2017 IEEE/ACM 39th IEEE International Conference on Software Engineering Companion

Software Certification in Practice: How Are Standards Being Applied?

Gabriel Ferreira
Carnegie Mellon University

***Abstract*—Certification schemes exist to regulate software systems and prevent them from being deployed before they are judged fit to use. However, practitioners are often unsatisfied with the efficiency of certification standards and processes. In this study, we analyzed two certification standards, Common Criteria and DO-178C, and collected insights from literature and from interviews with subject-matter experts to identify concepts affecting the efficiency of certification processes. Our results show that evaluation time, reusability of evaluation artifacts, and composition of systems and certified artifacts are barriers to achieve efficient certification.**

I. Certifying Software Systems

Software certification plays a fundamental role in defining guidelines and assessing the fitness of software systems in environments where failures can result in serious consequences. Failures can seriously affect business, the environment, and society, for example, when medical devices emit excessive radiation or when satellite launches explode [1]–[3].

Recent domains, such as IoT ecosystems and self-driving cars, pose new challenges that are not addressed by current certification efforts. Vendors are often unsatisfied with the inefficiency and value proposition of certification efforts and express concern on scalability issues. For example, the long development and certification time of aerospace software might be unacceptable to automobile industry [4].

Our study aims at identifying technical, social, and economic issues around certification efforts that affect the efficiency of certification efforts. We performed a multiple case study of two certification standards [5], *Common Criteria* and *DO-178C*, where we cross-validate insights from interviews, literature survey, and standards' documents. Recertification, reuse of evidence, and composition were common themes that emerged in our analysis. We expect our analysis can provide a starting point into the discussion of solutions for challenges across recent domains to increase the efficiency and cost-effectiveness of certification efforts.

II. Research Design

Our goal is to step back and to provide an overview of the social and technical challenges around certification standards affecting their efficiency and effectiveness, grounded in perceptions of various stakeholders. Specifically, we performed a multiple case study of two certification standards: *Common Criteria* which assesses security requirements of infrastructure and end-user software and devices and *DO-178C* which assesses safety requirements in software-based aerospace systems.

TABLE I
Roles of interviewees in the certification process.

Developers	CC1, CC6, CC7, DO1, DO3, DO4, DO6, DO8
Evaluators	CC2, CC3, CC4, CC5, CC9, CC10, DO2, DO5, DO7

We interviewed 18 experts working with these standards in various roles and cross-validated our results with an analysis of the standards' documents [6]–[8] (over 1200 pages) and a survey of papers discussing those standards. For the survey, we searched the web and digital libraries (IEEE Xplore, ACM), using "*Common Criteria*" and "*DO-178C*" as the main keywords to find papers that reflect on challenges faced by the two certification efforts. In total, we selected seven papers discussing Common Criteria [9]–[15] and five papers discussing DO-178C [16]–[20].

Table I shows the roles of interviewees in the certification process[1] for both Common Criteria (CC) and DO-178C (DO).

We use standard qualitative content analysis and coding techniques [21], [22] to summarize, reduce, and analyze our text-based data from interviews, papers, and standard's documents.

III. Results and Discussion

Applicability concerns regarding time, recertification, and composition were common topics in our interviews and in the literature.

A. Evaluation Time

Interviewees of the two standards had different views on the urgency of releasing products and on whether the evaluation process duration is an issue, with Common Criteria developers much more concerned about timely certification than DO-178C stakeholders.

For DO-178C, the evaluation time is aligned with the time required to develop complex aviation products, such as airplanes or satellites. The development can take years, usually from three to five depending on their complexity (DO6, DO7), giving vendors enough time to think and plan the evaluation of their products. Also, the mandate certification process requires them

[1]In addition to developers and evaluators, we also interviewed a policymaker (CC8), member of the Common Criteria certification scheme.

978-1-5386-1589-8/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSE-C.2017.156

to plan certification from the beginning and prevent them from rushing a product to the market.

