Successes in Deployed Verified Software (and Insights on Key Social Factors)

June Andronick(✉)

CSIRO's Data61 and UNSW, Sydney, Australia
june.andronick@data61.csiro.au

Abstract. In this talk, we will share our experience in the successful deployment of verified software in a wide range of application domains, and, importantly, our insights on the key factors enabling such successful deployment, in particular the importance of the social aspects of a group working effectively together.

Our formally verified microkernel, seL4, is now used across the world in a number of applications that keeps growing. Our experience is that such an uptake is enabled not only by a technical strategy, but also by a tight integration of people from multiple disciplines and with both research and engineering profiles. This requires a strong social culture, with well designed processes, for working as one unified team. We share our observations on what concrete social structures have been key for us in creating real-world impact from research breakthroughs.

1 The Dream

Precisely fifty years ago, Tony Hoare, in his seminal paper [1], outlined a dream; a dream where verifying properties of programs can be achieved by purely deductive reasoning; a dream where such reasoning could be applied to non-trivial programs as long as considerably more powerful proof techniques became available; a dream where software systems would not be deployed unless they were formally verified; a dream where verified software would have become the standard produced by industry; a dream where it would be legally considered negligence to deploy unverified software.

We share this dream, and –with many others– have contributed towards it by demonstrating that verified software is feasible and can be deployed on real-world systems. We can, however, observe that, in fifty years, this dream has not been fully achieved yet.

The main reason for verified software not yet being the standard could be phrased as: it has not yet achieved the status of being the *state-of-the-art*. Ten years ago, Hoare was invited to write a retrospective article [2], to share his personal views on progress made since his first article forty years before, and reflect on what he had hoped for back then and what actually happened. One thing he realised he had not predicted correctly was what actually would drive the push for more verified software; he had thought that it would be the fear

M. H. ter Beek et al. (Eds.): FM 2019, LNCS 11800, pp. 11–17, 2019.
https://doi.org/10.1007/978-3-030-30942-8_2

of expensive lawsuits for damages due to software errors. This didn't happen because "the defense of 'state-of-the-art' practice would always prevail".

We thus need to make verified software become the state-of-the-art practice. For this we need (a) to lower the cost (Hoare said "Far more effective is the incentive of reduction in cost" [2]) and (b) to have more success stories, where insights can be shared. Together, these will not only bring economic incentives for all software producers to follow the path of verified software, but lead to 'no more excuses' *not* to follow that path.

Here we share our observations about the social structures and incentives that have allowed us to bring together a large group of people with diverse –sometimes even disjoint– technical backgrounds and to make them work effectively towards a goal that must blend relentlessly formal techniques on the one hand with uncompromising real-world performance on the other. In the last ten years, we have been designing, developing, maintaining, and evolving the world's largest and most verified artefact, ported across multiple hardware platforms, as well as a collection of tools and frameworks for the verification of real-world software. In the last five years, our technology has seen an increasing uptake by companies, governments and the open-source community. This has encouraged a number of initiatives and projects pushing further this pervasive verified software dream. Reflecting on our own experience of what made it possible to push the boundaries of the state-of-the-art into deployed systems, our main insight would be (1) having a single group with both researchers and engineers, and both operating-system (OS) and formal method (FM) experts, all working very closely together, towards a shared vision, and (2) having this vision being not only technical, but also social: making sure this diverse range of people work effectively and efficiently together. We will first give an overview of where our verified software is deployed and the key steps leading to this uptake, and then share our observations on the key social factors that allowed these successes.

2 Successes in Deployed Verified Software

Our story of successfully pushing verified software in deployed systems across a variety of domains contains a few important milestones.

Performance. The first milestone, and starting point, was the research breakthrough making formal program verification scale to an entire operating system kernel, *while maintaining high performance*. This consisted in the formal proof, in the Isabelle/HOL theorem prover [7], of the functional correctness of the seL4 microkernel [3,5], followed by the proof of the security properties of integrity [9] and confidentiality [6] as well as correctness of the binary code [8]. Note that the focus on performance as an equal objective as the correctness was a key factor in the later uptake and deployment; and this was made possible only by the close collaboration between the two disciplines' experts, as we will describe in the next section.

Retrofitting. A second key milestone was to move this research outcome towards a technology transfer in industry by demonstrating the practicality of building whole secure systems on the seL4 trustworthy foundation. We worked with companies to *retrofit* their existing systems into secure architectures, with isolated trusted components, running on seL4 guaranteeing the isolation (as describes in [4]). The key effort that created the most impact was the High-Assurance Cyber Military Systems (HACMS) program, funded by the US Defense Advanced Research Projects Agency (DARPA), where we collaborated with Boeing, Rockwell Collins, Galois, HRL, the University of Minnesota, MIT, University of Illinois, CMU, Princeton University, and US Army's TARDEC to transition the seL4 technology to real-world autonomous vehicles. These included a Boeing-built optionally piloted helicopter and an autonomous US-Army truck, both originally vulnerable to cyber-attack, that we demonstrated to be able to resist these cyber-attacks and others after being re-architected to run on seL4. This kind of work is mainly engineering focused, with a join effort between the systems engineers and the proof engineers, keeping the focus on formal guarantees for the security of the overall system. Such projects are also an important source of input about the real-world requirements that need to be addressed.

