© Peter Sommerlad

Professor Peter Sommerlad
Founder and Leader
of the Institute for Software
at the HSR Hochschule für Technik
in Rapperswil, Switzerland

Back to the Future of Software?

As a software person by heart and soul I set my lifetime goal to rid the world of bad software. I understand that this will not happen, unless there is no more hardware, but “the way is the goal”. I feel like I’ve been taken back in a time machine when I’m around embedded software people – especially from automotive. They discuss issues, that the “normal” software community has solved or abandoned decades ago.

One reason for revisiting my past seems to be, that only few of the embedded software engineers are software people by education, but slipped into the role from other disciplines. Another reason is the conservative nature of the environment due to limited hardware. However, ignoring lessons learned and repeating mistakes made by others in the past is inefficient engineering, especially now, where 32-bit processors and megabytes replace 8bit and few kilobytes.

What I observe is reluctance to important aspects of good software design and modern technology, such as encapsulation with clear interfaces, separation of concerns, parameterisation, efficient abstraction, current languages, compilers, and IDEs, and striving for simplicity. Especially the last issue is often misunderstood as an excuse for sticking with “simple” programming languages that only provide primitive constructs and lack higher-level abstraction capabilities.

Programming with such dumbed-down languages results in bloated code without easy understandability. Simplicity of software means encapsulating abstractions and providing and using interfaces to interact with other components, so that less code is required, increasing maintainability. To get to such design means rework software to clean up mess, often avoided, because pay-off is non-intuitive or changes are considered dangerous and expensive. But it is called “soft” for a reason.

A last thing worth mentioning, is misunderstanding process and certification with quality and its non-intuitive cost model. Lower quality software is more expensive, even produced with a certified mature or modern – often misunderstood – “agile” process. In most other products, lower quality leads to reduced costs and increased profits. Not in software, unless you are a contractor billing by time and material.

If you are responsible for embedded, get “back to the future” and catch up with the now in software engineering, e.g., encourage tool vendors to implement recent international standards and avoid repeating mistakes others made decades ago. Give your people time to learn and improve, but do not take as long as others before you. Such a change looks tough and requires effort, but it will also heavily improve your engineering future allowing to solve harder problems with less code.