Robotics Software Engineering and Certification: Issues and Challenges

Roberto Pietrantuono and Stefano Russo
Università degli Studi di Napoli Federico II
Via Claudio 21, 80125 Napoli, Italy
{roberto.pietrantuono, stefano.russo}@unina.it

***Abstract*—As robots become increasingly intelligent and autonomous, spread well beyond the traditional area of industrial automation, and find many new critical applications – from robotics medicine to anthropic domains - we advocate the need for certification for robotics software. We discuss some relevant issues in robotics software engineering and certification, and outline some important challenges for the dependable software engineering community.**

***Keywords*-Software certification, Robotics, Standards**

I. Introduction

For decades, robots have spread mainly in the field of industrial automation; nowadays, robots are becoming ubiquitous and increasingly autonomous, and are more and more used in critical applications. Major examples are robotics surgery, medical and assistive robots, service robots, Industry 4.0.

Surgical robotics is a rapidly evolving field aiming to minimize the pain and risk associated with surgery, while increasing precision and likelihood of excellent surgical outcomes. The introduction of surgical robots into the operating room combines technological and clinical breakthroughs to improve the quality and outcome of the surgery.

Medical robotics is wider in scope, concerning devices used also for medical training, prosthetics, and assisting people with disabilities. Nowadays, robotic devices are used to deliver rehabilitation therapy to patients, and perform a growing number of other health-related tasks. Biologically inspired robots are becoming popular, giving raise to the field of assistive robotics. An *assistive robot* performs physical tasks for the well-being of a person with a disability. The task is embedded in the context of normal human activities of daily living and would otherwise have to be performed by an attendant.

Aerial and underwater robotics envisage a new generation of robots to support human beings in activities requiring the ability to interact actively and safely with environments not constrained on ground; an example is the inspection of buildings and infrastructures such as dams. In a broader view, *service robots* are foreseen to become a major part of the world robotics market in the upcoming decades, with applications in critical scenarios such as fire-fighting.

In *Industry 4.0*, the current fourth industrialization generation [1], human operators are foreseen to co-produce with robots and heavy machines, and "*the strength and speed of industrial robots are potentially lethal to human beings*" [2]. The Final Report of the Industrie 4.0 Working Group recognizes that "*safety and security are both critical to the success of smart manufacturing systems. It is important to ensure that production facilities and the products themselves do not pose a danger either to people or to the environment*" [3].

In all such emerging fields where robots go beyond the more traditional areas of manufacturing systems, and perform critical tasks or enter anthropic domains envisaging physical human-robot interaction, reliability, safety and security of robots operations are major concerns [4] [5] [6] [7]. This raises several challenges in robotics software engineering: in Section II we discuss the need for certification, and in Section III we identify some challenges at three levels: standardization; engineering processes; organizational, cultural and educational issues. Section IV provides some concluding remarks.

II. The need for certification

As intelligence is added to robot systems to achieve greater autonomy, and robots find unprecedented applications, their ability to work safely and securely in critical contexts becomes essential. The challenge is not just to build innovative robots, but ones which can be justifiably trusted. Quoting [8], while "*computing technologies are integral parts of any autonomous robotic system, they are often considered ancillary to their development*"; this is probably a reason for the little attention to robots' software engineering issues, and in particular to the one of providing assurance of their safe and secure operation.

Software certification is an indispensable factor for the spread of robots in critical application fields such as those outlined in Section I. The incredible diffusion of Artificial Intelligence (AI) in robotics gives to software a prominent role, and raises its complexity. This is likely to make the provisioning of evidences about correct operation of robots in scenarios where they may come into direct contact with (possibly untrained) humans and/or take autonomous decisions a much more difficult and expensive activity. We believe that the increasing sophistication and "weight" of software in future intelligent robots poses new and diverse challenges to their software engineering and certification.