Focus on Grand Challenges. This leads to the third key ingredient: keep tackling the grand challenges not yet addressed. Our engineering work, pushing our technology on deployed systems, harvests further requirements calling for still more research advances, such as extending the verification guarantees to timing protection, or concurrent execution on multicore platforms, or increasing the cost-effectiveness of verifying application code or porting the proofs to new platforms. These open questions then constitute our research agenda and roadmap.

Open Source. Finally, the last key contributing factor to the uptake of our technology was the open-sourcing of seL4, both code and proofs, as well as all the infrastructure, tools, and platforms to help building whole secure systems. The first reason why this contributed to the uptake is that a kernel is only a part of the solution, and transitioning to using it requires a retrofit, a re-architecting of an existing system, which is not a decision taken lightly. Being able to explore and 'play' with it before 'buying into it' has been instrumental to people choosing to transition. The second reason open sourcing has been critical is that it builds a community and an ecosystem supporting and extending the technology, infrastructure, libraries, and platforms, helping with the scalability of the support for transitioning. The caveat and challenge is to ensure that the verification guarantees keep being maintained.

These few key milestones have led to an increased uptake of the seL4 kernel and associated technology in real-world systems across a number of domains: automotive (e.g. HRL, TARDEC), aviation (e.g. Boeing, Rockwell), space (e.g. UNSW QB50), military (e.g. Rockwell Soldier Helmet), data distribution (e.g. RTI Connext DDS Micro), Industry 4.0 (e.g. HENSOLDT Cyber), component

OS (e.g. Genode platform), security (e.g. Penten Altrocrypt). Some of these projects are a result of DARPA's call for specific funding to build the seL4 ecosystem, through a number of Small Business Innovation Research (SBIR) grants.

Much work is still to be done (and is ongoing) to lower the bar to transition to seL4-based systems, and to ensure the verification guarantees are maintained and extended, but these successful deployments are contributing to pushing the dream of verified software becoming the default.

3 Insights on Key Social Factors

A major aspect of what we want to communicate here is the importance of *s*ocial factors, within our group[1], that we have discovered are key contributors to the technical aspects of what we have done. Our experience is that the successful uptake of our technology comes from having a single group hosting both FM and OS people, and both researchers and engineers, working effectively together, as a tightly integrated team. We want to share concrete examples of the social structures that enabled this tight integration for us. Some can be expected and are not unique to our group; we here simply share which ones seem to have been key for us.

Achieving the dream of pervasive verified software requires a combination of academic research and industrial engineering. Today, these mostly live in separated worlds. Industrial engineering brings the real-world requirements, requires usability and performance, but is product-focused and aims at profitability. Hoare said *"The goal of industrial research is (and should always be) to pluck the 'low-hanging fruit'; that is, to solve the easiest parts of the most prevalent problems, in the particular circumstances of here and now."* [2]. Academic research, on the other hand, is innovation-focused, aiming at generic solutions, with a timeframe allowing grand-challenges to be solved in a novel way. Hoare said that *"the goal of the pure research scientist is exactly the opposite: it is to construct the most general theories, covering the widest possible range of phenomena, and to seek certainty of knowledge that will endure for future generations."* [2]. When it comes to verified software, academic research is still crucially needed to increase the scalability and applicability, while industrial engineering is critical to produce specific instances that work.

There have been many studies on the barriers to the adoption of formal methods and the ideas for closing the gaps between academic research and industry practices. These studies paint the world as composed of two separate entities; the formal methods on one side, and the application domain on the other; or the research on one side, and the industrial engineering on the other — with a boundary in between that needs to be crossed, as a 'baton' transferred from one part of the world to the other.

[1] the Trustworthy Systems group, in Data61, CSIRO, https://ts.data61.csiro.au.

Our view is that success in deployable verified software comes with having one single world, one single team[2], tighly integrated. It is the notion of *tight* integration that is crucial. That is what prevents the (undesirable) re-creation, within the group, of the binary world we are presently forced to inhabit outside it. If we don't succeed there, then the same boundaries and gaps will be created — where work is 'handed over' by one set of people to another set of people for their consideration. Instead, people need to work hand in hand, day by day, sometimes even hour by hour, sharing their perspective of the issues, solutions, design decisions, all along the way.

In our group, this is illustrated by the fact that 'every project involves every subteam', meaning that the majority of our projects involve *both* OS and FM people and *both* researchers and engineers. Our engineering practices and processes on the OS side and FM side are also tightly integrated; for instance, any change in the code, from any side, starts with a discussion on the implications for the 'other side'; we have a continuous integration process that manages our implementation code base as well as all our proof code base (now more than a million lines of Isabelle/HOL), making sure they are always in sync, that changes to code that is not yet verified can be seamlessly integrated, as well as changes to verified code that happen to not break any proofs, whereas any changes that break the proofs are clearly marked as such and follow a process where a team is allocated to their verification and changes cannot be integrated until the proofs are re-established.

