TIM Lecture Series
When Are Software Systems Safe Enough?

Chris Hobbs

> *"We have an ethical duty to come out of our mathematical sandboxes and take more social responsibility for the systems we build – even if this means career threatening conflict with a powerful boss. Knowledge is the traditional currency of engineering, but we must also deal in belief. The techniques of persuasion must become part of the engineering toolbox. If the safety integrity of a system is compromised by a bad management decision, it is our duty to speak truth to power and change belief systems. The alternative is to risk enduring regret for the shortened lives of the people who put their faith in our skills."*
>
> Les Chambers
> Systems engineer and author

Overview

The TIM Lecture Series is hosted by the Technology Innovation Management (TIM; timprogram.ca) program at Carleton University in Ottawa, Canada. The lectures provide a forum to promote the transfer of knowledge between university research to technology company executives and entrepreneurs as well as research and development personnel. Readers are encouraged to share related insights or provide feedback on the presentation or the TIM Lecture Series, including recommendations of future speakers.

The seventh TIM lecture of 2015 was held at Carleton University on October 29th and was presented by Chris Hobbs, a Software Safety Consultant at QNX Software Systems (qnx.com). The lecture covered the changing nature of safety-critical software over the last 20 years, including a brief discussion of the standards that are directing development in the medical, industrial, and automotive fields. Hobbs also demonstrated some of the tools recommended in the safety standards and which are used during design verification.

Summary

By an enormous margin, most of the computers active today are embedded into devices and are invisible to users. Increasingly these embedded devices are being deployed in applications where injury to human life or the environment can occur if a failure occurs. Examples include embedded systems in cars, aircraft, nuclear power station controllers, railway signals, railway braking systems, and medical devices.

In this TIM Lecture, Chris Hobbs described his recent work with railway signalling systems, robots performing hip surgery, industrial robots working alongside humans, medical analyzers, undersea drill-heads, and autonomous and semi-autonomous cars. The development of these types of system places great stress on the validation and verification not only of the product, but, more importantly, its architecture and design.

Hobbs cautioned that, "A system cannot be safe. It is a matter of whether it is safe enough." And, determining whether a system is "safe enough" requires an understanding of both risk and safety, and the context in which a particular system will be used. The International Organization for Standardization (ISO, 2011) defines risk as a "combination of the probability of occurrence of harm and the severity of harm", whereas unreasonable risk is "risk judged to be unacceptable in a certain context". In contrast, safety is described as the "absence of unreasonable risk according to valid societal moral concepts".

So, how can we test whether a particular system is safe enough? The International Software Testing Standard–ISO/IEC/IEEE 29119 (2013) – states that, due to the complexity of systems and software, it is impossible to test a system exhaustively; testing becomes a sampling activity. Even dynamic testing is "not sufficient to provide reasonable assurance that software will perform as intended".

In part, the goal of software testing is to assess the availability and reliability of a system. Availability asks, "Does the system give an answer?", whereas reliability

TIM Lecture Series – When Are Software Systems Safe Enough?

Chris Hobbs

asks, "Is the answer correct?" Although both of these aspects are important, depending on the system and its functional context, either availability or reliability might be more important for safety. For example, it is safer for some systems to determine that, if a reliable answer cannot be given, then no answer should be given. However, in other contexts, even some degree of unreliable information may be safer than no information at all. Unfortunately, when it comes to testing, there are many more techniques for assessing availability than reliability. Developers must carefully consider this balance and determine which aspect is more important for the safety of their system – and make design decisions accordingly.

Given that we cannot design completely safe systems, we must somehow decide what is safe enough. Hobbs described three methods commonly used to assess risk and decide whether a given system is safe enough:

1. *As low as reasonably practical (ALARP):* society determines what levels of risk are unacceptable and broadly acceptable, and in between there is an area where financial decisions (often based on the value of a human life) will influence risk-mitigation efforts.