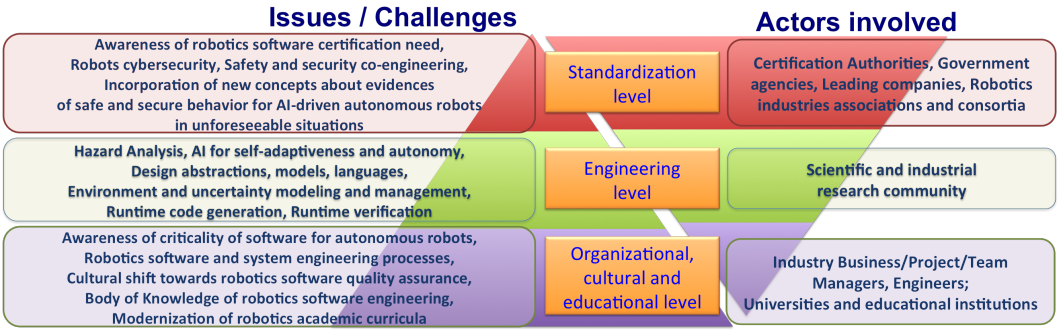

Fig. 1. Issues and challenges towards a culture of robotics software certification

In the following, we highlight some issues and challenges, that we envisage considering the current standards, the peculiarities of this domain as well as at the gap with respect to other domains, and by considering what practices from more mature sectors could be leveraged for robotics. Figure 1 shows the envisaged issues and actors, described in the next Section, which we refer to three levels: standardization, software engineering and cultural/organizational/educational aspects.

III. Toward robotics software engineering and certification

A. Standardization issues

Certification requires standards. As for *safety*, some existing standards are applicable to robot systems, but they contain generic (product) requirements; an example is the European Machinery Directive 2006/42/EG. The main software-related standard considered applicable to the robotics domain are:

- Part 3 of the well-known IEC/EN 61508 – which inspired various domain-specific standards - sets requirements for management, development and validation of code for the functional safety of electronic safety-related systems [9];
- IEC/EN 62061 for functional safety of safety-related *control* systems [10], which is the machinery-specific adaptation of the above IEC/EN 61508;
- ISO 12100 for risk assessment and reduction of safety of machinery [11];
- ISO 13849-1 for safety of control systems [12];
- ISO 10128 for safety requirements for industrial robots – specifically, for manipulators for industrial environment, part 1, and for robots systems and integration, part 2 [13];
- ISO/TS 15066 for collaborative robots [14], which applies only to industrial robot systems;
- ISO 13482 for safety requirements of *personal care robots*, a kind of service robots [15]; it concerns specifically three types of earthbound robots, namely mobile servant, physical assistant, and person carrier robots.

A further important standard for robot safety in factories is the ANSI/RIA R15.06 [16], the US adoption of ISO 10218.

Software safety issues in current standards appear insufficient for the emerging robotics fields described in Section I. Some standards, like ISO 12100, hardly apply to systems outside of their strict (industrial) target. Others, like EC/EN 61508, are process-oriented (i.e., they prescribe asks to perform to improve safety and formally analyze software) and may still be deemed applicable. Nevertheless, the software-specific issues of AI-driven robots in anthropic environments appear so far underestimated. The situation is different from other critical domains, where software safety assurance is known to pose issues remarkably different from mere development and test. In those sectors where specific standards do exist – like CENELEC EN 50128 [17], RCTA DO-178C [18] and ISO 26262 [19] for railway, airborne and automotive software systems, respectively - this has been well understood, and software certification is deemed essential. What appears to be necessary for future AI-driven autonomous robots are new concepts about the provisioning of evidences of correct behavior, different from the traditional ones of checklists of known hazards, or prescription of process activities to statically encompass all possible operating conditions.

(Cyber)security is "a lower and sometimes forgotten priority" for robots manufacturers [7]. Even in highly regulated sectors like railways and avionics, standards are still mainly focused on safety. However, robots operating in anthropic domains have to be more secure than other embedded systems. Robots exploiting online services (so that the term *cloud robotics* has been coined) will have vulnerabilities, but robotics standards do not to explicitly address cybersecurity. The tight relation between safety and security is today recognized, so the issue of their co-engineering is felt important, yet it is still a research area, and future standards are called to account for it.

