A Reply to “How the Boeing 737 disaster looks to a software developer”

Jack K. Horner

Last modified: 4/22/2019 5:18 PM

Greg Travis’s recent op-ed in *IEEE Spectrum* ([1]) is one of the best summaries of the Boeing 737 Max 8 disasters I’ve seen. The following comments reinforce and extend his highly informative observations:

1. The 737 Max 8 tragedies, as [1] rightly says, concern failures in *systems*, not just *software*, engineering. In this case, the system includes software, hardware, end-user training, societal values, legal and regulatory considerations, and the relevant interactions among all these domains.

2. Turing's celebrated Halting Problem ([3]) notwithstanding, it has been informally known in the software engineering community for at least 50 years that there is no procedure that would allow humans to exhaustively *test* the behavior of software that (a) contains more than ~300 lines of code, and (b) has on average one binary branch per 10 lines of code. It is not unusual for modern flight-software systems to contain several hundred thousand, to a few million, lines of code, developed in environments that themselves contain between one and 20 million lines of code, none of which has been exhaustively tested. This implies that only a very tiny fraction of the behavior modes of a typical production-class software system can ever be tested, and thus, in general, ever be known ([2]).

Worse is true. Let a finite agent be an agent who, among other things, must acquire (non-deductive) knowledge by some sequential procedure and who has a finite (even if huge compared to a human) lifetime. It can be shown that it is impossible for any finite agent, no matter how long her finite lifetime is, to exhaustively verify that a software system implements *arithmetic* ([4]).

3. The question of who, if anyone, is at fault in the Max 8 tragedies will have to be determined by various jurisdictions. If those determinations are to be made comprehensively and dispassionately, how costs (including airline ticket costs) must trade against other values in which we have an interest (e.g., safety) will have to be made thoughtfully and explicitly. As of now, there is much to be done to reach that goal.

---

[1] Travis G. (18 April 2019). How the Boeing 737 Max disaster looks to a software developer. *IEEE Spectrum*. https://spectrum.ieee.org/aerospace/aviation/how-the-boeing-737-max-disaster-looks-to-a-software-developer.

[2] Symons JF and Horner JK. (2017). Software Error as a Limit to Inquiry for Finite Agents: Challenges for the Post-human Scientist. In Powers T. (ed.) *Philosophy and Computing: Essays in Epistemology, Philosophy of Mind, Logic, and Ethics.* Springer. https://doi.org/10.1007/978-3-319-61043-6_5.

[3] Turing A. (1937). On computable numbers, with an application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society*, Series 2, Volume 42, 230–265. doi:10.1112/plms/s2-42.1.230.

[4] Horner JK and Symons JF. (2019). Why there is no general solution to the problem of software verification. Forthcoming in *Foundations of Science.*

Jack K. Horner helped to develop software for command-and-control, large-sensor, satellite-based-navigation, medical-diagnostic, and radiation-hydrodynamics systems. In retirement, he maintains an active research interest in automated-deduction, computational-biology, and population-dynamics/resources-simulation applications. He is currently writing a book (with John Symons) about how software has changed the nature of scientific knowledge. The views expressed in these comments are not intended to represent the views of his former employers. He can be contacted at jhorner@cybermesa.com.