In contrast, Common Criteria vendors often have finished products, and sometimes sales pending on a certificate. Due to long evaluation times, sometimes taking one year on average to complete, it is not uncommon that vendors ship product releases that succeed the ones under evaluation. Interviewee CC6 warns: "*I think the danger is that the certification efforts don't keep up with technology everyone's gonna get waivers and no one's gonna care*". One goal behind the recent push from the U.S scheme toward revising Common Criteria is to reduce certification time from years to months by replacing the huge amount of documentation required by well-specified test requirements (CC4, CC8).

Key insights: Long certification cycles are not realistic in many domains and both communities seem to generate different expectations and feelings of urgency for vendors. Should the need of documentation be minimized? Can documentation be replaced by mechanisms that are readable by humans but also amenable to automation?

B. Recertification

A common concern in our interviews were issues regarding recertification of products that are maintained over time, from small incremental changes to larger changes between releases. Both standards have provisions for recertification.

Both standards expect developers to perform an impact analysis, which includes adequately documenting the changes in the product being maintained. The degree of reuse of evidence is a result of negotiation between vendors and evaluators and it usually involves arguments about the *extent of the changes* and the *time passed since the product was evaluated*.

For minor changes in certified products, participants report "*reusing as much evidence as possible*" (CC1, DO3, CC9, CC10). Previously unmodified parts of the product are not reevaluated (DO4, DO5, CC5, DO7). For major changes, a complete recertification is expected with each release (CC1, CC6, CC7). Also grace periods exists with which a certification can be maintained despite smaller changes: "*products can be maintained for two years before they have to be re-evaluated*" (CC4). However, the situation is, at least, alarming when vulnerabilities are found in certified security products: "*If your product uses an affected version of OpenSSL and it has already been certified or validated, you are not required to recertify*" [23].

Beyond the reuse of evaluation artifacts, the *reuse of knowledge* across evaluations emerged as a topic in our interviews. Participants state that vendors tend to stick with testing laboratories and reuse their knowledge about the products and technologies used in it to reduce certification costs associated with learning curves (CC2, CC5). At the same time, recertification is a common sore issue for many developers; it often takes too long and requires too much work (CC6).

Key insights: Although provisions for recertification exists in both standards, practitioners are concerned with the costs and the duration of recertification. The current certification practices are incompatible with frequent releases that are increasingly adopted with the DevOps movement in large parts of the software industry.

C. Composition and Supply Chains

Similar to reuse over time in recertification, a sore topic in our interviews was reuse of libraries and other artifacts, broadly used in software development to reduce time and cost. Just as reduction of time for certification and recertification, support for composition is frequently discussed among practitioners and policymakers.

Although challenging, the composition of functionality is common in practice and supported by both standards at a coarse level of granularity—e.g., composition of application software with a certified database system and a certified operating system (CC2, CC5, DO7, CC6, [20]). As CC5 puts it: "*the developer is required to make a statement of compatibility and from it can be pretty much understood how the two products are connected, what functionalities the composite product uses from the platform.*"

At a more fine-grained composition with third-party dependencies (libraries, services), though, is more challenging. The use of external dependencies is influenced by the size and the criticality of the products to be certified. High assurance components tend do avoid dependencies (CC1, DO1, DO2), whereas dependencies are more common in small and low-assurance products where they are not certified (CC1, DO2). DO2 exemplifies that the entertainment system in an airplane may use an uncertified streaming library.

There is already an increased interest in having mechanisms to support fine-grained composition (CC8), such as the ones between cryptography standards and Common Criteria. However, there is still a lack of compositional assurance mechanisms to reduce the reliance on vendors to follow effective security engineering principles when composing commercial off-the-shelf products [9], [11], [13]. Current mechanisms are limited to testing known interactions between products.

Support for platforms, ecosystems, and app markets, where an end user may compose (possibly certified) artifacts from different sources are not in scope of either certification standard. Assuring correctness despite possible interactions among various combinations of extensions also seems incompatible with current testing-based strategies.

Key insights: We need to rethink how platforms/ecosystems should be certified. Are automated and compositional analyses beyond testing applicable to certification? How can systems with fast update cycles be certified? How to set up incentives and provide support to rich supply chains and more systematic reuse?