A further aspect to consider is that robot manufacturers move toward adopting common "standardized" platforms. Probably the main such initiative is the Robot Operating System [20]. As suggested in [7], this gives the opportunity for industry consortia to publish "security and testing standards to be followed by robot application developers". More in general, leading companies and consortia may drive *de facto* robotics standards which will help to improve practices and to raise awareness of safety and security assurance.

Standardization is an essential stage along the road to software certification. The research community can contribute to build up a solid background, developing new techniques and providing evidence, through concrete applications, both of the need

for software certification in robotics and for possible solutions to specific problems. This is preparatory for standardization and industrial exploitation. Innovation-driven companies too are called to support efforts in this direction (e.g., through investments, collaborations with academy, by leading consortia and standardization groups). The aim should be to create awareness and concrete evidence that software certification is a must for robotics industry market. Indeed, developing standards for software certification is a long-lasting process requiring the involvement of companies, of certification authorities as well as of governmental agencies.

B. Software engineering issues

The culture of certification is the one of providing *evidences*. Providing a faithful assessment with an acceptable confidence is an as much important as challenging engineering task. Safety and security are system (not merely software) properties, which demand for evidences they are properly addressed throughout the whole manufacturing process, for achieving high levels of confidence in the proper system operation, and for meeting criteria of standards (both at process and product level). Software assurance is thus part of system assurance, but, as already perceived in other domains, it requires specific techniques and a specialized software engineering culture. *Robotics software engineering* is the emerging area – witnessed by initiatives like the IEEE Technical Committee on Software Engineering for Robotics and Automation - where the scientific and practitioners community can contribute toward a robotics software certification culture.

A peculiarity of robot systems, compared to other critical domains, is the role of the *environment* where they operate. The environment is not merely a *passive* area where a robot operates, but it becomes *active*, since it is directly engaged in accomplishing the robot's intended function. For instance, in the area of personal care robots, concepts like a "shared workspace" between robot and human are defined in the cited standard [15], where an intended contact is foreseen, which, of course, entails a risk related to autonomous decisions and actions of robots. The active role of the environment has a great impact on (critical) requirements elicitation (e.g., on hazard analysis), on design and modeling, and on V&V, which all will have to account for an *insufficient design-time knowledge* (i.e., *incompleteness* of requirements), for *highly dynamic contexts* (hence with *changing* requirements) and for *high non-determinism* (i.e., *uncertain* requirements). Some of the challenges regarding these engineering activities are:

- *Requirements/Hazards analysis*. In traditional critical domains, safety assurance means providing confidence that the system will operate correctly even under environmental conditions different from the nominal ones. As unintended conditions can seriously compromise the system, engineers strive to foresee and enumerate at design time all potential adverse situations (e.g., by hazard analysis). This approach has clear limitations because of the difficulties in capturing all the events of interest. It generally works when the environment is known and is quite stable at operational time. A "static" hazard analysis may be not enough in *active* environments. Design time specification of the conditions where the system will operate captures a small part of possible interactions with humans and environment. A serious challenge is how to gain knowledge about the surrounding environment *dynamically*, and how to "update" hazard analysis at runtime by the acquired knowledge so as to assure safety in operation. Like for autonomous vehicles, the explosion of AI within robotics raises key challenges as for assurances, because it is not sufficient to show that a robot can learn and adapt: it is required to provide pre-delivery *evidence* that the system will learn correctly and will not learn wrong behaviors leading to potentially dangerous actions/decisions.
- *Architecture and design*. While "*the production of software for robotic systems is often case-specific*" [21], it is increasingly designed according to Model Driven Engineering (MDE) practices, similarly to the embedded systems domain [22]. This is not surprising, because automation engineers have always been using models, especially for the design of control algorithms. However, the culture of providing evidences requires additional skills with respect to mastering good design principles and practices [23]. Differently from other domains, architecting software for autonomous robots demands for greater focus on *environment and uncertainty modeling*, as well as on *runtime modeling* (i.e., models that evolve at runtime) as *design principle*. Examples are context models and computation-independent models, used in critical domains [24], and runtime models developed for adaptive systems [25]. The usage of these models for safety assurance and certification of robot systems, and their possible adaptation/extension to make them able of supporting *evidence production*, needs to be explored. An initial and noticeable effort toward this direction is done by Schneider *et al.* [26], who target the issues coming with certifying *adaptive systems*.
  An important research issue is the extension of modeling languages with support for certification-oriented engineering activities. Examples are the construction of safety arguments or Failure Modes and Effect Analysis [27]. For instance, safety arguments notations should be extended to explicitly account for uncertainty, as well as for the role played by humans and the environment in interacting with robots.
  An additional issue arises for *cooperative robots*. These are systems in which more robots interact to achieve an intended goal. There is again a clear similarity with the automotive domain, if we consider platoons of autonomous vehicles interacting through vehicle-to-vehicle communication. Research is needed to adapt existing distributed software systems design techniques and patterns to cooperative robots. The challenge is not only to design properly interacting robots, but to provide evidences of their correctness in all operating conditions.

