Constellation Program Return to the Moon: Software Systems Challenges – Autonomy and Autonomicity a Solution?

Dr. David J. Atkinson
Program Manager, JPL Exploration Systems Engineering
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr., Pasadena, California USA 911089
David.J.Atkinson@jpl.nasa.gov

Abstract

This letter, based on a keynote talk at EASe-2006 introduced NASA's Constellation Program, which is developing new space systems for renewed human exploration of the moon, and eventually, Mars. A selection of challenges for software systems were introduced that arise from the special circumstances of Constellation Systems. These challenges illustrate a variety of the types of problems that must be addressed related to software quality, automation, autonomicity and autonomy. For example, Constellation program-level systems engineering and integration activities are tasked with ensuring interoperability, reuse, compatibility, and evolutionary upgrade of all systems. To further compound the challenges, Constellation missions represent a mixing of the human space-flight processes with those of NASA's robotic exploration missions. These factors and others give rise to many unique and/or significantly more complex engineering than has been previously faced in the development of space systems. In this context, software reliability and safety become critical qualities for what will arguably be the most complex software systems artifact ever created.

1. Introduction

The realities of today's missions; reducing costs while staying safe, and the vision for tomorrow's missions; extreme, novel, flexible and self-sustaining, dictate the need for automation, autonomy (self-direction/self-governance) and autonomicity (self-management),

The Constellation Program, managed by NASA's Exploration Systems and Mission Directorate, is charged with developing, integrating, and operating the space systems that will enable further human exploration of the moon, and eventually, Mars. Among the most critical of these systems is the Crew Exploration Vehicle (CEV), slated to replace the aging Space "Shuttle" fleet as the primary space transportation system to near-Earth space soon after 2010 and to lunar space around 2018. Other in-space systems include the Lunar Surface Access Module (LSAM), launchers, in-space propulsion, lunar surface habitat, and lunar surface systems including instrumentation and robots of various types. Supporting these integrated systems on the ground will be mission operation systems and ground data systems. Information technology, computing, software engineering and automation are technology areas that have each rapidly advanced since the last time NASA developed major space systems. There are significant challenges for software systems introduced by the unique requirements, scale, and complexity of this enterprise. Among the most challenging for software systems are those arising from automated, autonomous or autonomic systems.

2. Exploration Systems, Automation, Autonomicity and Autonomy

Although the on-going NASA procurements of some of the Constellation systems preclude a detailed discussion of system and mission requirements, it is possible to examine some of the more well know and previously disclosed scenarios involving automation, autonomicity and autonomy. Unlike many previous NASA systems, there are a great many capabilities and functions in the Constellation systems that must be automated (human-independent) or semi-automated

(human involved in the process). Autonomy and autonomicity capabilities take this one step further, indicating functions that must be able to be performed independently of Earth control and communication. In NASA parlance, the crew may participate in "autonomous" functions (i.e. definition of autonomous is without control from the Earth); similarly, some autonomous functions may be fully automated. Autonomicity is where the software or hardware is self-managing – a specialized form of fully automated autonomy.

A salient example where automated, semi-automated, autonomous and autonomic functions all come together is in the area of Integrated Systems Health Management (ISHM). Unlike robotic spacecraft, which in the case of serious error "fail safe", i.e., recovery may be guided in time by ground operations, human spaceflight systems must "fail operational", to preserve human health and safety as the number one priority. One of the more studied aspects of ISHM in the Constellation program (where significant technical issues remain unresolved) is in the area of aborts. For instance, earlier in the definition of the Crew Exploration Vehicle, there was a requirement that the vehicle be able to "abort to Earth" at any time, even if the crew is incapacitated. It is likely there will be some constraint put on this requirement due to the enormous technical challenge of some of the potential failure scenarios.

Another example where challenges arise for automated or autonomous software systems are "locus of control" situations. There may be planned or unplanned operational situations that require primary control to shift from ground to flight, crew to vehicle, or the opposite, in various combinations of automated and autonomous modes. Examples include launch, space systems rendezvous and docking, EDL (entry-descent-landing), emergency situations, and telecommunication interruption due to occultations. Studies are underway now regarding situations that require "transfer of authority" and how these may be decomposed into functional requirements on flight systems, crew, ground systems and operations.

3. Software Reliability

Software reliability is a major issue for the Constellation Program. It is certain that many millions of lines of code must be developed. The Program's goal is to avoid introduction of defects as well as detect and remove (or mitigate) defects at each stage in the software lifecycle. Software systems reliability can be affected throughout Constellation's mission lifecycle by programmatic, management, and technical factors. Any of these factors can increase the likelihood of defects or cause defects to be introduced.

