integrate DevOps into the larger system development process. There is no mention of DevOps at all. This is not a critic to the book which is focused on Agile principles, nor on its enabling technologies and cultures. But it obviates the overlook that the Space Industry administers on DevOps.

The ECSS does not require a specific software development methodology to be used, but rather bases its software engineering standards on ISO/IEC 12207 [6,7,8]. This standard promotes strong customer-supplier collaboration and can be easily mapped onto the Agile Scrum framework. It is important for standards like ECSS to stay updated and allow for company innovation, and it is recommended for companies to have a good understanding of ECSS and how it can be tailored to their specific project needs.

1.3.2. ECSS and DevOps

A Google search for "ECSS DevOps" on Google provides a good overview of the lack of a stance for the European Space Standardization on DevOps.

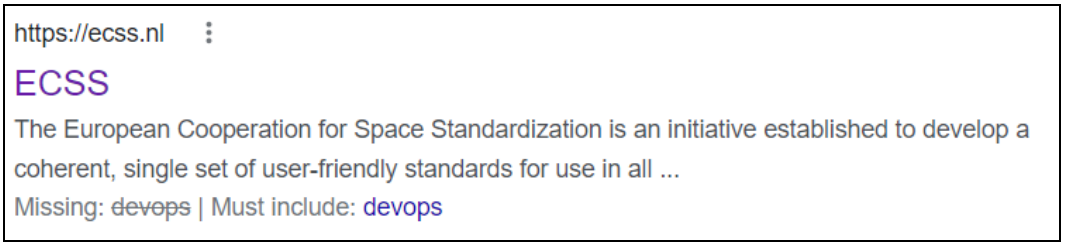


Figure 1. Google result when searching for "ECSS DevOps"

This does not mean that ECSS is unaware of Agile solutions, as can be disproved from the sections above. Furthermore, overlapping elements such continuous build and development are mentioned by the ECSS Agile handbook.

But the combination of Agile and DevOps is a very common, effective and efficient product development workflow that we believe it to be worth specific mentioning. Agile allows the customer to have an active voice in the development process and to make changes as necessary, while DevOps helps to ensure that the necessary changes arrive speedily to the product, and that it is of the highest quality. This combination of Agile and DevOps also enables teams to develop and deploy applications faster and with less risk. The usage of this combined approach has proven to be extremely successful and is common practice. In fact, *Agile software development principles, values and practices are required for successful DevOps adoption* [1].

Therefore, the omission of this evident complementarity points to a potential blind spot on the European Standardization environment. Traditional industries looking to implement or adapt existing solutions into Agile, may achieve sub-optimal solutions if DevOps is ignored.

A small note should add that the Consultative Committee for Space Data Systems (CCSDS) also omits DevOps, and a single entry is found on its site, a call for papers in 2021 did mention DevOps usage for ground systems engineers.

2. Space Project Management

ECSS provides a common framework for managing space projects, and helps to ensure that they are carried out in a consistent, efficient, and effective manner.

The ECSS defines a number of key principles and practices for space project management, including a clear and well-defined mission statement, which sets out the goals and objectives of the project, and provides a common reference point for all project activities; a systematic and structured approach to project planning and execution, which involves breaking the project down into smaller, manageable activities, and defining the interdependencies between them; a robust and transparent governance structure, which ensures that the project is managed effectively and efficiently, and that decisions are made in a consistent and fair manner; and a focus on quality, reliability, and safety, which ensures that the project delivers the required results, meets the necessary standards, and operates safely and securely.

Overall, the ECSS provides a comprehensive framework for managing space projects, and helps to ensure that they are carried out in a consistent and effective manner.

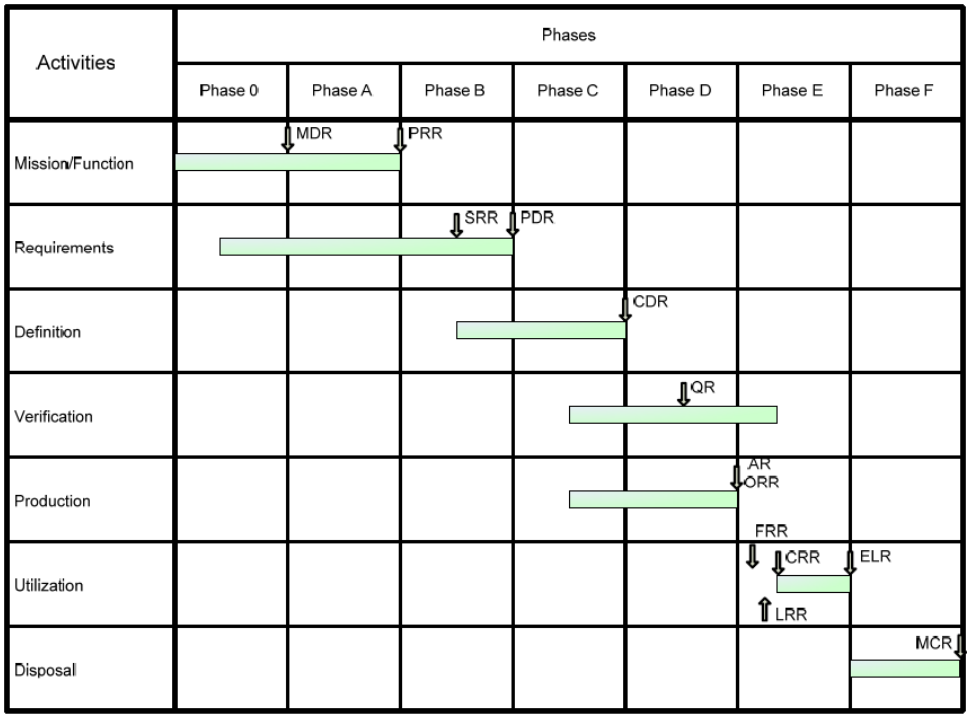


Figure 2 - Project management activities as per ECSS-M-ST-10C

2.1. Phases

ECSS sets standards for space engineering and project management, the phases of a space project defined by the ECSS are described below. Each of these phases is interconnected and builds upon the work done in the previous phases. For example, the feasibility study conducted in Phase A will inform the detailed design work done in Phase C, and the testing and analysis conducted in Phase D will verify that the spacecraft is fit for its intended purpose.

2.1.1. Phase 0: Mission analysis/needs identification

This phase involves identifying the goals and objectives of the mission, and determining the requirements and constraints that must be met in order to achieve them. This involves conducting a thorough analysis of the mission context, including its scientific, technical, and operational aspects.

2.1.2. Phase A: Feasibility

In this phase, the feasibility of the mission is assessed, including its technical, financial, and schedule feasibility. This involves conducting a detailed analysis of the proposed mission concept, and assessing its potential risks and challenges.

2.1.3. Phase B: Preliminary Definition

In this phase, the mission concept is further refined and defined. This involves developing a detailed mission plan, as well as defining the requirements for the spacecraft and its subsystems.

2.1.4. Phase C: Detailed Definition

In this phase, the detailed design of the spacecraft and its subsystems is developed. This involves conducting extensive testing and analysis to ensure that the design meets all of the requirements and constraints defined in the previous phases.

2.1.5. Phase D: Qualification and Production

In this phase, the spacecraft and its subsystems are built and tested to ensure that they meet the requirements and constraints defined in the previous phases. This involves conducting extensive testing and analysis to verify that the spacecraft and its subsystems are fit for their intended purpose.

2.1.6. Phase E: Utilization

In this phase, the spacecraft is launched and begins its mission. This phase involves conducting the activities necessary to accomplish the mission objectives, such as deploying the satellites and conducting scientific experiments.

2.1.7. Phase F: Disposal

In this phase, the spacecraft is decommissioned and disposed of in an appropriate manner. This may involve deorbiting the spacecraft, or safely disposing of it in another way.

