

Letters to the Editor



Dear Paul,

I am writing to you in response to Gavin McCall's presentation at SSS'24 [1]. I was disappointed that someone developing software for a safety application has so little confidence in his own team's ability to develop high integrity software and so much confidence that strangers could do better.

Gavin's presentation started with an example where a developer was faced with the choice between developing 500,000 Source Lines Of Code (SLOC) of bespoke software, and using open-source software instead. It is very likely that the open-source software will contain many features that are not needed in this particular application. Supposing the open-source software contains 2 million SLOC, it follows that the open-source software would need to exhibit a defect rate (expressed as defects per thousand lines of code) four times better than the bespoke software to result in the same number of defects overall.

In my experience, most safety applications contain far less than 500,000 SLOC. It is true that there has been an exponential increase in the size of many software applications. Software productivity hasn't improved exponentially, neither have team sizes. Most of the increase in code size has come from reusing third party software, including Commercial Off The Shelf (COTS) and open-source software. While this can be a means of delivering software quickly when the software only needs to be "good enough", it is not an effective means of developing high integrity software. I am reminded of Blaise Pascal, who wrote "I have made this [letter] longer than usual because I have not had time to make it shorter".

Gavin's presentation used the Linux kernel as the only example of open-source software. I agree that the Linux kernel appears to be very reliable. I don't know how it compares with operating systems targeted specifically at safety applications, such as BlackBerry QNX, Green Hills Software INTEGRITY or DDC-I Deos. However, Linux is much more than just the kernel.

It doesn't follow that all Linux distributions are suitable for supporting safety applications. The SIL2LinuxMP project tried for three years to qualify Linux as a pre-existing software element following route 3S in IEC 61508-3, with the goal of providing a minimal run-time environment for safety-critical applications up to SIL 2 (See "Certifying Linux: Lessons Learned in Three Years of SIL2LinuxMP" [2]). The Linux Foundation's Enabling Linux in Safety Applications (ELISA) project is currently looking at how to make it easier for companies to build and certify Linux-based safety-critical applications, though they emphasise they are not producing a safe Linux distribution (see "ELISA - Advancing Open Source Safety-Critical Systems – ELISA" [3]).

The fact that the Linux kernel is good does not mean that all open-source software is good. You cannot extrapolate from the specific to the general. Most open-source software is of much lower quality than the Linux kernel. I was asked to assess an open-source software library developed by a major corporation. The open-source software library contained 381,512 SLOC of software, but only 5,291 SLOC of tests. This is typical of most open-source software developments, where the number of developers greatly exceeds the number of testers.

I have been privileged to have worked with some world-class programmers, including but not limited to Dave Thomas [4], Peter Amey [5] and Rod Chapman [6]. The software development process that I helped to develop at Praxis, called Correctness-by-Construction, was recognised by the US Department of Homeland Security as one of only four software development processes worldwide that can consistently deliver a defect rate of less than 1 defect per thousand lines of code (see "Processes to Produce Secure Software" [7]).

Open-source software is not a silver bullet, nor is it a substitute for sound software engineering practices. The processes and techniques required to deliver high integrity software have been well-known for decades and applied by responsible software developers. The state of the art continues to advance, as described by Rod Chapman in his excellent talk at HISC 2023: "Automated Reasoning in and about the Cloud" [8].

Yours,

Eur Ing Dewi Daniels CEng FBCS

References

- [1] RAFIA - Using open-source software in an Automotive safety application - a case study, Gavin McCall, <https://scsc.uk/re1007.39:1>
- [2] https://www.researchgate.net/publication/323445105_Certifying_Linux_Lessons_Learned_in_Three_Years_of_SIL2LinuxMP
- [3] ELISA - Advancing Open Source Safety-Critical Systems – ELISA, <https://elisa.tech/>
- [4] Dave Thomas (programmer) – Wikipedia, [https://en.wikipedia.org/wiki/Dave_Thomas_\(programmer\)](https://en.wikipedia.org/wiki/Dave_Thomas_(programmer))
- [5] Peter Amey, <https://www.sigada.org/Peter-Amey.pdf>
- [6] Rod Chapman <https://www.linkedin.com/in/rod-chapman-7b60266>
- [7] Processes to Produce Secure Software, <https://apothecaryshed.files.wordpress.com/2018/11/secure-software-process1.pdf>
- [8] Automated Reasoning in and about the Cloud, High Integrity Software Conference, <https://www.his-conference.co.uk/session/automated-reasoning-in-and-about-the-cloud>