3.1. Programmatic Factors

Programmatic factors include the mission architecture and operations concepts. For example, fundamental changes in requirements, lofty expectations, changes in organizations and stakeholders, and the familiar constraints on cost and schedule that inevitably take hold.

In Constellation, the mission architecture is very complex with many operations scenarios. Concepts for mission operations require that multiple alternative, contingency mission plans be available at any time to ensure safety and success of the overall mission objectives. These introduce significant complexity to the functional requirements on systems that include software. A significant programmatic challenge for software systems arises from multiple government and industry stakeholders, each with their own views, standards, and processes. This adds another level of complexity, principally to the development process.

Systems engineering and integration "expectations" for software systems also introduce reliability challenges. Reuse is potentially one of the most significant. A system engineering goal is to enable Constellation software artifacts to be reusable across systems, and across missions, over a significant lifetime (a decade or more). Added to the reuse mix is the opportunity (or threat, some would say) of reusing legacy systems. Reuse will be discussed in more detail below.

Interoperability, a factor relating to the "locus of control" scenarios described above, also introduces special challenges. Constellation systems will be implemented by multiple organizations (NASA, the Prime contractors, subcontractors). They must be readily operated by common operations techniques and technology, including a shifting locus of control. The level of interoperability sought for Constellation significantly exceeds any current civil space system.

Compatibility is another system engineering objective. The goal is to enable key components to be interchangeable among project systems (e.g., CEV, LSAM). A component of any significant complexity will have some of its functionality delivered by software.

3.2. Management Factors

There are multiple Management factors that strongly affect software reliability. These include planning, knowledge, commitment, communications, flexibility, and personnel transitions.

A major management factor we are focusing on right now is ensuring that strong planning is performed early enough in the Constellation Program. Numerous governing rules in NASA as well as industry best practices support this objective. An example product of early planning is the "Software Management Plan", a document with a rather obvious title that nevertheless can enable or disable many software engineering practices within projects and thus have a profound effect on overall software quality.

A second major factor is how Constellation program managers perceive the risks associated with software systems. As in many organizations that came of age before the maturation of serious computing power and software, senior management has its own expertise base centered in hardware engineering, operations, and other disciplines. Software remains something mysterious for many managers. Through some tough lessons-learned by NASA and other organizations, many managers recognize software's importance for delivering an increasing amount of functionality in space systems and the associated cost of poor software quality. Managing risk through rigorous analysis, planning, testing and other methods will be a key approach.

3.3. Technical Factors

Technical factors that affect software reliability are most familiar to those of us working on such systems. These include software engineering process, development environments, tools, legacy systems, and the overall product lifecycle.

4. Software Product Lines

As mentioned earlier, "reuse" is one of the key objectives of the Constellation program. The concept of reuse includes both hardware and software systems, either new, legacy, commercial, or some combination thereof. Many case studies have shown that the systematic development and reuse of core assets substantially reduces implementation time and increases product quality. Core assets include requirements, architecture, documentation, code, test plans, experience, management processes and other products of the software engineering lifecycle. The best practices for large-scale reuse have been captured and defined into a practice called software product lines.

The case studies show that it is critical that the foundations for software product lines are laid early in the project lifecycle, and that the practice receives support from senior project management. This dovetails into our observations regarding management factors that influence reliability.

One of the major foundations that must be laid early is the system architecture. It must be developed with support for reuse as a key objective. Without this focus, evolution of the software system will occur in an ad-hoc manner as upgrades are apportioned among the multiple contractors and other development organizations.

Our systems engineering team is currently focusing on identifying and mitigating the risks and obstacles to effective reuse within the Constellation Program. Concurrently, the development of a comprehensive software architecture and framework is underway.

5. Conclusion

This brief letter has described a number of very difficult challenges for software systems in the NASA Constellation Program. The challenges arise from all quarters – programmatic, management, technical, as well as the mission itself. With a higher degree of automation, autonomicity and autonomy than any other civil space systems (excluding small robotic systems), the challenges for reliably delivering safe and useful capabilities in Constellation are profound. We approach these challenges systematically, and while they are daunting, the opportunities arising from space exploration are exciting and give us a deep motivation to succeed.

6. Acknowledgement

This article is the IEEE Computer Society – Task Force on Autonomous and Autonomic Systems Apr/May 2006 – Letter released through the TFAAS Newsletter – Issue 4 @ http://tab.computer.org/aas/

The article is based on part of a keynote talk given at the 3rd IEEE International Workshop on the Engineering of Autonomic and Autonomous Systems (EASe'06), April 27, 2006, Columbia, USA. http://www.ulster.ac.uk/ease.
DOI: 10.1109/EASE.2006.4

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.