~~*2.2. Reviews*~~

~~Each project includes several reviews, detailed below:~~

- ~~MDR mission definition review~~
- ~~PRR preliminary requirements review~~
- ~~SRR system requirements review~~
- ~~PDR preliminary design review~~
- ~~CDR critical design review~~
- ~~QR qualification review~~
- ~~AR acceptance review~~
- ~~ORR operational readiness review~~
- ~~FRR flight readiness review~~
- ~~LRR launch readiness review~~
- ~~CRR commissioning result review~~
- ~~ELR end of life review~~
- ~~MCR mission close out review~~

~~The reviews listed above are key milestones in the development of a space project, and are typically conducted at the end of each project phase. These reviews serve as a way to assess the progress of the project and ensure that it is on track to meet its goals and objectives.~~

~~The MDR (mission definition review) is typically conducted at the end of Phase 0 to review the mission concept and plan to ensure that it is feasible and aligns with the overall goals and objectives of the project. Similarly, the PRR (preliminary requirements review) and SRR (system requirements review) are conducted at the end of Phases A and B, respectively, to review the requirements for the spacecraft and its subsystems to ensure completeness and adequacy. The PDR (preliminary design review) is conducted at the end of Phase B to review the preliminary design of the spacecraft and its subsystems. The CDR (critical design review) is conducted at the end of Phase C to review the final design of the spacecraft and its subsystems before production and testing. The QR (qualification review) is conducted during Phase D to review the results of testing and analysis, and the AR (acceptance review) is conducted at the end of Phase D to review the spacecraft and its subsystems to ensure readiness for the next phase. The ORR (operational readiness review) and FRR (flight readiness review) are conducted during Phase E but before launch, respectively, to review the readiness of the spacecraft and its subsystems. The LRR (launch readiness review) is conducted just before launch to review the readiness of the spacecraft and its subsystems, as well as the launch vehicle and other elements of the launch system. The CRR (commissioning result review) is conducted after launch to review the results of the commissioning phase, and the ELR (end of life review) and MCR (mission close out review) are conducted at the end of the mission to review the performance of the spacecraft and its subsystems, as well as the success of the mission.~~

~~As an example of the reviews listed above, let's consider a mission to explore the outer reaches of the Jupiter system. The mission begins with the MDR (mission definition review) at the end of Phase 0, where the mission concept and plan are reviewed to ensure that they are feasible and consistent with the overall goals. During Phase A's~~

~~PRR (preliminary requirements review), the requirements for the spacecraft and its subsystems are reviewed to ensure that they are complete and well-defined. The SRR (system requirements review) at the end of Phase B reviews the detailed requirements of the spacecraft and its subsystems to ensure that they are adequate for the design phase. The PDR (preliminary design review) at the end of Phase C reviews the preliminary design of the spacecraft and its subsystems to assess whether it meets the requirements and constraints defined in the previous phases. The CDR (critical design review) at the end of Phase C reviews the final design of the spacecraft and its subsystems to make sure that it is ready for production and testing. The QR (qualification review) during Phase D reviews the results of the testing and analysis to ensure that the spacecraft and its subsystems are fit for their intended purpose. The AR (acceptance review) at the end of Phase D reviews the spacecraft and its subsystems to ensure that they are ready to be accepted for use in the next phase. The ORR (operational readiness review) at the end of Phase E reviews the readiness of the spacecraft and its subsystems for the operational phase of the mission. The FRR (flight readiness review) before the launch of the spacecraft reviews the readiness of the spacecraft and its subsystems for launch and the operational phase of the mission. The LRR (launch readiness review) just before the launch of the spacecraft reviews the readiness of the spacecraft and its subsystems for launch, as well as the readiness of the launch vehicle and other elements of the launch system. The CRR (commissioning result review) after the launch of the spacecraft reviews the results of the spacecraft's commissioning phase, including any issues or problems that may have arisen. The ELR (end-of-life review) at the end of the spacecraft's mission reviews the performance of the spacecraft and its subsystems during its operational phase, as well as any issues or challenges that may have arisen. Lastly, the MCR (mission close-out review) at the end of the spacecraft's mission reviews the overall performance of the spacecraft and its subsystems, as well as the success of the mission in achieving its goals and objectives.~~

~~These reviews, phases and processes are sequential, and seemingly opposed to Agile methodologies. But they are in fact not completely incompatible with Agile. There are reasons for each of the reviews, but nothing opposes that during each activity Agile and DevOps solutions are not used. In fact, modern space projects are increasingly moving towards Agile and DevOps approaches, and the reviews mentioned above can be adapted to such approaches to ensure project success. One example of this is the use of automated testing, which allows more frequent reviews of progress and reduces the need for manual testing. Additionally, Agile and DevOps approaches allow for more frequent and incremental updates, which can be reviewed more quickly and effectively than with traditional approaches. Agile and DevOps approaches also allow for more efficient collaboration between stakeholders, which can lead to better decisions and faster progress. Even with these reviews, Agile and DevOps can still be used to ensure that the project is progressing as planned and that it meets the goals and objectives set for it. In fact, many space projects already partially use Agile or DevOps inspired approaches to ensure success. What lacks is the integration of these methodologies in a full and consistent way with ECSS project management. The reviews listed above are key milestones in the development of a space project, and are necessary for the successful completion of the project. However, these reviews can also be merged with modern, Agile and DevOps approaches to ensure that the project is progressing as planned and meets the goals and objectives set for it, while simultaneously reducing risk and increasing efficiency.~~

~~2.2.~~2.2. *Software Development*

The reviews listed above are relevant to software development in the space domain, including the development of mission control software. In the context of software development, these reviews can serve as a way to assess the progress of the software development project and ensure that it is on track to meet its goals and objectives.

For example, the MDR (mission definition review) can be used to review the requirements and constraints for the mission control software, and ensure that they are feasible and well-defined. The PRR (preliminary requirements review) can be used to review the initial requirements for the software, and ensure that they are complete and well-defined. The SRR (system requirements review) can be used to review the detailed requirements for the software, and ensure that they are adequate and can be used to guide the design and development work.

Like in the development of the Space segment, for software development the PDR (preliminary design review) can be used to review the initial design of the software, and ensure that it meets the requirements and constraints defined in the previous phases. The CDR (critical design review) can be used to review the final design of the software, and ensure that it is ready for implementation and testing. The QR (qualification review) can be used to review the results of the testing and analysis conducted on the software, and ensure that it is fit for its intended purpose. The AR (acceptance review) can be used to review the software to ensure that it is ready to be accepted for use in the next phase. The ORR (operational readiness review) can be used to review the readiness of the software for use in the operational phase of the mission. The FRR (flight readiness review) can be used to review the readiness of the software for use in the operational phase of the mission, including its readiness for launch.

3. Hypothesis

The development of software in the space domain can be subject to a range of bottlenecks or obstacles that can hinder progress. Some of the common bottlenecks that may arise in the development of space software. The development of space software can require significant resources, including funding, personnel, and equipment. In case of limited availability of resources, if these are not available in sufficient quantities, it can be difficult for the project to progress as planned, and it may be necessary to adjust the project scope or schedule to compensate.

Furthermore, Space projects often involve complex technical challenges and a high degree of uncertainty. This can make it difficult to predict the outcomes of the project, and can lead to delays and unexpected challenges. Also, space projects often rely on advanced technologies and systems, which can be challenging to develop and implement. If these technologies are not available or are not reliable, it can be difficult for the project to progress as planned, and it may be necessary to explore alternative solutions. Overall, the development of software in the space domain can be subject to a range of bottlenecks and obstacles that can hinder progress. To overcome these challenges, it is important for project managers to have a clear understanding of the goals and objectives of the project, to have access to sufficient resources, and to be prepared to adapt to changing circumstances and challenges.

Agile software development and other methodologies may help to mitigate some of the risks associated with the development of space software. Agile development focuses on rapid iteration and flexible adaptation to changing needs, which can be particularly beneficial in the development of space software, where technological and environmental conditions can change rapidly. It can help to ensure that the software is developed with a focus on the user's needs and requirements, and can help to ensure that the development process is as efficient and effective as possible.