ACKNOWLEDGMENT

G. Ferreira is supported by a doctoral research grant from CAPES, Brazil. Grant n. 13713/13-2.

REFERENCES

[1] N. G. Leveson and C. S. Turner, "An Investigation of the Therac-25 Accidents," *Computer*, vol. 26, no. 7, pp. 18–41, 1993.

[2] M. Dowson, "The Ariane 5 Software Failure," *SIGSOFT Software Engineering Notes*, vol. 22, no. 2, p. 84, 1997.

[3] P. G. Neumann, "Risks to the public," *SIGSOFT Software Engineering Notes*, vol. 41, no. 5, pp. 19–26, 2016.

[4] *NHTSA. 2016. Federal Automated Vehicles Policy*. [Online]. Available: https://one.nhtsa.gov/nhtsa/av/av-policy.html.

[5] R. K. Yin, *Case Study Research: Design and Methods*, ser. Applied Social Research Methods. SAGE Publications, 2009.

[6] *Common Criteria for Information Technology Security Evaluation, Version 3.1 Revision 4*. [Online]. Available: http://www.commoncriteriaportal.org/cc/.

[7] *Common Methodology for Information Technology Security Evaluation, Version 3.1 Revision 3*. [Online]. Available: http://www.commoncriteriaportal.org/cc/.

[8] *RTCA. 1992. DO-178C–Software Considerations in Airborne Systems and Equipment Certification. Radio and Technical Commission for Aeronautics*.

[9] S. Lipner, "Criteria, Evaluation, and the International Environment: Where have We Been, Where Are We going?" *Information Security: Proc. International Conference on Information Security (IFIP/Sec)*, vol. 7, no. 1, pp. 5–13, 1991.

[10] S. Lipner, "Twenty Years of Evaluation Criteria and Commercial Technology," in *Proc. IEEE Symposium on Security and Privacy (IEEE S&P)*. IEEE Computer Society, 1999, pp. 111–112.

[11] J. Hearn, "Does the Common Criteria Paradigm Have a Future?" *IEEE Security Privacy*, vol. 2, no. 1, pp. 64–65, 2004.

[12] K. S. Shankar and H. Kurth, "Certifying Open Source – The Linux Experience," *IEEE Security and Privacy*, vol. 2, no. 6, pp. 28–33, 2004.

[13] F. Keblawi and D. Sullivan, "Applying the Common Criteria in Systems Engineering," *IEEE Security Privacy*, vol. 4, no. 2, pp. 50–55, 2006.

[14] T. Maibaum and A. Wassyng, "A Product-Focused Approach to Software Certification," *Computer*, vol. 41, no. 2, pp. 91–93, 2008.

[15] S. Lipner, "The Birth and Death of the Orange Book," *IEEE Annals of the History of Computing*, vol. 37, no. 2, pp. 19–31, 2015.

[16] M. Thomas, "Unsafe Standardization," *Computer*, vol. 40, no. 11, pp. 109–111, 2007.

[17] M. P. E. Heimdahl, "Safety and Software Intensive Systems: Challenges Old and New," in *Proc. Conf. on The Future of Software Engineering (FOSE)*. IEEE Computer Society, 2007, pp. 137–152.

[18] J. Rushby, "New challenges in Certification for Aircraft Software," in *Proc. Int'l Conference on Embedded Software (EMSOFT)*, 2011, pp. 211–218.

[19] I. Dodd and I. Habli, "Safety certification of airborne software: An empirical study," *Reliability Engineering and System Safety*, vol. 98, no. 1, pp. 7–23, 2012.

[20] J. Hatcliff, A. Wassyng, T. Kelly, C. Comar, and P. Jones, "Certifiably safe software-dependent systems: Challenges and directions," in *Proc. Conf. on The Future of Software Engineering (FOSE)*. ACM Press, 2014, pp. 182–200.

[21] M. Schreier, *Qualitative Content Analysis in Practice*. SAGE Publications, 2012.

[22] K. Charmaz, *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*, ser. Constructing grounded theory. SAGE Publications, 2006.

[23] *Corsec. 2014. Heartbleed and Your Security Certification (Blog post*. [Online]. Available: https://www.corsec.com/how-heartbleed-affects-your-security-certifications/