For this tight integration to work, the *frequency* of the personal interactions is crucial. Our group has experienced a few different physical setups, in different locations, and our observation is that having people in the same location, same building, if possible same floor is highly desirable: a proof engineer can just walk up to the OS engineer to check e.g. whether a change in the code to ease verification would have a performance impact; the impromptu encounters at the coffee machine create the opportunity to share a viewpoint on e.g. a desired kernel change; the kind of discussions needed across disciplines work best as face-to-face discussions, with the support of a white board for design brainstorming, or for sharing the knowledge between disciplines.

Another very important social aspect is ensuring *good* communication despite the difference in backgrounds, or even sometimes languages and terminologies. For instance, like many other groups, we run weekly talks and quarterly dive-ins to update the rest of the group on progress in various project or share knowledge in a specific area. Maybe unlike many other research groups, these talks cross discipline boundaries and we strive –and in fact need– to keep them at a level all can understand i.e. the OS-based talks have to be FM-comprehensible, and vice versa. And *everyone* must give one of these talks, on a regular schedule. This way everyone get the opportunity to share their views, to attract interest in their work, and to grow their skills in explaining their work. This fosters a culture inside the group of knowledge sharing and awareness of other people's

[2] and if possible, importantly, one single shared coffee machine, surrounded by plenty of whiteboards.

work, which is essential when having to then deliver together on a given project. Being able to effectively communicate technical work to people outside of the field is not easy. To help with this, we run annual 'bootcamps' focusing on training ourselves on communication and presentation skills, and learning how to best adapt to various kinds of audience. This has an important direct impact on getting traction externally to increase the uptake of our technology, and verified software in general. Importantly, it also enables the needed information sharing and productive collaboration within the group.

Creating a one-team culture goes beyond the communication aspect. It requires a technical vision that everyone shares, shapes and contributes to. But it also needs a culture of achieving this vision *together as a team*, where we have the urge to see each other succeed, where we help and support each other in solving hard problems and delivering on projects, and where everyone contributes to creating an environment where everyone can thrive. One way this is achieved in our group is that a lot of activities such as trainings, social events, or cultural awareness initiatives are done *by people from the group*, and tailored to what our groups needs. For instance, our bootcamp mentioned above includes sessions on active listening, mental health, life balance, and all sessions are given by members of the group that have either training or first-hand experience in the topic, and are delivering tailored information and practice that they know are relevant to the type of work we do. The impact of this approach is that the trust that people have in their peers amplifies the impact of the message, the learning experience or the social interaction. It also extends the scope of the collaboration between people from purely technical to all social aspects of the group's life.

All of the above creates and fosters an environment where you can get a unique combination of people with different expertise and profiles that can work well together to achieve their shared mission. Dealing with a truly wonderful mix of personalities, backgrounds and cultures does create a number of challenges, but it also creates the required structure to tackle and solve research grand challenges, while producing systems, tools and frameworks that the world can use and deploy, and while building a community of users, partners and contributors. And this is what is needed to achieve the dream of shifting the whole world's mentality towards accepting verified software as the norm.

Acknowledgements. The author would like to thank Gerwin Klein and Carroll Morgan for their feedback on drafts of this paper.

References

1. Hoare, C.A.R.: An axiomatic basis for computer programming. CACM **12**, 576–580 (1969)
2. Hoare, C.A.R.: Viewpoint - retrospective: an axiomatic basis for computer programming. CACM **52**(10), 30–32 (2009)
3. Klein, G., et al.: seL4: Formal verification of an operating-system kernel. CACM **53**(6), 107–115 (2010)

4. Klein, G., Andronick, J., Kuz, I., Murray, T., Heiser, G., Fernandez, M.: Formally verified software in the real world. CACM **61**, 68–77 (2018)
5. Klein, G., et al.: seL4: Formal verification of an OS kernel. In: SOSP, pp. 207–220. ACM, Big Sky, October 2009
6. Murray, T., et al.: seL4: from general purpose to a proof of information flow enforcement. In: 2013 IEEE Symposium on Security and Privacy, pp. 415–429. IEEE, San Francisco, May 2013
7. Nipkow, T., Wenzel, M., Paulson, L.C. (eds.): Isabelle/HOL. LNCS, vol. 2283. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45949-9
8. Sewell, T., Myreen, M., Klein, G.: Translation validation for a verified OS kernel. In: PLDI, pp. 471–481. ACM, Seattle, June 2013
9. Sewell, T., Winwood, S., Gammie, P., Murray, T., Andronick, J., Klein, G.: seL4 enforces integrity. In: van Eekelen, M., Geuvers, H., Schmaltz, J., Wiedijk, F. (eds.) ITP 2011. LNCS, vol. 6898, pp. 325–340. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22863-6_24