However, Agile processes should also be complemented with structured testing and validation in order to ensure the quality and reliability of the software. They should also be used in tandem with other development methodologies and techniques, such as requirements management, risk management or DevOps, in order to ensure that the project is on track and that risks are managed effectively.

To mitigate the above, processes should exist to automate the entire projects wherever possible. This involves creating a workflow to ensure that all tasks are completed in the correct order, as well as tracking progress to ensure that nothing is missed. This is where DevOps can help to reduce the bottlenecks in the development of software in the space industry. By improving the efficiency and effectiveness of the software development process. By using DevOps practices and tools, organizations can speed up the development and delivery of software, and can reduce the time and effort required to get new features and updates into the hands of users.

For example, the use of automation can help to streamline and accelerate the development and testing of space software. By automating routine tasks, such as building, testing, and deploying software, organizations can reduce the amount of manual effort required, and can get new features and updates into the hands of users more quickly. Additionally, the use of continuous integration and delivery (CI/CD) can help to reduce the time and effort required to release new software updates.

A disadvantage of using CI/CD is that it can introduce more complexity to the software development process, as it requires the implementation of automated testing and deployment processes. This can be difficult to set up and maintain, especially for organizations that are not already familiar with the tools and practices involved. Additionally, if the CI/CD pipeline is not set up correctly or is not properly maintained, it can lead to issues such as unreliable builds, failed deployments, and broken software. Additionally, if the tests fail it can also cause delays in the release. By using CI/CD tools, organizations can automatically build, test, and deploy new software updates as soon as they are ready, rather than waiting for a scheduled release. This allows organizations to release new features and updates more frequently, and to respond quickly to changes and feedback from users.

This means that issues get fixed faster, users get the fixes faster and can improve on it. The effective collaboration and communication is essential for successful DevOps. By fostering a culture of collaboration and communication among different teams and functions, organizations can improve their ability to coordinate and work together effectively, and can reduce the risk of delays and misunderstandings. This can be particularly important in the space industry, where projects are often complex and involve a high degree of uncertainty.

The clear benefit of DevOps is the way it streamlines the software development process, resulting in faster and more frequent releases of software. As we've shown above, this has - so far - not been taken into consideration by ECSS. In fact, due to the complexity of the software development process in the space industry, the lack of DevOps can be a key factor in the success or failure of a project. For example, if the development process is slow or inefficient, this can lead to delays or cost overruns, which can have a significant impact on the success of the project. By introducing DevOps practices and tools, organizations can reduce the time and effort required to develop and deploy space software, and can ensure that projects are completed on time and on budget.

The traditional resistance to DevOps in the space industry has been due to the perceived complexity and uncertainty that comes with the software development process. However, with the increasing availability of tools and technologies, and with the growing understanding of the benefits of DevOps, organizations in the space industry are beginning to embrace DevOps practices and tools. We've also shown that DevOps does not exclude proven ECSS standards, and that it can be used to complement and improve existing processes. By adopting DevOps practices, organizations in the space industry can accelerate their development process, reduce their costs, and ensure that their projects are successful.

3.1. DevOps and Reviews

It should clear that DevOps does not replace, remove, or decrease the value of the existing quality gates, such as PDR or CDR. Having more deliveries does not mean less reviews. However, DevOps does mean more information is available at the time of each review. It allows users to provide feedback within each phase and activity, not just at the review. Thus it is expected that more issues have already been clarified and fixed by the time of the review. DevOps also provides a continuous feedback loop, and but does not exclude the need for reviews, or the reviews to be reduced or eliminated entirely. In fact, DevOps can be used to provide more information to the reviewers, and to make the review process more efficient.

For example, DevOps can be used to provide more detailed information about the project at the time of the review, such as the current status of the project, the progress to date, any issues that may have arisen, and any changes that have been made. This can help the reviewers to more quickly evaluate the project, and to more accurately determine whether the project is on track and ready to move forward. Additionally, DevOps can be used to provide continuous automated testing and validation of the software, which can help to reduce the time and effort required to complete the review process.

It can also help to ensure that the software maintains the requirements and constraints defined in the previous phases. It can thus enhance and improve the review process. For example, DevOps can be used to provide automated testing and validation during the PDR, CDR, and QR reviews. This can help to ensure that the software meets the requirements and constraints defined in the previous phases, and can reduce the time and effort required to complete the review process. It can also be used during the ORR, FRR, and LRR reviews to ensure that the software is ready for launch, and that the launch system is functioning correctly. In fact, it can be used from the start of the software development, through the full software development cycle.

3.2. DevOps and Workload

A common misconception is that more deliveries imply more workload. This can be opposite to reality, but workload needs to be shifted differently. That is, DevOps does not require more work, but rather, it moves the work to different points in the development cycle. It can provide a more efficient way of working, as it allows for quicker feedback, more frequent releases, and more efficient testing and validation. Rather than one large acceptance test with each delivery, faster feedback loops allow for smaller tests and quicker fixes, and the workload can be spread out over the duration of the project. This can help to reduce the amount of work required at any one point in time, and can help to ensure that the project is on track and meeting its goals and objectives. Yes, full reviews are still needed. They have a particular impact on regression testing, to ensure that newer fixes did not break existing functionality. However, the DevOps process can make this review process more efficient, as it provides a more detailed view of the project and its progress, and allows for automated testing and validation throughout the development process. This means less spot-checks on new functionalities, but the same level of full-regression testing. It also means that any issues found can be quickly fixed and re-deployed for testing. Shortening the time between the detection of the issue and its fix, as well as the amount of time needed for deployment and testing. This can reduce the time and effort needed for the full review, and can help to ensure that the project is on track and meeting its goals and objectives. However, all of this is done without an increased workload. Users and testers will always raise anomalies, but then spend time finding - often grotesque - solutions to work around issues that developers are equipped to quickly fix. But if each delivery is slow and manual, then a short term solution is required. This multiplies the total workload needed to fix an issue. DevOps does not require an increased workload, but instead moves the work to different points in the development cycle, and can thus provide a more efficient way of working.

3.3. DevOps and Quality

DevOps is not a replacement for quality assurance and testing. Quality assurance and testing is still necessary to ensure that the software meets the requirements and constraints defined in the previous phases. However, DevOps can be used to improve the quality assurance and testing process. For example, automated testing and validation can be used to continuously test the software throughout the development process, rather than waiting for a scheduled release. This can help to reduce the time and effort required to test the software, and can help to ensure that the software is of high quality and reliability.

4. Case Study

4.1. The Mission Operations Facility (MOF)

The Meteosat Third Generation (MTG) mission is an international collaboration between the European Space Agency (ESA), the European Organisation for the Exploitation of Meteorological Satellites (EUMETSAT) and the European Commission (EC). MTG is a new generation of weather satellites, that are designed and built to provide observations of the Earth's atmosphere and surface.

The MTG ground segment is comprised of all the software, hardware and facilities necessary that allow the operation of the mission. The Mission Operations Facility (MOF) is a critical component of the MTG mission. It contains all of the virtual machines (VMs) necessary for the successful operation of the mission, including the Mission Control System (MCS), Flight Dynamics, Mission Planning, Automation, Simulation, and Operations Preparation.

The MOF is divided into different environments, each one is an isolated instance of the necessary software and hardware to operate the mission. The environments are:

- IV: Initial Verification environment, where most of the initial development and delivery activities take place.
- VAL: Validation environment, where the validation team is responsible for independent testing and verification of the delivered products.
- OPE: Operational environment, where the mission control system is operated in nominal operations mode.
- BSCC: Backup and Contingency Control environment, which is a mirror of the OPE environment with the added capability to be used in contingency operations.
- Sandpits: standalone test environments used for testing of operational scenarios and anomalies.

