Towards a Method for the Construction of Robust, Compiled Autonomous Spacecraft Executives

Mark S. Boddy   Steven A. Harp   Kyle S. Nelson[*]
{ mark.boddy — steven.harp — kyle.nelson }@adventiumlabs.org
Adventium Labs, Minneapolis

1 Introduction

This paper summarizes the results of an investigation, funded by NASA's Intelligent Systems program, into methods and means for using compiled automation to speed the development and deployment of increasingly autonomous spacecraft. The techniques most readily adaptable to implementing a high-level "autonomous executive" are still close to research, hard to validate in any rigorous sense, and new enough to have little in the way of a track record on previous missions. The fact that it may be much easier to verify that an artifact has certain desired formal properties than it is to generate an object with those properties has been common knowledge for a long time, and is a significant advantage for compilation approaches.

2 Compiled Automation

Compilation as an abstract concept denotes the transformation from one form of specification, usually an abstract one, to another form that is suitable for operational use. For space applications, on-line operations may be light-hours distant from the compiler. We do not, however, equate "off-line" with terrestrial computing, and in some cases it may be sensible for the spacecraft to bring the compiler along. Among the benefits of compilation for a wide variety of types of inference are predictability, specialization to a particular context, modularization, and improvements in speed and space required.

3 Lessons from Galileo

Galileo was a highly successful mission that explored the Jovian system (Harland 2000). It was also a mission that faced numerous technical challenges. The most widely known of those was the failure of the High Gain Antenna (HGA) to deploy. Two other significant challenges included reconfiguring the spacecraft to use the low gain antenna (LGA) as the primary configuration channel, and the sheer quantity of effort involved in planning close encounters on each orbital tour. On each pass, Galileo was given a sequence of commands that detailed what it was supposed to do and when it should do it during a given encounter. The Sequence Integration team, consisting of 25 engineers, started with the activities necessary to keep the spacecraft healthy and then worked on the activities necessary to meet the science objectives. Each sequence took about 2 months to prepare, matching the period of Galileo's orbit. Once created, this sequence was tested, then uploaded to the spacecraft. This is exactly the process of compilation as we have described it, but using a time-intensive, highly manual process. The HGA deployment efforts and LGA reconfiguration similarly involved ground-based analysis that was reduced to code, tested and verified, and uploaded to the spacecraft.

4 Scope of Study

The major functional requirements of spacecraft determine the sorts of functions, or combinations of functions, for which we are interested in finding methods of compiled automation. The functions of interest are indicated by the shaded region in Figure 1. This shaded region sits on top of spacecraft systems and functions that are already substantially automated (and in many cases compiled), but is still close enough to the hardware to require guarantees on correctness, coverage, and timely behavior.

[*]Supported by NASA Grant NAG-2-1634

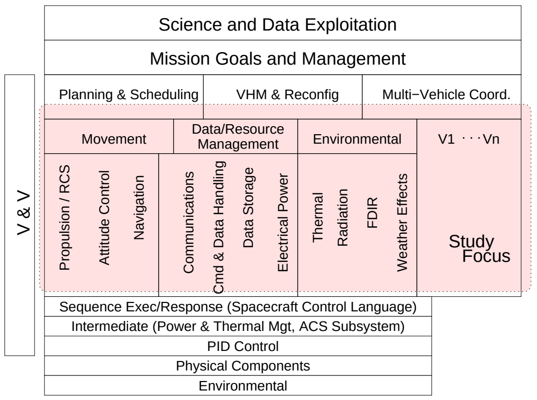

Figure 1: Functional mission architecture

5 Case Study: Jupiter Icy Moons Orbiter

The Jupiter Icy Moons Orbiter (JIMO) is a proposed mission to orbit three planet-sized moons of Jupiter (Callisto, Ganymede and Europa) (NASA JPL). The round trip light time (RTLT) to Jupiter and back is nearly two hours. Hence, there cannot be any expectation of fine-grained control of JIMO from Earth. The expected data rates and data volume generated by the science instruments together with the operational complexity of the orbital passes will make business-as-usual mission operations (e.g., coding detailed timing instructions and uploading to the spacecraft as was done on Galileo) both extremely difficult and detrimental to mission objectives. Consequently, JIMO will require as much automation as possible to handle day to day tasks (e.g., attitude control, backup power maintenance, etc.). These tasks, however, are also mission critical and although automation is an attractive option, that automation must be verified to the level of other flight software.

According to current mission timelines, the time en route to Jupiter will be approximately two years longer than the time remaining before launch of the spacecraft. This time can be exploited by making a deliberate decision to split the JIMO development into two distinct phases. The first phase, covering the period up to launch, would concentrate on the hardware components of the spacecraft and those software functions required for the launch and cruise phase of the mission. The second phase, covering the period after launch up to arrival at Jupiter, would focus on the additional software functions required for use at Jupiter, for example software that processes the science data. Compiled automation can satisfy this approach through the generation of new software as an artifact, or a set of artifacts, that can be independently verified on the ground prior to being uploaded to the spacecraft.

6 Summary

Moving from an opportunistic and *ad hoc* approach to compiled automation to an approach that is principled, predictable, and verifiable would be a huge win, whether or not the end result was to put the underlying reasoning processes on-board. The techniques required are known, the methods of implementation are consistent with current architectures, and their introduction can be gradual so as to reduce risk and ensure an incremental rather than a radical transition in operational practices. Considering the automation of individual functions, rather than “compiled autonomy” as an indivisible whole is driven from the underlying technology: it is difficult to talk about how to compile something without being fairly precise about what the “thing” is (in this case an algorithm, or a form of inference).

This natural partitioning of functions is also very appropriate, considering the current status of spacecraft autonomy. At the current state of the art, it is difficult to conceive of building an on-board, automated reasoner that could be counted on to respond appropriately to the failures dealt with during the Galileo mission, making the full scope of operational and configuration changes required.

Quite a bit more detail on this project is available in the Final Report (Boddy, Harp, & Nelson).

References

Boddy, M.; Harp, S.; and Nelson, K. Clockwork: Requirements definition and technology evaluation for robust, compiled autonomous spacecraft executives. www.adventiumlabs.org/publications.htm.

Harland, D. M. 2000. *Jupiter Odyssey.* Springer-Praxis. About the Galileo mission.

NASA JPL. Jupiter icy moons orbiter home page. http://www.jpl.nasa.gov/jimo/.