NASAs view on how to handle embedded real-time software complexity

A study from 2007 still valid

Wolfgang Spahn

2022-10-31

DrWSpahny

Final Report

NASA Study on Flight Software Complexity

Commissioned by the NASA Office of Chief Engineer
Technical Excellence Program
Adam West, Program Manager

"The demand for complex hardware/software systems has increased more rapidly than the ability to design, implement, test, and maintain them. … It is the integrating potential of software that has allowed designers to contemplate more ambitious systems encompassing a broader and more multidisciplinary scope, and it is the growth in utilization of software components that is largely responsible for the high overall complexity of many system designs."

Michael Lyu
Handbook of Software Reliability Engineering, 1996

"While technology can change quickly, getting your people to change takes a great deal longer. That is why the people-intensive job of developing software has had essentially the same problems for over 40 years. It is also why, unless you do something, the situation won't improve by itself. In fact, current trends suggest that your future products will use more software and be more complex than those of today. This means that more of your people will work on software and that their work will be harder to track and more difficult to manage. Unless you make some changes in the way your software work is done, your current problems will likely get much worse."

Watts Humphrey
Winning with Software: An Executive Strategy, 2001

Editor:

Daniel L. Dvorak
Systems and Software Division
Jet Propulsion Laboratory
California Institute of Technology

Figure 1: NASAs study from 2007 [1]

In 2009, after having suffered from various challenges on robustness of flight control systems, NASA started an investigation to study "the factors that have led to the accelerating growth in flight software size and complexity".

14 years later embedded real-time software are omnipresent in a wide range of industries far beyond space. New software and development approaches have been brought in to help engineers not just "rocket scientist" to address the challenge. But principles are still the same, and even today difficult to fulfill. NASAs report is summarizing well, what to consider, and how to master increase in (over)complexity.

It utilises a very practical definition of complexity "how hard something is to understand or verify.". Which summarizes nicely the human and technology aspect, as "to understand", "to verify" can always mean two things the teams needs more help and guidance, or the system is per se complex. Maybe too complex for the team or its size to handle.

It provides a workable approach for distinguishing "complexity from essential functionality, which comes from vetted requirements and is therefore unavoidable, and incidental complexity" that needs to be reduced.

Recommendations on team development, uncoordinated local decisions, trade(off) analysis, requirements management, good software architecture, doing workarounds versus fixing, fault management software, working on platforms, using components off the shelve, addressing residual errors are as fresh as of today as they were 15 years ago, and have to be seen under cultural aspects, not just technological ones.

From **nasa2007?** :

"*Some of the recommendations summarized above are common sense but aren't common practice. We identified four cultural reasons for the current state.*

- *First, cost and schedule pressure lead managers and developers to reduce or eliminate activities other than the production of code. Improvements that require time and training are vulnerable, particularly if their benefits are hard to quantify.*
- *Second, there is a lack of enforcement; some recommendations already exist in NASA documents and/or local practices, but aren't followed, possibly because nobody checks (and because of schedule pressure).*
- *Third, there is strong pressure to re-use software from a previous mission because it is "flight-proven" and presumed to be of lower risk. That reasoning sometimes has merit, but can inhibit better approaches, particularly when legacy software is partly to blame for difficulties on previous missions.*
- *Finally, there is no programmatic incentive for project managers to "wear the big hat" and contribute to infrastructure and process improvement for the benefit of future missions. This can be changed, as evidenced by many companies that have transitioned from stovepipe product development to product-line practices.*"

Over the years, I have seen very similar issues arise in the networking industry. I think it applies to real-time software activities in general. The size of the team may change the way you approach it, but it does not take away the challenge. The success of initially small private space startups tackling even higher complexity shows us the importance of managing (over)complexity to achieve success.

References

[1] D. Dvorak, "NASA study on flight software complexity," *AIAA Infotech at Aerospace Conference and Exhibit and AIAA Unmanned...Unlimited Conference*, Apr. 2009, doi: 10.2514/6.2009-1882.