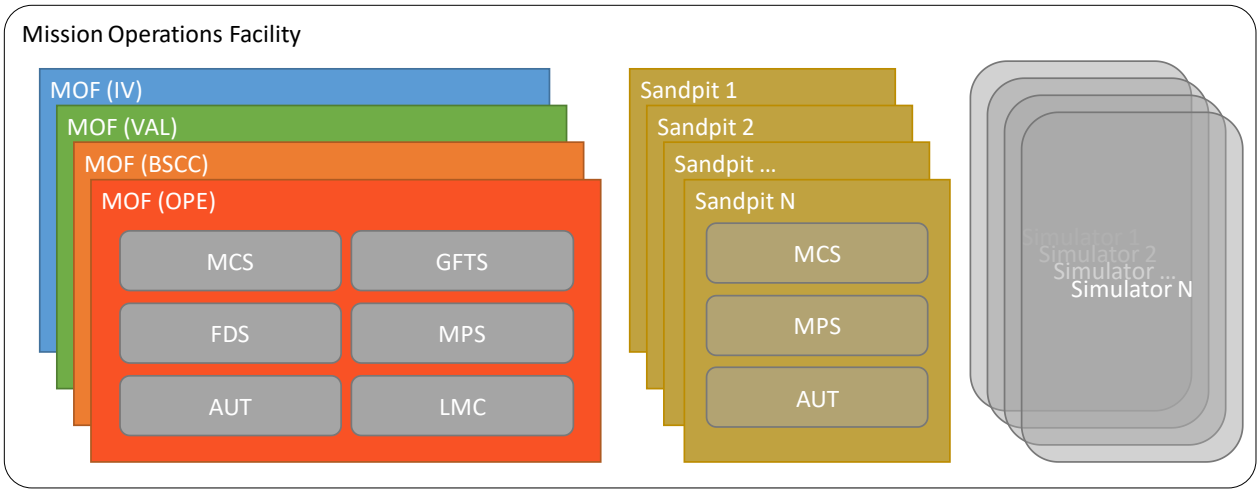


Figure 3 - Overview of the MOF components

The specific responsibilities of the MOF in the MTG mission include providing the necessary applications for the successful ~~operation~~ monitoring and control of the mission. The MOF together with the Payload Data Acquisition and Processing (PDAP) ensure the successful operation of the mission. The MOF includes the MCS, which is used to

operate the satellite and send commands and telemetry, as well as the Automation system, which is used to automate the use of the MCS and support conditional executions, parameter processing, and decision trees.

The Mission Planning system is used to orchestrate the Automation system and support mission-level planning of activities, including the creation of Gantt charts to visualize planned executions. The Flight Dynamics system is used to plan spacecraft manoeuvres and provide the necessary inputs to other systems to ensure successful satellite operation. Finally, the Simulation system provides the ability to simulate the spacecraft and ground stations, allowing for safe testing of procedures.

The various environments and sandpits in the MOF serve different purposes. OPE is used for nominal operations, while BSCC serves as a backup to OPE at a different site. VAL is used for validation of applications and procedures, and IV is used for initial integration and testing. The sandpits, on the other hand, are used for testing by users, and include a simulator, MCS, and Automation systems.

In order to ensure that the MTG mission is carried out successfully, the MOF provides the necessary software tools and systems. This includes the MCS, Automation system, Mission Planning, Flight Dynamics, and Simulation systems, all of which are critical for the successful operation of the satellite and the execution of the mission plan. The MOF plays a vital role in the MTG mission, providing the necessary tools and systems for the successful operation of the satellite and the execution of the mission plan. Without the MOF, the mission would not be able to proceed as planned.

4.1.1. The Generic File Transfer System (GFTS)

GFTS is used to orchestrate the copying of file products across the MTG Mission Operations Facility and it has a distributed architecture with different nodes responsible for managing different file types, it can also send files to non-GFTS machines. Configuration of GFTS is done via XML files, one for each GFTS node and one global one, and each environment has its own independent configuration.

GFTS configuration is pre-customized for the IV environment and delivered with each MOF version, and external interfaces are simulated with stub machines. Changes are required for each environment and sometimes tests with external entities are necessary which may have become unaligned with the MOF deliveries, requiring specific changes on the GFTS configuration of any specific environment.

4.2. Technical and Processual Challenges

The initial Generic File Transfer System (GFTS) configuration update process was complex and time-consuming.

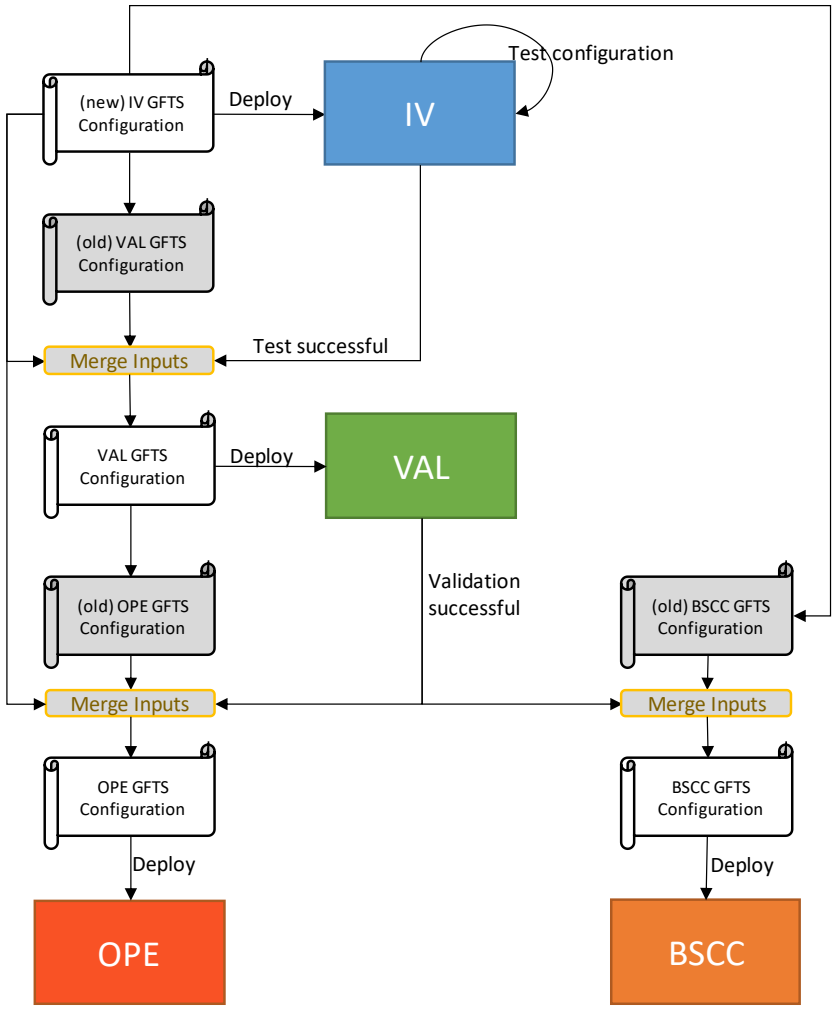


Figure 4 - Simplified diagram of the original GFTS configuration update process

It started when a configuration file was delivered by the company responsible for the configuration. However, these files contained stub machines, which needed to be checked and validated on the integration environment (IV). Once this was done, the configuration was customized for the validation (VAL) environment.

The configuration files were large and unsorted, making it difficult to compare them side-by-side to identify changes. To check and validate the configuration in the IV environment, the process involved manually copying multiple configuration files to each node. The GFTS application was then started on each node, and the new configuration was imported. The central node file was imported first, followed by the other files. Once this was done, the configuration was checked to ensure that it had been loaded properly and that all files were valid.

After validating the configuration in the IV environment, the next step was to update the configuration in the VAL environment. This was a slow and error-prone process, as it involved manually comparing and updating each of the many configuration files. Any changes that had been made to the VAL configuration since the IV configuration was delivered needed to be taken into account.

Due to the large and unsorted nature of the XML configuration files, this process required carefully comparing each configuration element side-by-side using the GFTS client applications and updating the configurations accordingly. This added to the complexity and potential for errors in the configuration update process.

