Flight Software Delivery Review
S. D'Amico, T. Guffanti, and S. Lowe, 2/20/2024

Background

The Stanford Space Rendezvous Lab (SLAB)'s Guidance, Navigation and Control (GNC) flight software stacks for the VISORS and SWARM-EX distributed space systems are approaching a key delivery milestone in mid-2024. This delivery milestone is important because the spacecraft will be shipped respectively from Georgia Tech and from CU Boulder, where they are currently being integrated and tested, to the provider of launch services. From that point onwards the opportunity of updates of on-board flight software will decrease substantially, reaching then the critical path of space operations in a few months around the end of 2024.

Strategy and Timeline

The objective of this document is to provide a succinct description of the review of SLAB’s flight software before such delivery milestones occur. Given that VISORS and SWARM-EX are academic low-cost projects largely conducted by students, the flight software review will be internal and as lean as possible. This is done by targeting critical functionalities and safety aspects and by focusing on materials directly linked to the review scope only. The review team will be composed of members of SLAB.

The review will occur in three steps:

1. **Presentations and walkthroughs** by developing team about two months before flight software delivery.
2. **Code inspection and usage** by review team lasting no more than one month from (1).
3. **Closure meeting** to summarize the review outcomes, corrective actions, and recommendations.

This timeline will provide the GNC software development teams with about 1 month to respond to the review feedback comments. The following sections provide a description of specific review objectives and a description of the review steps outlined above.

Review Objectives

The specific objectives of the review are listed below in four categories:

1. **Integrated architecture and functionality**: have/has GNC...
   a. ...modes of operations been identified, including objectives and execution logic?
   b. ...modes been functionally tested to meet the objectives above?
   c. ...interfaces been properly specified, including internal and external functions/avionics?
   d. ...interfaces been tested in compliance with above?
   e. ...modes and interfaces been tested on the spacecraft software and hardware?
   f. ...modes and interfaces been finalized considering ground operations?

   *Note*: modes should be as complete as possible to support various phases of mission operations, e.g. navigation in solo (i.e., only absolute orbit determination by 1 spacecraft), control computed but not executed (i.e., control functions are computed and outputs stored in telemetry, but maneuvers are not sent to the propulsion system for execution), etc.
2. **Performance compliance (algorithms and code):** for each mode, have/has GNC...
   a. ...performance requirements been identified, including main functions and overall system?
   b. ...performance been tested, including main functions and overall system?
   c. ...performance been tested on the spacecraft hardware, including memory and compute?
   d. ...performance limits been stressed and identified?
   e. ...performance robustness and limits have been analyzed in face of main uncertainty sources (dynamics, actuation, navigation, spacecraft properties, etc.)?

   *Note:* here performance does not refer only to navigation and control, but also to code in terms of memory allocation and compute during max path tests.

3. **Safety and risk mitigation (algorithms and code):** have/has GNC...
   a. ...safety requirements been identified, including nominal and contingency operations?
   b. ...safety features been implemented and tested to meet the requirements above?
   c. ...inputs and outputs been checked for range violation?
   d. ...redundancy within and across all spacecraft been exploited to the maximum extent?
   e. ...failure scenarios been identified and tested to meet requirements above?
   f. ...risk mitigation strategy tested on the spacecraft hardware?
   g. ...safety limits been stressed and identified?
   h. ...safety robustness and limits have been analyzed in face of main uncertainty sources (dynamics, actuation, navigation, spacecraft properties, etc.)?

   *Note:* we know that there are irreducible risks, but we should also know what they are and how the system can get accidentally into that critical path. How can we minimize that likelihood?

4. **Software readability, maintainability, efficiency, compatibility:** are/have code functions...
   a. ...self-descriptive through proper choice of variable names, functions, clear logic?
   b. ...commented properly to explain complex logic, non-obvious code choices, why code does something (not restating what it does to avoid redundancy)?
   c. ...headers to describe objective and I/O interfaces?
   d. ...coded in a consistent manner w.r.t. style, naming conventions, memory allocation, etc.?
   e. ...a mechanism to track changes and version numbers?
   f. ...minimal redundancy and overlap between individual functions?
   g. ...compatible on-board compiler and operating system?

   *Note:* harmonization is key, make sure that code style is consistent throughout and that functions are not repeated, e.g. numerical integration, force modeling, time functions.

**Presentations and walkthroughs**

The first step of the review is a block of presentations and code walkthroughs performed by the flight software development team members. The objective of these presentations is to present material that can help and guide the reviewers as efficiently and clearly as possible towards assessing the specific objectives above. It is left to the development team to structure the presentations appropriately; however, it is highly recommended that:

1. The flight software is revised, and analysis conducted in the time preceding the review having in mind the review objectives above.
2. The individual presentations address the review objectives explicitly.
3. The discussion proceeds from higher to lower levels of description gradually.
4. The presentations use different “languages” or “means of expression” to convey information as needed, e.g. including graphics and tables for results and trades; block schemes for architecture, state machines, and logic; bullet lists for discussion and features; excerpts of commented code for walkthroughs, etc.
5. Each team member focuses on his own work and field of expertise.
6. Sufficient time is left for Q&A after each presentation.

At the end of the presentations, the review team should have enough information to start answering all the questions posed by the specific review objectives nominally, i.e. under the assumption that the material describes the software accurately.

**Code inspection and usage**

The second step of the review involves both the review and development teams. The review team learns the preliminaries from the presentations and walkthroughs and conducts a thorough inspection of the code. Ideally some code test cases are executed to understand its usage and output. This step might require dedicated 1:1 meetings with the development team. Various tools can be used to facilitate the code review, including static analysis tools and modern AI chatbots. Upon review, the reviewer adds review comments to a shared Google sheet or directly to Gitlab as an issue monitored by the development team.