GNC Flight Software Development and Verification
A process, its evolution, and the challenges involved.

Leonardo B. Farconi[1], V. Gabriel Moyano[2], Tamara G. Rojo[2], Jörg Brauer[3], Christof Efkemann[3], Jose L. Redondo Gutierrez[1], Pascal Pieper[1], Erin C. Cold[1], Jan Sommer[2], Felix C. Passenberg[1], Janosch Reinking[1], Andrea Monti[2]

The GNC Department at the Institute of Space Systems has developed the capability to implement GNC algorithms into complete flight-worthy systems. A custom development and verification (D&V) process was created, trying to tailor and combine the DLR Software Engineering Guideline and RTCA DO-178C Level DAL-C standard. This iterative process embeds verification within development, ensuring that core processes are executed and minimum required artifacts are produced for software quality assurance. For the process to run smoothly, specific Roles are defined based on the expected activities to be performed by each role. These activities are either defined in general terms or based on Quality Gates (QGs), which in

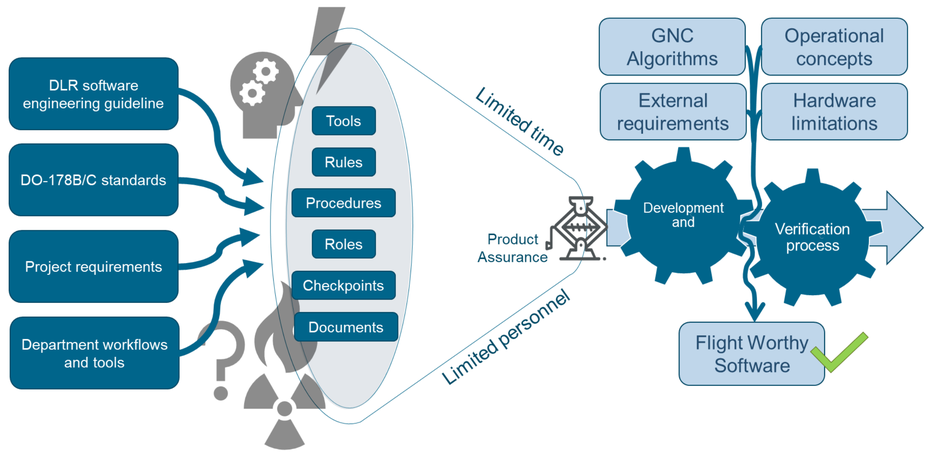

Figure 1 – The problem to be solved. Source: Created by the author.

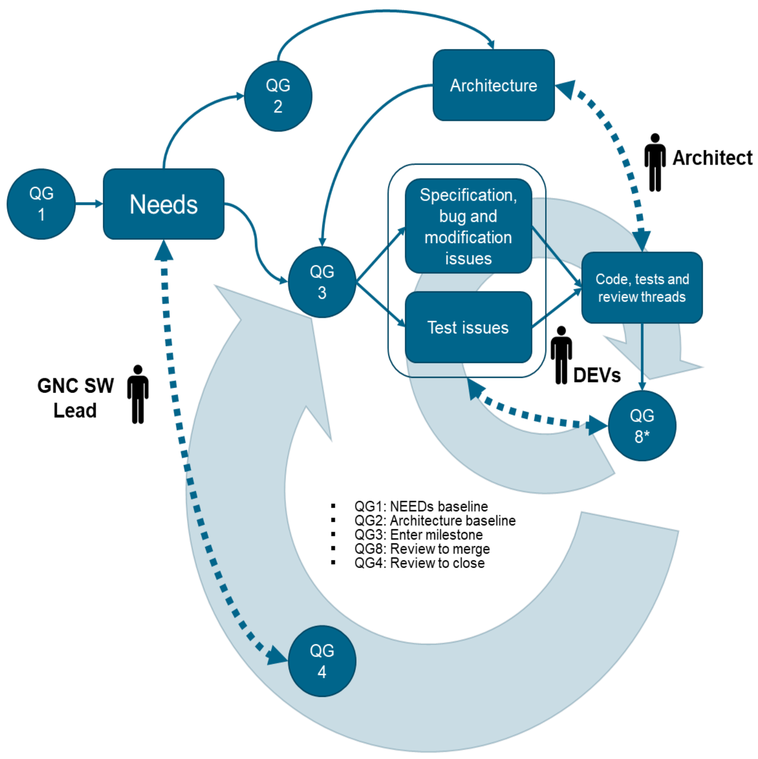

Figure 2 – Our solution so far. Source: Created by the author.

turn provide a framework for ensuring that the software development process is rigorous enough but still effective. Figure 1 summarizes the different inputs and outputs of this continuous tailoring process, while Figure 2 provides an overview of the D&V process, with its roles and quality gates. While some aspects of the DLR guideline or DO standard cannot be automated, they rely on team

adherence to established strategies outlined in Basic Policies. These policies include guidelines such as "No self-approval", "No tests allowed to fail", and procedures for managing code coverage suppression. Given this, The D&V Process can then be presented in the form of flowcharts that relate the Roles to the Quality Gates and other specific activities in time, as presented in Figure 3. The verification aspects of the process require additional detail, outlined in an internal guideline that explains testing levels and tools and how they're defined and used based on specification level. Additionally, specification and documentation artifacts from the standard/guideline are created as Gitlab issues following specific templates and linking strategies that enable automatic tracking and traceability of software development work, as presented in Figure 4.