Once the configuration in the VAL environment had been validated, the same process was repeated for the OPE and BSCC environments. This further added to the complexity of the process, as different configurations had to be kept in mind for each environment. For example, the Instrument Data Processing Facility (IDPF) receives files from OPE but not from BSCC.

To avoid errors in the configuration update process, the main requirement was to be diligent. However, as the process became more complex and time-consuming, it became unsustainable. This led to each environment having its own independent configuration, which was not ideal. Newer – necessary – changes were done ad-hoc and hundreds of independent – yet related – configurations were co-existing. Furthermore there was no version control of the changes, making mistakes hard to track and correct, and leading to increased risk of errors and inconsistencies.

This is clearly sub-optimal, and in fact counter-productive in terms of cost-effectiveness, scalability and long-term maintainability. In order to address this issue, the process needed to be updated to allow for a more efficient and effective way of updating the shared GFTS configuration.

4.3. Solution

As the multiple environment configurations grew further apart a solution was required to solve the problem of managing the configuration of the Generic File Transfer System (GFTS). This however required more than a simple manpower increase or automation The problem was in the process, which was error prone and produced cumulative errors that could always (eventually) snowball into an incoherent and inconsistent state that was clear to none.

It was thus clear that a process change was necessary.

A new process could never fully automate the “merge inputs” steps listed above, since it requires human knowledge of multiple different ICDs (which can change) and system knowledge.

The decision was then to facilitate the merge inputs step as much as possible.

A few decisions were made:

- A central configuration site should exist where each file/poller/account could be seen side-by-side for each environment. The central site should be as intuitive and as easy to use as possible.
- Automation should facilitate the process as much as possible, so that the amount of manual editing and configuring is minimised.
- Version control was a requirement for all the configuration files.

The solution is a DevOps-inspired solution using an Excel spreadsheet as its knowledge base (for all environments) and CI/CD pipelines to automate the generation. This solution was developed as part of the Monitoring and Control IVV team's efforts to ensure the successful launch of the Meteosat Third Generation (MTG) mission.

The solution involves the use of an Excel file to store the GFTS configuration data for each of the four MTG environments (IV, VAL, OPE, and BSCC). This Excel file is stored in a Git repository, and each commit triggers a set of DevOps pipelines which are used to generate the XML configuration files, perform consistency checks, compare the configuration data to previous commits, and generate documentation. The Gitlab Issues functionality is used to track changes to the configuration. The pipelines also generate documentation via the pipelines, which make each environment comparable not only via Excel, but also via diagrams. The changes are deployed automatically onto the project’s Gitlab wiki.

CI/CD pipelines are used to validate the Excel file, compare it with previous versions, generate the XML configuration files, keep the XML files under version control, and generate the documentation for the GFTS system. Additionally, there is a separate pipeline that can be manually triggered to re-generate the Excel file from the XML configuration.

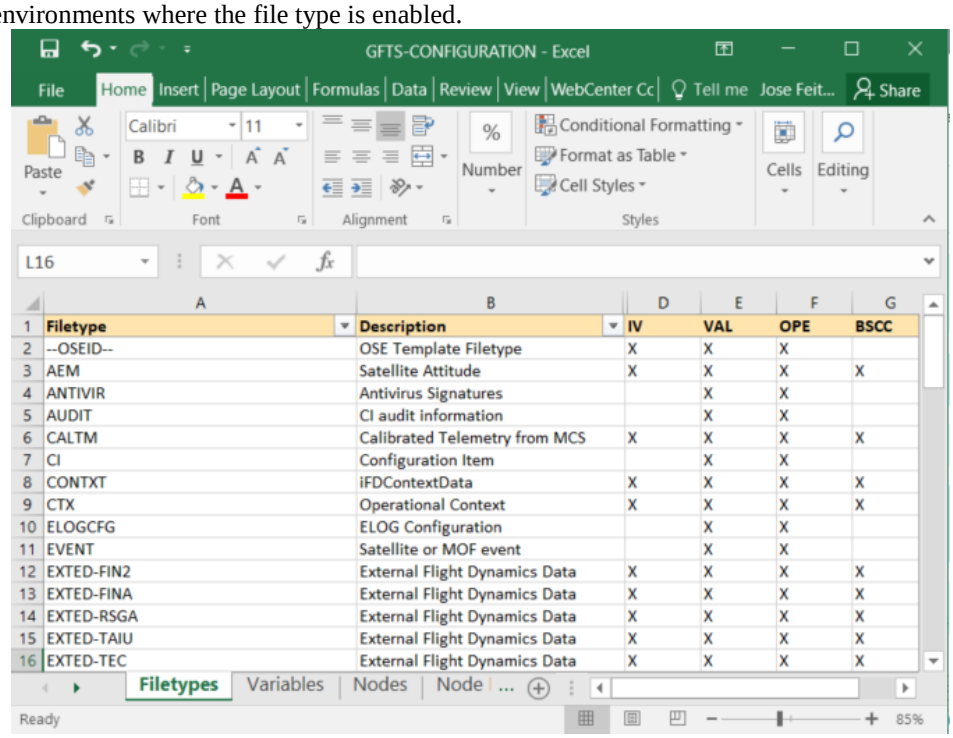
The Git repository contains the GFTS Excel configuration file, as well as a folder with the generated XML configurations. When changes are made to the Excel file and committed to the repository, the CI/CD pipeline generates XML configuration files based on the updated Excel file. These generated XML files are then stored in the repository as artefacts, with each file being renamed to include the branch name and commit ID. These XML files are then used for deployment to the various environments. With each deployment to the validation, operational, or backup environment, a tag is created on the Git repository for the commit that generated that version of the configuration.

4.3.1. GFTS Configuration (Excel)

The Generic File Transfer System (GFTS) is a system for transferring files between different machines and is deployed on the different mission environments. To configure the GFTS system, a GFTS Excel configuration file is used to specify the settings and options for the system in each environment. This Excel file is organized into separate

sheets for different aspects of the GFTS configuration, such as file types, nodes, and pollers. It contains the information for all environments.

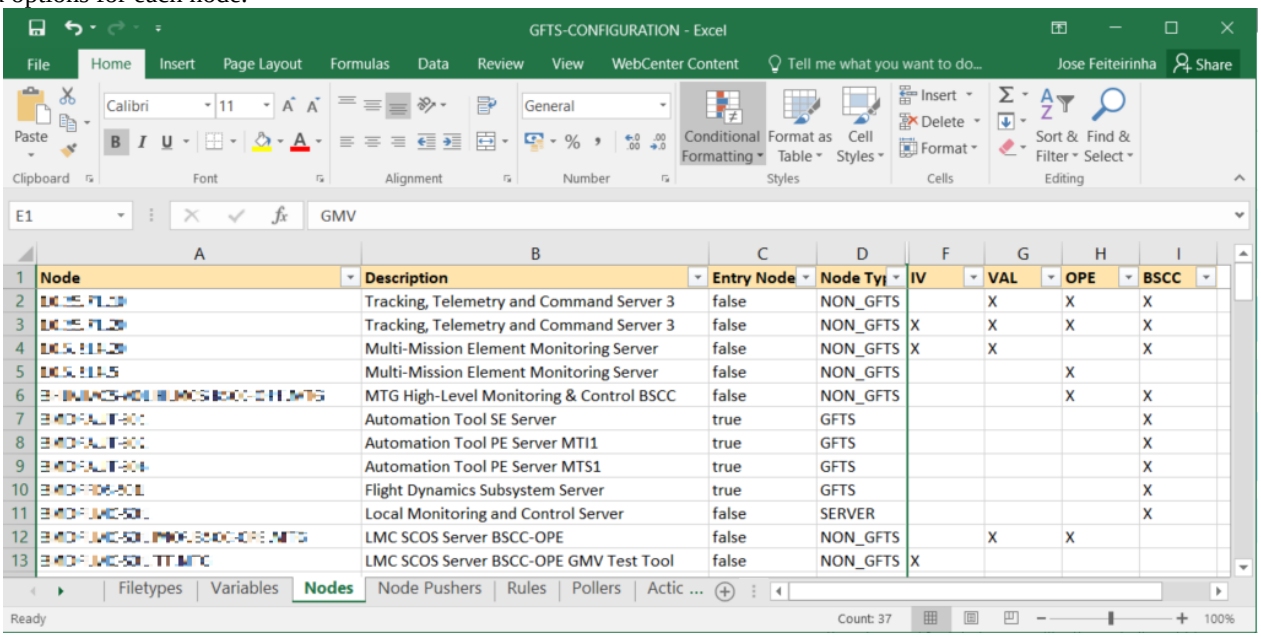
The information is organized in different worksheets. The file types sheet contains information about the different file types that are supported by the GFTS system. This includes the file type identifier, the file type description, and the associated environments where the file type is enabled.



