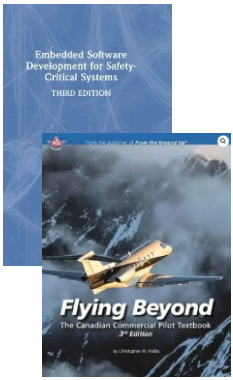
60 Seconds with ... Chris Hobbs



Chris Hobbs was first required to make a decision about shipping some potentially dangerous software in the late 1980s and has been involved with this topic since. He programs in Ada, C, Python and, more recently, Rust and particularly specialises in applying formal mathematical models to prove the correctness of algorithms and system design.

He is a member of the Canadian Mirror Committee considering version 3 of IEC 61508 and a stakeholder in the development of UL 4600. He is also a member of two SCSC working groups: one on the production of Assurance Cases and one providing guidance on the safety assurance of complex, socio-technical systems.



The third edition of his book "*Embedded Software Development for Safety-Critical Systems*" has just appeared.

Apart from software development, Chris is a pilot and the author of various aviation books including "*Flying Beyond: The Canadian Commercial Pilot Textbook*". Working with various instructors, he has also been learning the 24 songs of Schubert's Winterreise cycle for the past 24 years.

What first attracted you to working in the field of System Safety?

In the late 1980s I was working on a piece of equipment for sale to British Telecom. A technician maintaining the equipment could command it to connect test gear to the external, copper line. Before it did so, the equipment checked to see whether there was a high voltage on the external line and, if there was, wouldn't close the relays, thereby protecting the technician.

A short time before deployment, a developer reading the code noticed that, under certain unlikely circumstances, the high voltage check would be skipped. As team leader, I was under pressure to sign the equipment off: as management pointed out, a high voltage was rare and the circumstances of the test being skipped were also rare. I had none of the tools I have now to decide whether or not I should sign it off. After a diversion into high-dependability software, I returned to a focus on safety in 2009.

What aspect of your career are you most proud of?

The fact that I've never had a "career". I've had a long sequence of interesting and challenging "jobs", and have even on some occasions moved from one job to a more challenging job where the pay was less.

Tell us one interesting unusual fact about yourself!

I spent two years living in Libya when I was 11 and 12 years old – my father was a civilian engineer working at the RAF base in El Adam. This gave me the opportunity to follow dangerous pursuits (e.g. putting a match to the cordite fuses in unexploded munitions left over from the Second World War), but it also meant that I missed my Eleven-Plus examination. On returning to the UK, my father had to plead with the headmaster of Huntingdon Grammar School (Oliver Cromwell's *alma mater*) to get me a place in the second form there.

What advice would you give to yourself age 12?

When my children left home, I gave them a piece of worldly advice that should have been told to me when I was 12: "Never say you understand something when you don't".

What future changes would you like to see in the field of System Safety?

I would like to see IEC 61508, ISO 26262, EN 50716 and other standards accept the possibility of random software failure. IEC 61513 gets it right – "*Since the triggers which activate software faults are encountered at random during system operation, software failures also occur randomly.*" – but many of the standards explicitly state that all software failures are systematic. Edition 3 of IEC 61508 (due out in early 2026) has dug even deeper than Edition 2 into this hole. The misunderstanding seems to stem from inexperience with multi-threaded systems on multi-core processors, but even simple programs have been failing since the 1950s on certain (uncontrolled) inputs by users. This lack means that our safety analyses of any system containing software are fundamentally flawed.

What's your most favourite quote or motto?

"Many promising reconciliations have broken down because while both parties came prepared to forgive, neither party came prepared to be forgiven." Charles Williams

If you could learn to do anything, what would it be?

In 2002 I was offered a job with a startup company in California. I decided not to take it, but realised that if I was not going to take up that challenge, then I would need a different big challenge to compensate. With my wife as accompanist at the piano', I decided to learn the 24 songs in Schubert's Winterreise song cycle. Following weekly lessons from a variety of teachers and evening practices we have begun to explore the depth of those songs: perhaps the peak of any male singer's ambition. You may say I'm clearly a slow learner (23 years to learn 24 songs), but if I could learn to do anything, I would want to sing those 24 songs as Ian Bostridge CBE does (but in a lower key!)

I understand you train commercial pilots – what's the most important lesson for flying safely?

I have retired from the actual training, although my commercial pilot textbook is still being used (now in its 3rd edition). I think that the most important lesson is understanding the equipment (and its interface). The simulator is the place to be learning which buttons to push when approach control changes a clearance via a particular waypoint to a vectored approach, not in thick cloud with the possibility of icing! And, of course, a pilot needs to know the limitations of that equipment.

If you could be any fictional character, whom would you choose?

I assume that Oliver Mellors is excluded on grounds of good taste and so I must choose Hamlet's friend, Horatio. He seems to represent sanity, intellectual neutrality, friendship and the ability to offer difficult advice without personal bias.

How do you see the rise of Artificial Intelligence affecting the future of safety engineering?

I find it difficult to define what is meant by the term AI: in the popular imagination it seems to be becoming synonymous with Machine Learning and, in particular, with Large Language Models (LLMs). I think that, with LLMs, AI has taken a significant step backwards since the "commonsense knowledge" work (e.g., CYC and ConceptNet) of the early part of this century. The LLMs are now devouring works generated by other LLMs with no semantic knowledge of what is being produced. And I don't think that Artificial General Intelligence (AGI) is achievable.

But this leaves open questions on the safety of autonomous devices, particularly road vehicles. We have seen the accidents caused by Phil Koopman's long-tail events: the inability to recognise a woman pushing a bicycle or a woman flying through the air.

For safety-critical systems, I see the danger in AI being not with the technology, but with people's interaction with that technology. We have seen teenagers believing what AI bots are telling them and if AI is deployed to assist humans working in potentially dangerous conditions, those humans must very clearly understand the limits of the AI system.