- *Implementation*. Besides the practices recommended by standards in other domains (use of certified compilers, avoidance of unsafe coding techniques such as use of pointers, dynamic objects, automatic type-conversion, etc.), a greater emphasis should be put on integrity checks and defensive programming because of dynamic environments. *Runtime* code generation (e.g., triggered by a model change at runtime), with different requirements in terms of space and time efficiency compared to design-time code generation, is a further issue to explore. This also indirectly implies a tool qualification process of runtime code generators, with related challenges. Specific programming abstractions and languages for robotics software represent a further area of investigation.
- *Verification and Validation (V&V)*. Software testing in critical domains is always a predominant cost factor, which might be amplified for critical autonomous robots. Specifying and executing test cases for checking intense human- and environment-robot interactions need new techniques. Deriving test cases by modeling the environment would definitely help (e.g., through a Computation-Independent Test model [28]), possibly complemented by stochastic reasoning (e.g., via Markovian and Bayesian models) to deal with uncertainty (for instance, by modeling a variable operational profile in terms of stimuli to a robot, and generating tests accordingly).
  Runtime testing (i.e., generating and running tests at operational time exploiting field data about new observed behaviors) and runtime verification techniques are further interesting research areas to investigate for robotics.
  Finally, the issue of testing cost becomes of paramount importance because of the huge additional cost that would come from adopting environment-aware techniques: a wide research area will regard test automation for robotic systems as well as simulation/emulation-based testing.

Overall, there is room to exploit existing principles, methodologies and techniques, e.g., used for adaptive or cooperative systems, for safety and security improvement of robots' software, by conceiving proper adaptation/extension or tailoring of sound methods. This indeed opens new research directions. But while developing new methods or borrowing existing ones is certainly good to increase robotics software quality (reliability, security, etc.), it may be not enough in view of certification. The challenge is that certification demands for *assurance methods* able to provide – before systems are deployed or put into market - evidences of correctness even under conditions unforeseen at design time. Considerable research effort is thus needed for the definition of engineering methods supporting such evidences for robots which will be used in our daily life and will be governed by artificial intelligence. In this sense, robotics may borrow advances which will be made available by other domains. A relevant example is the automotive sector, which is seeing a great increase of AI techniques too, facing similar problems as for co-operativity, environment-awareness and autonomy.