	Filetype	Description	IV	VAL	OPE	BSCC
2	--OSEID--	OSE Template Filetype	X	X	X	
3	AEM	Satellite Attitude	X	X	X	X
4	ANTIVIR	Antivirus Signatures		X	X	
5	AUDIT	CI audit information		X	X	
6	CALTM	Calibrated Telemetry from MCS	X	X	X	X
7	CI	Configuration Item		X	X	
8	CONTXT	iFDContextData	X	X	X	X
9	CTX	Operational Context	X	X	X	X
10	ELOGCFG	ELOG Configuration		X	X	
11	EVENT	Satellite or MOF event		X	X	
12	EXTED-FIN2	External Flight Dynamics Data	X	X	X	X
13	EXTED-FINA	External Flight Dynamics Data	X	X	X	X
14	EXTED-RSGA	External Flight Dynamics Data	X	X	X	X
15	EXTED-TAIU	External Flight Dynamics Data	X	X	X	X
16	EXTED-TEC	External Flight Dynamics Data	X	X	X	X

Figure 5 - Screenshot of the Filetypes worksheet

The nodes sheet contains information about the different nodes or machines that are used in the GFTS system. This includes the node name, the hostname or IP address, the username and credentials, and the associated settings and options for each node.



	Node	Description	Entry Node	Node Ty	IV	VAL	OPE	BSCC
2	[redacted]	Tracking, Telemetry and Command Server 3	false	NON_GFTS		X	X	X
3	[redacted]	Tracking, Telemetry and Command Server 3	false	NON_GFTS	X	X	X	X
4	[redacted]	Multi-Mission Element Monitoring Server	false	NON_GFTS	X	X		X
5	[redacted]	Multi-Mission Element Monitoring Server	false	NON_GFTS			X	
6	[redacted]	MTG High-Level Monitoring & Control BSCC	false	NON_GFTS			X	X
7	[redacted]	Automation Tool SE Server	true	GFTS				X
8	[redacted]	Automation Tool PE Server MTI1	true	GFTS				X
9	[redacted]	Automation Tool PE Server MTS1	true	GFTS				X
10	[redacted]	Flight Dynamics Subsystem Server	true	GFTS				X
11	[redacted]	Local Monitoring and Control Server	false	SERVER				X
12	[redacted]	LMC SCOS Server BSCC-OPE	false	NON_GFTS		X	X	
13	[redacted]	LMC SCOS Server BSCC-OPE GMV Test Tool	false	NON_GFTS	X			

Figure 6 - Screenshot of the nodes sheet in the Excel file (node hostname/ip addresses have been blurred)

The pollers sheet contains information about the different pollers or agents that are used in the GFTS system to poll for and transfer files. This includes the fileset, alias, and environment for each poller, as well as optional information such as the enabled status, host machine, username and credentials, polled folder, file mask, file type, and destination machines for the transferred files.

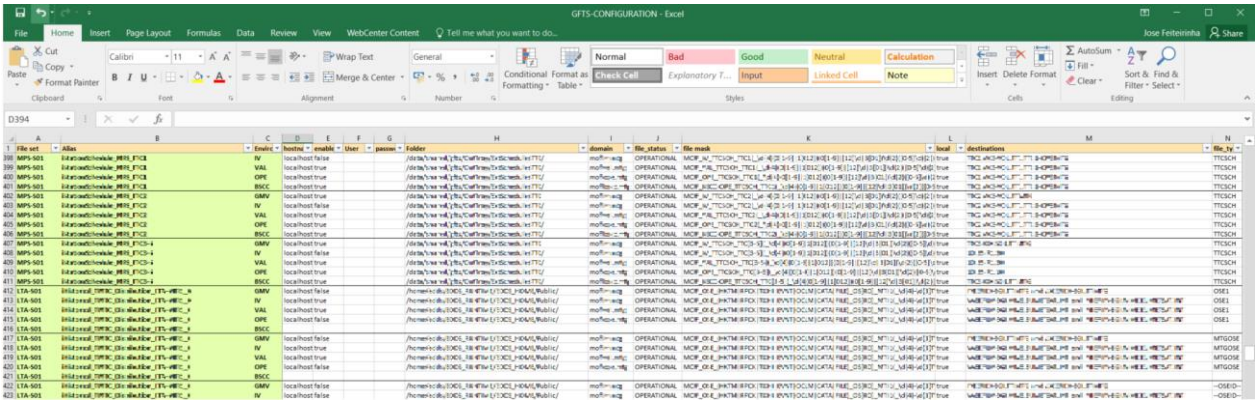


Figure 7 - Screenshot of the pollers sheet. Network information has been intentionally blurred.

Each poller is represented by multiple rows in the pollers sheet, with each row representing the configuration for that poller in a specific environment. The pollers are initially sorted by fileset and then by alias, and a strong line is added between each poller to make it easy to see the different pollers and their configurations.

In addition to the main sheets for file types, nodes, and pollers, the GFTS Excel configuration file also includes other sheets for specific settings or options that are applicable to the GFTS system as a whole. These additional sheets can provide further details and information about the configuration of the GFTS system in each environment. However – for simplicity – they are not included in this document, as the goal is to explain the changes and benefits brought in by the new processes, not to detail the singular configuration of the MTG file transfer system.

In summary, the GFTS Excel configuration file is an important tool for organizing and storing the configuration information for the GFTS system. It provides a user-friendly way to specify the settings and options for the GFTS system in each environment, and allows for easy access and modification of the configuration as needed.

4.3.2. Version control

The Git repository for the GFTS system contains the configuration files and CI/CD pipelines used to manage and deploy the system. The repository uses branches to manage different versions of the configuration files, with the main branch containing the latest, stable version used for deployment to various environments. Tagging is used to track when specific versions of the configuration were deployed.

The repository contains the GFTS Excel configuration file, as well as a folder with the generated XML configuration files. The Excel file is the primary source of configuration information for the GFTS system, and is used to specify the settings and options for the system in each environment. The XML configuration files are generated based on the information in the Excel file, and are used for deployment to the different environments. The Git repository also contains the CI/CD pipelines that are used to automate the process of generating and deploying the XML configuration files. These pipelines ensure that the configuration files are generated consistently and accurately, and allow for efficient and reliable deployment of the updated configuration to the various environments.

It uses branches to manage different versions of the configuration files. The main branch contains the latest, stable version of the configuration, which is used for deployment to the validation, operational, and backup environments. When a deployment is made to one of these environments, a tag is created on the Git repository for the commit that generated that version of the configuration. This allows for tracking of when specific versions of the configuration were deployed. Other branches can be used for development and testing, and changes made in these branches can be merged into the main branch once they are ready for deployment. This allows for collaboration and coordination among team members working on the GFTS system. The Git repository and its branches are an important part of the process for managing and deploying the configuration of the GFTS system. They provide a way to store and manage the configuration files, and enable efficient and reliable deployment of the updated configuration to the different environments.

4.3.3. DevOps Pipelines

The GFTS Excel configuration file is used to store and organize the configuration information for the Generic File Transfer System (GFTS), a system for transferring files between different machines or environments. The Excel file is organized into sheets or tabs for different aspects of the GFTS configuration, such as file types, nodes, and pollers. It contains the configuration information for all environments, and is stored in a GIT repository.