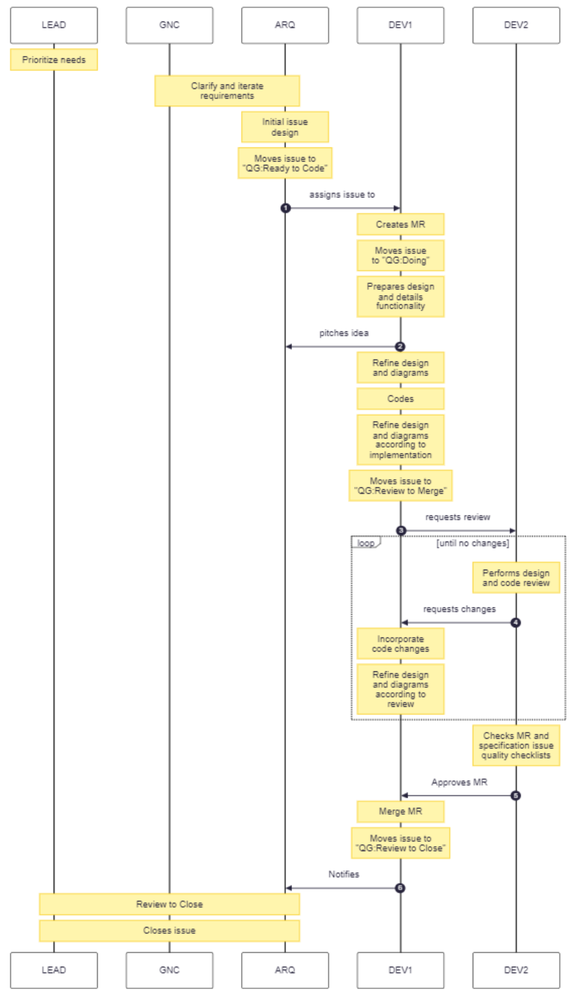

Figure 3 – The issue development sub-process. Source: Created by the author.

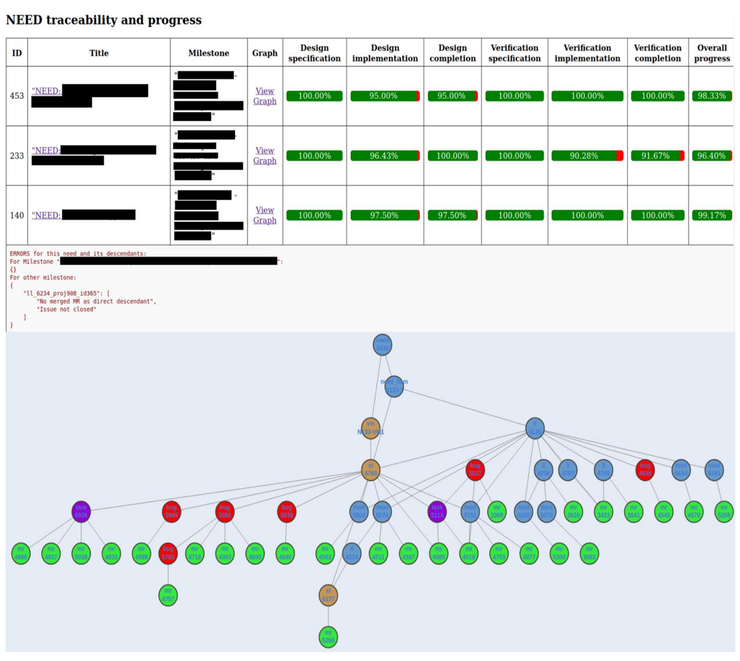

Figure 4 – The traceability tool. Source: Created by the author.

Past challenges	Current solutions	Current challenges	Possible solutions
Absence of a development or verification process, relying on unwritten rules, personal memory, and personal taste.	Established process with clear roles, actions and frameworks. Virtual “Suggestion box” and recurrent “Idea rounds” allow all participants to propose changes.	Difficulty of motivating team to follow a relatively complex process (compared to no process).	?
Too many quality gates would focus too much on design before implementing, or create unnecessary bottlenecks.	Only “review to merge”, with the requirement to keep the architecture and issues updated. Allows prototyping solutions.	Checking architecture and issues design description consistency takes time.	Automated consistency check with use of LLMs?
Too many roles (Product assurance support), created bottlenecks for merging.	DEVs have the highest and most distributed responsibility for PA.	PA responsibility slows DEVs down and reduces interest. Quality checklists are tedious but hard to automate.	Use of LLMs for automated quality checks? Should DEVs be focused on development only and not take care of PA tasks?
Concentration of responsibilities and knowledge on the Lead led to bottlenecks.	Architect role helps disseminate and keep consistency of architecture decisions.	Code modifications either introduce overhead in traceability or are avoided due to the overhead. Participation in architecture choices still reduced.	?
Architecture specification too complex, causing concentration of knowledge and difficulty in understanding.	Adoption of C4 diagrams, different levels of abstraction, and Clean Architecture concepts.	?	?
No traceability to NEEDs created lack of context and left features not implemented or not tested.	Strong focus on traceability in the specification templates gives direct context to persons involved.	Keeping the traceability between all issues consistent takes time.	?
Manual testing consumed too much time, lead to human error, lack of repeatability, or late access to testing in the target.	Automated testing with the target computer with queue for running CI/CD at every MR to stable branch.	Pipeline takes too long and one resource blocks parallel MRs. Choosing which tests to NOT run for every MR is not trivial.	Create parallel resources?

Figure 5 – Challenges and solutions. Source: Created by the author.

This process has faced many different iterations along its use throughout the years and projects. The main struggle has always been balancing between the rigor of each quality gate or policy, and the speed and flexibility required to not block the developers or reduce their momentum. Figure 5 summarizes the main challenges faced, past and current, the solutions found and the lessons learned.

[1]Institute of Space Systems, German Aerospace Center, Bremen
[2]Institute of Software Technology, German Aerospace Center, Braunschweig
[3]Verified Systems International GmbH

DLR