2. *Globalement au moins aussi bon (GAMAB):* a new system must offer a global level of risk no worse than that offered by an existing equivalent system.

3. *Minimum endogenous mortality (MEM):* risk is assessed based on the underlying likelihood of death by accident, and new systems must not add more than a particular level of risk to that baseline amount, which is country/market-dependent.

In any case, developing a software system to an acceptable of safety does require careful attention to risk and some additional work. Hobbs estimates that development to a safety-critical standard requires only 10% additional effort above a "professional development" standard, but he notes that many companies are actually developing software to a much lower standard, which makes the additional costs associated with developing safe software seem high. For companies already used to developing commercial-grade software, developing safety-critical software does not require that much extra effort.

However, aside from the additional costs in time and development effort, there is also the certification process, which is not easy. Lloyd and Reeve (2009) reported on the certification attempts of 16 companies and found that, at the time of sampling, only 25% of those attempts that had reached an outcome resulted in successful certification: more than half of the companies failed simply by not completing the certification process.

A key element of design for safety – and one that is required by safety standards – is the development of a safety culture within development organizations. A safety culture includes aspects such as accountability for decisions related to functional safety, highest prioritization given to safety, a proactive attitude towards safety, processes that include checks and balances, deliberate allocation of required skilled resources, and fostering and valuing intellectual diversity (ISO, 2011).

Hobbs highlighted that developers are currently facing significant challenges in designing and implementing these types of systems. In the remainder of the lecture, he demonstrated development and testing tools, gave an overview of some example standards from the automobile industry and how they are structured, and he identified major areas where research is required, such as tool integration, standards and tools for security, and tools to help developers manage the competing demands for performance, availability, reliability, security, and safety.

In summary, the lecture focused on the following key messages:

• Almost all of today's computers are embedded devices.

• An increasing number of those devices are performing safety-critical roles.

• The software for those devices needs to be dependable.

• We can no longer test software to ensure that it is working properly.

• There are many problems with embedded devices: ephemeral and difficult-to-diagnose bugs, hardware susceptibility, and a lack of tools. And, these problems are getting worse.

• Security is now an integral part of safety.

• There are international standards on the development of safety-critical software.

TIM Lecture Series – When Are Software Systems Safe Enough?

Chris Hobbs

References

ISO. 2011. *ISO 26262: Road Vehicles — Functional Safety.* Geneva: International Organization for Standardization (ISO). https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-1:v1:en

ISO/IEC/IEEE. 2013. *The International Software Testing Standard.* Geneva: International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC) / Institute of Electrical and Electronics Engineers (IEEE). http://www.softwaretestingstandard.org

Lloyd, M. H., & Reeve, P. J. 2009. *IEC 61508 and IEC 61511 Assessments – Some Lessons Learned.* Paper presented at the 4th IET International Conference on Systems Safety. London: The Institute of Engineering and Technology (IET). http://dx.doi.org/10.1049/cp.2009.1540

About the Speaker

Chris Hobbs is a Software Safety Consultant at QNX Software Systems in Ottawa, Canada. He was educated as a mathematical philosopher, but finding few jobs available for mathematical philosophers, fell enthusiastically into computer programming where he has spent the last 40 years avoiding management positions and remaining at the leading edge of software development. At QNX Software Systems, he is part of a team focussed on deploying QNX's operating system into safety-critical systems. He works on the safety certification of QNX's products and spends a lot of time with QNX's customers, helping them to design systems to meet specific safety requirements. He is the author of *Embedded Software Development for Safety-Critical Systems* and *The Largest Number Smaller Than Five.*

This report was written by Chris Hobbs and Chris McPhee.

Citation: Hobbs, C. 2015. TIM Lecture Series – When Are Software Systems Safe Enough? *Technology Innovation Management Review*, 5(12): 56–58. http://timreview.ca/article/953

Keywords: safety, risk, security, software systems, safety-critical systems, standards, testing