CI/CD pipelines are used to validate the Excel file, compare it with previous versions, generate the XML configuration files, keep the XML files under version control, and generate the documentation for the GFTS system. Additionally, there is a separate pipeline that can be manually triggered to re-generate the Excel file from the XML configuration.

The Git repository contains the GFTS Excel configuration file, as well as a folder with the generated XML configurations. When changes are made to the Excel file and committed to the repository, the CI/CD pipeline generates XML configuration files based on the updated Excel file. These generated XML files are then stored in the repository as artefacts, with each file being renamed to include the branch name and commit ID. These XML files are then used for deployment to the various environments.

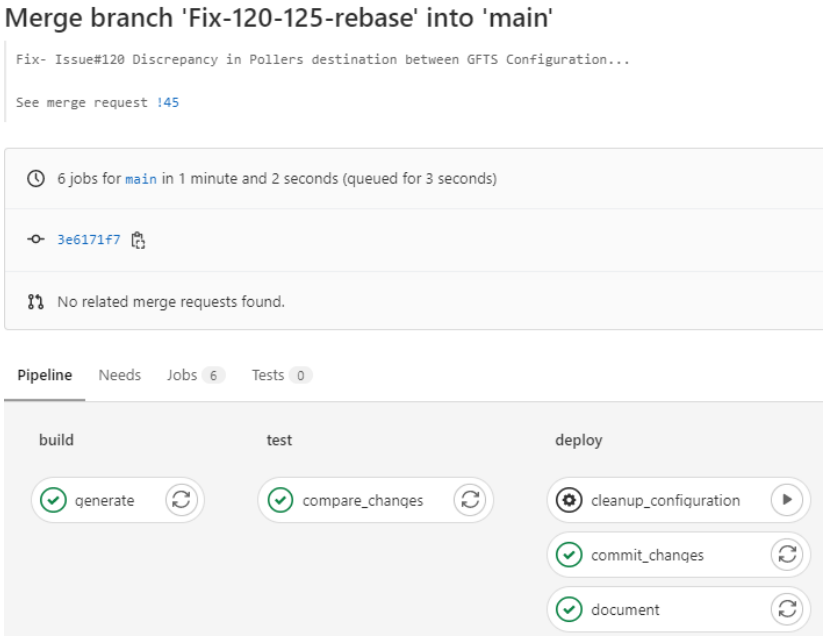


Figure 8 - Screenshot from a successfully executed pipeline triggered by a commit to the Excel file

The Git repository is used to store and manage the GFTS Excel configuration file and the generated XML configuration files. It allows for version control of the configuration files, and enables collaboration and coordination among team members working on the GFTS system. The CI/CD pipelines are used to automate the process of validating, generating, documenting and deploying the XML configuration files for the GFTS system. They ensure that the configuration files are generated consistently and accurately, and allow for efficient and reliable deployment of the updated configuration to the various environments.

4.3.4. Generated documentation (Wiki)

We use CI/CD pipelines to generate a full GitLab Wiki. Each commit to the Git repository triggers the pipeline, which updates the Wiki with the latest information. The Wiki includes a list of file transfers, along with diagrams that make it easy to understand and navigate the information. The Excel file in the Git repository contains the list of files transferred through the MTG ground segment. This information is used to generate the Wiki page, although some sensitive information such as user credentials is excluded. The Excel file also includes other information such as folder paths and file masks, but this is not included in the Wiki page.

The diagrams on the Wiki page are generated using PlantUML, a tool that uses simple text language to create UML diagrams. The text is written using a special syntax that describes the diagram, and then it is fed to the PlantUML program to generate the image. This makes it easy to create complex diagrams quickly and easily.

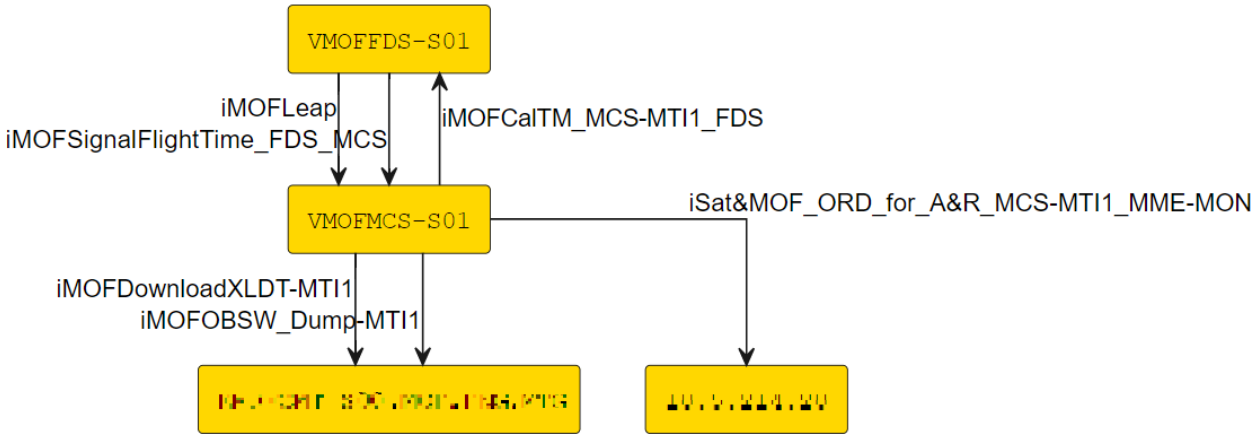


Figure 9 - Example of automatically generated diagram of file transfers from/to an MCS machine. Some information has been intentionally blurred.

The Wiki page generated by the CI/CD pipeline provides a user-friendly way to view and understand the information contained in the Excel file. It allows users to easily navigate the different aspects of the file transfer process using hyperlinks, and the diagrams provide a clear visual representation of the information. Overall, the use of CI/CD pipelines to generate a Wiki page is a valuable tool for managing the file transfer process.

4.3.5. CI/CD & Validation Tools Repository

All the Excel and XML manipulation and validation code has been placed in a separate repository from the configuration repository. This was so that Excel configuration repository is as simple as possible.

The code itself was written in Python, since it has a large set of available libraries and Python developers are common. The goal when selecting the language was to facilitate maintainability since, albeit it hasn't seen frequent changes, it must be maintained (e.g. if a bug is found or a new GFTS changes the configuration schema). This also enables the usage of Gitlab issues to track user requests, developer tasks and software bugs.

The Excel configuration repository contains only one file, the CI/CD configuration file, which calls the validations tools repository.

4.4. Impact

We presented a specific scenario where DevOps was used to create a new process for managing incoming configuration changes of GFTS file transfers. Previously, the process of managing incoming configuration patches was largely manual and involved connecting to multiple machines to compare lists of files and configurations using a GUI that was not intended for this purpose. This process was inefficient and prone to error. In order to improve the process, we decided to implement a new solution that would allow us to easily change and view multiple configurations in a user-friendly way. We chose to use DevOps principles and practices in order to automate the process and enable tracking of changes and issues.

The new process, which uses DevOps principles and practices, has resulted in significant benefits77. Changes are now easier to track, faster to deploy across all environments, simpler to troubleshoot, and are automatically saved under version control. Additionally, no specific tools need to be installed on the user's computer, as Gitlab runners automatically handle the work.

Although the training of the team to use the Excel file went smoothly, the usage of DevOps generated artefacts, checking the new validation results has required thorough training and repetition until enough team members were comfortable and capable of performing the tasks and teaching newcomers. Knowledge of tools such as Git is not yet a given for all engineers, even when using the GUI.

The goal of this project was not to use DevOps, but the implementation of the new process gradually took on many of the characteristics of DevOps. This has resulted in a more efficient and reliable process for managing incoming configuration changes of GFTS file transfers. However, it is important to note that using DevOps in large, critical missions like satellite on-board software or mission control systems may still carry some risks and

challenges. It is important for organizations to carefully consider the potential risks and challenges and determine if the benefits of using DevOps outweigh them in their specific scenario.

