Embedded software certification under a software engineering methodological perspective

Luca Pazzi*
Riccardo Corrado
University of Modena and Reggio Emilia
Via Vignolese 905, I-41100, Modena, Italy

Luigi Luppi
LUPPI & ASSOCIATI
V.le Corassori 54, I-41100, Modena, Italy

1 Introduction

Embedded systems pose a lot of challenges to the research community, both to academia and to industry. Moreover, the ever increasing diffusion of embedded devices in everyday life will soon bring the problematics of their safety certification to the attention of national as well as supranational institutions, which are, of course, already committed towards the safety certification of the products which are commercialized within their country-member boundaries. The European Union, for example, since the emanation of directive 2001/95/CE, is trying to bring commercial products sold within its boundaries towards a unitary and standardized level of safety. It may be observed that such a directive does not indicate explicitly any specific product typology, hence it leaves room for new product classes, such as embedded systems.

*Corresponding author, e-mail: pazzi@unimo.it

If national and supranational institutions are the natural seat of the new certification process, the rest of the feasible scenarios is still under deep fog. It is indeed necessary to face the following issues:

1. The very notion of safety for both human-controlled and automatic embedded devices is still under investigation. If, on the hardware front, the systemic notion of reliability is a well acquainted topic, on the embedded software front it is well known that system safety is very different from system reliability, since it is well known that a reliable system is not necessarily a safe system;

2. Even in the case of a widely accepted notion of safety, it is necessary to have some *methodological* guidelines in order to drive the certification process;

2 Towards a new notion of safety

The quest for a notion of safety suitable for embedded system certification is, in our opinion, essentially a software engineering problem. It may be observed that, at the moment:

- no tools or methodologies exist for determining, in all cases, whether a system is safe [1]. Although model checking made steps ahead in order to face more complex problems, it is still limited by the exponential growth of the number of global states of the system;

- current methodologies rely on Harel's Statecharts [2] model of computation, which has not a definite semantics;

- current methodologies lack any form of modularization; in this sense it is not possible to encapsulate part of a system, ascertain its safety, and to assemble it, by a *safety preserving* methodology, with other safe modules in order to get a safe system.

It is therefore rather straightforward to draft a methodology able to overcome the problems above; such a methodology should:

1. be truly modular, which in turn means that it should:

   (a) allow for safe modules, that is safety should be certifiable of a single module;
   (b) allow for modular safe assembly of safe modules;

2. underpin a computational model having a clear and computable semantics;

3. underpin a conceptual paradigm which is able to deal with a world of stimuli and responses in which embedded software is called to operate.

3 Towards a methodology-based safety certification

As observed, the different countries as well as their respective supranational organisations have the institutional duty to pave the way towards safety certification. Since we prospected that a notion of system security requires the adoption of methodological instruments, at least two scenarios may be likely hypothesized. A methodology may be indeed chosen by a certifying institution if it is able to ascertain the safety of a system under:

1. some scientifically indisputable result, such, for the example, one or more theorem proofs;

2. some weaker scientifical result, such as those coming from statistical evidence, in the same way evidence-based medicine operates.

The certifying institution may adopt either directly a single methodology, or a set of methodologies, each possibly owned by selected certifying firms, which become official (institutional) certifying entities;

In both cases, the certification process is consequently officially recognized by one or more countries, and as such it is likely to be adsorbed in the body of laws of each country. In case a country is not ready to choose one of the two scenarios above, it may be the case for a third one, in which a methodology which can be shown to be reliable and winning on the market, creates a *de facto* standard, and as such it may be likely used in a court of law to plead for the not-liability of a manufacturer who adopted it.

Whichever the road that will be chosen, we observe that a new market and business opportunities spring for certification firms, private and public

research institutions as well as lawyers and patent agencies. It may in fact observed that, in case one or more methodologies are adopted by a country, and in case such methodologies are patented, then the certification process may become a huge font of revenues for the patent holders.

4 Conclusion

Embedded software and system certification requires, as a prerequisite, the solution of pure software engineering problems. The main challenge consists in shifting from traditional structured programming, which may be shown to be not adequate for the new application field, towards a new state-based modular programming, which has yet still to come. Such a new paradigm will have necessarily to surpass Harel's model of computation, which is indeed not semantically well founded, as well as to give effective software tools, such as modularity and composability of safety assurance.

Finally, as observed, both the expected growth of the market as well as the feasible revenues coming from finding and patenting effective methodologies should convince research institutes in promoting investments in the field.

References

[1] Bruce Powel Douglass. *Real-time UML: Developing Efficient Objects for Embedded Systems*. Addison-Wesley, 1998.

[2] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.