5. Discussion

DevOps is a proven solution, used commonly across multiple industries. However, it is particularly hard to advocate in a scenario such as the space domain where existing processes are seemingly incompatible with Agile methodologies. Although we have proven this not to be necessarily true - by keeping the necessary reviews - there is still the issue of the tangibility of the benefits provided by DevOps. DevOps can help organizations to become more agile and responsive to changing business needs by streamlining software development and deployment processes. It enables teams to quickly and easily make changes to their systems. However, these are not necessarily requirements to large missions that take years to plan and execute. Again, the benefits of increased agility are intangible. They are difficult to measure and quantify, making it hard for organizations to see the value of implementing DevOps. We've also seen that DevOps has been mostly ignored by international space standard organizations, particularly ECSS has a full book on Agile software development, but the full body of ECSS fails to mention DevOps one single time.

5.1. Intangible benefits of DevOps

The main reason why many of the DevOps benefits are intangible is because they are related to improved efficiency, quality, and communication. These are not easy to measure and quantify, but they can be felt in the overall improvement of the organization. For example, DevOps can help create a better culture of collaboration and trust between teams, which can lead to greater productivity, innovation and satisfaction. Additionally, automated processes and continuous integration can help reduce time to delivery, resulting in faster time to the end user. Finally, DevOps can help improve the quality of products by automating testing and reducing human error. Some more relevant, yet intangible benefits are shown in the sections below.

5.1.1. Increased collaboration

DevOps enables teams to move away from silos and work more closely together. This enhances collaboration, enabling teams to work better together to produce better results. These are hard to measure, but having lower bottlenecks facilitates dialog. If an issue is requested and ready for testing in 24h, the user will be able to quickly check and provide feedback. If the fix has issues, he will be able to quickly notify the developers and thus get a fast response.

If the user would have to wait months for the fix to arrive, and then receive an incomplete fix, it's is only natural that the level of satisfaction would be lower, independently of the quality of the development team.

DevOps helps establish a shared context across different teams, which helps provide a better shared understanding of the problem at hand.

5.1.2. Context awareness and Recall

Analog to the previous point, a developer may receive a list of issues to fix, does the fixes and then goes on to do other work for 6 months. If he then receives notice that the fix was incomplete, he would have to recall his line of thought 6 months prior. This is an intangible loss of time, and made worse if the issue is - as often the case - to a different developer.

With DevOps, the user/developer feedback loop is shortened, and the context awareness is better. The developer does not need to recall so much from the past, as he is more actively involved in the development process. This leads to faster results, and money saved. Often also to better results, since more feedback can be quickly provided, and a shared context exists.

5.1.3. Improved agility

DevOps enables teams to move quickly and efficiently. With automated processes and improved collaboration, teams can move from idea to production faster than ever before. This enables teams to respond to contingencies and other urgencies quickly, efficiently and as a team. This increases team satisfaction, as a sense of achievement and team spirit is created.

5.2. DevOps and Quality Gates: Not Mutually Exclusive

When it comes to software development, many project management strategies have been developed to maximize efficiency and ensure high quality releases. Two of the relevant strategies for this article are DevOps/Agile and the typical quality gates / reviews of ECSS.

While many people might assume these two strategies are mutually exclusive, in reality, they can be quite compatible. The core idea of DevOps/Agile is to build, test, and release software quickly, in small batches. This allows for rapid feedback, which is essential for ensuring the highest quality software possible. At the same time, ECSS quality gates focus on the traditional approach of taking larger steps, with more formal reviews of each step.

How can these two approaches be reconciled? The answer is: by allowing users and validators to quickly check on the work in progress (WIP) while it is being done. This allows them to troubleshoot directly with the developer as the work is being done, rather than waiting for a more formal review. In other words, when the review comes, there will already have been some troubleshooting done, resulting in less to discuss. Now, this is not to say that DevOps/Agile should be required in place of the ECSS reviews. Instead, it should be seen as an optional approach to ensure the highest quality releases. Users and validators still have the option of ignoring DevOps/Agile and waiting for the formal review. However, by allowing for quick checks of the WIP, we can make the review process much smoother and more efficient, as well as ensuring higher quality releases. In conclusion, DevOps/Agile and the ECSS reviews do not have to be mutually exclusive. By allowing users and validators to quickly check on the WIP while it is being done, they can take advantage of the benefits of both approaches, resulting in more efficient and higher quality releases.

6. Conclusion

In this paper we presented the technical, logistical and processual challenges that were overcome by the use of the Gitlab CI/CD suite to create an automated process for synchronizing the Generic File Transfer System configuration across multiple environments. We detailed the impact of this DevOps based solution in supporting the Meteosat Third Generation launch and operational readiness.

In recent years, the use of DevOps principles and practices has become increasingly common in the software development industry, particularly in the context of Agile methodologies. However, the ECSS (European Cooperation for Space Standardization) framework for managing space projects, which is commonly used in the space industry, does not explicitly mention DevOps. This may lead some to assume that DevOps is not compatible with the ECSS framework, or that it is not relevant for space projects.

In fact, DevOps is highly complementary with Agile methodologies, and can be a valuable tool for improving the efficiency and effectiveness of space projects. This is demonstrated by our own experience, where we implemented a DevOps-inspired solution to improve the process for managing incoming configuration changes of GFTS file transfers. The new process, which uses DevOps principles and practices, has resulted in significant benefits, including faster deployment across all environments, easier tracking of changes and issues, and automatic saving of configurations under version control. Importantly, there was virtually no added effort in training the team to use the new approach and in fact they saved time, and it did not require any change to the planned ECSS reviews of the systems.

The ECSS framework does provide a set of principles and practices that can be applied using a variety of methods. This means that the omission of DevOps does not imply its exclusion. In fact, we show it can be easily integrated into the ECSS framework, and can help to improve the efficiency and effectiveness of space projects. A future ECSS revision process could take into account the input of industry practitioners.

In conclusion, while DevOps may not be explicitly mentioned in the ECSS framework, it is highly compatible with Agile methodologies and can be a valuable tool for improving the efficiency and effectiveness of space projects. Our own experience demonstrates the benefits of using DevOps principles and practices in the context of space (ground) project management, including the ease of training the team, the time savings and overall improved satisfaction that it can bring. Similar to the Waterfall Model, DevOps is compatible with ECSS project management standards, and both can be used complementary.

References

List of references

[1] Lucy Ellen Lwakatare, Pasi Kuvaja & Markku Oivo. “Relationship of DevOps to Agile, Lean and Continuous Deployment“

[2] Mohammad, Sikender Mohsienuddin, “DevOps Automation and Agile Methodology (August 3, 2017). International Journal of Creative Research Thoughts (IJCRT)”, ISSN:2320-2882, Volume.5, Issue 3, pp.946-949, August-2017, Available at SSRN: https://ssrn.com/abstract=3655581

[3] Van Casteren, Wilfred. "The Waterfall Model and the Agile Methodologies: A comparison by project characteristics." Research Gate 2 (2017): 1-6.

[4] Morgan B. Kamuto, et al. "Factors inhibiting the adoption of DevOps in large organisations: South African context", 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017, https://ieeexplore.ieee.org/document/8256556

[5] Calzolari, Gian Paolo, et al. "The ESOC Approach for World Wide File Transfer." SpaceOps 2008 Conference. 2008.

[6] Ahmad, Ehsan, et al. "ECSS standard compliant agile software development: an industrial case study." Proceedings of the 2010 national software engineering conference. 2010.

[7] European Cooperation for Space Standardization, ECSS Secretariat, ESA ESTEC, P.O. Box 299, 2200 AG Noordwijk, The Netherlands, ECSS-E-ST-40C Space Engineering—Software, March 2009.

[8] European Cooperation for Space Standardization, ECSS Secretariat, ESA ESTEC, P.O. Box 299, 2200 AG Noordwijk, The Netherlands, ECSS-Q-ST-80C Product assurance—Software product assurance, March 2009.