C. Organizational, cultural and educational issues

The cultural and organization level (see Figure 1) includes challenges that, even in presence of specific techniques for robotics software engineering, would hinder their industrial application. We believe there is the need for a much higher awareness of the relevance and criticality of software in autonomous robots, and hence of safety and security issues. This in turn needs to reflect on organizational processes tailored for robotics software production. We list some challenges:

- Adopting or tailoring practices (like the ones mentioned above) coming from other industrial sectors is not immediate. An example is Model Driven Engineering, used in the embedded systems domain [22]. It uses models as artifacts of the development activities, and derives outputs, such as program code, through automatic transformations [29]. Automation engineers use to prototype, simulate and test control algorithms with modeling tools such as MATLAB/Simulink, and then produce running code from models. Adopting state-of-the-art techniques and tools is however not enough for assurance and certification. Although modeling is recommended by safety standards, MDE techniques need to be carefully introduced in software processes, justifying their effectiveness, and assuring the compliance with certification standards. Most MDE approaches underestimate the problems of certifiable systems; this limits their application in critical domains. MDE of critical robots is much more than generating code of control algorithms from modeling tools; it requires proper skills and a re-organization of production processes (similarly to what discussed in [30] for the air traffic control systems domain), which may be hard where certification is in order.
- Besides using modeling and code generation tools, there is a trend in robotics software development to use Commercial Off-The-Shelf (COTS) or Free and Open Source Software (FOSS) technologies (like the ROS platform mentioned in Section III-A). Using software tools, COTS and FOSS poses well-known issues when software assurance and certification come into play, which demand for consciousness and for careful management of engineering processes – and it is known that the use of certified tools is not the panacea.
- Innovations in software practices are hard to introduce in manufacturing industries, partly because of the lack of specific culture by managers, who are often anchored to consolidated industrial processes where software is seen just as an intangible "add-on" to the concrete system, and tend to underestimate software issues.
- There is a cultural barrier to overcome in companies not originally born and organized for software development, as automation industries. The background of robotics practitioners is generally in mechanical and electrical engineering. The generic awareness about the criticality of robotics software needs to be supported by advanced software engineering knowledge and skills, integrated into

wider-scope certifiable systems engineering processes. Simply outsourcing software production is definitely not a solution for assurance and certification. There is the need for software quality assurance culture in robotics companies, and certification demands for professionals with qualifications - including the use of formally defined and managed software engineering processes - not currently in the background of most automation engineers. Industry-academia cooperation can help filling the gap.

- The cultural shift requires time and pass also through changes in academic robotics engineering curricula. Nowadays, software topics in university curricula even at the graduate level focus mainly on programming skills, and pay little attention to the aspects of software quality assurance. An important step would be the definition of a suitable body of knowledge to educate robotics engineers to deal with the advanced software assurance and certification challenges of future robotics systems.

IV. Conclusions

Thanks also to the increasing mutual influence of Artificial Intelligence and Robotics, in the next future robots are very likely to find application in many new fields other than traditional industrial automation, and to play an important role in our daily lives. We claim that this demands inevitably for robotics software certification, partially similarly to what is happening in the medical devices domain [31].

The software certification issue appears to have still very limited awareness in the robotics domain, yet it poses important challenges, which we believe invest standardization, engineering practices, as well as organizational, cultural and educational aspects. Addressing the outlined issues is by far not sufficient, but, indeed, robotics software certification cannot be achieved without coping with them.

Acknowledgement

This work has been supported by the PRIN 2015 research project "GAUSS" (2015KWREMX_002) funded by MIUR. We thank the anonymous reviewers for valuable comments.

References

[1] R. Drath and A. Horch, "Industrie 4.0: Hit or hype?" *IEEE Industrial Electronics Magazine*, vol. 8, no. 2, pp. 56–58, June 2014.

[2] L. G. Christiernin, S. Augustsson, and S. Christiernin, "Safety Critical Robot Programming and Testing for Operations in Industrial Co-Production," in *25th IEEE International Symposium on Software Reliability Engineering Workshops*. IEEE, Nov. 2014, pp. 29–32.

[3] H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0 – Securing the Future of German Manufacturing Industry," National Academy of Science and Engineering, Final Report of the Industrie 4.0 WG, April 2013.

[4] A. Albu-Schaeffer et al., "Physical Human-Robot Interaction in Anthropic Domains: Safety and Dependability," in *4th IARP/IEEE-RAS/EURON Workshop on Technical Challenges for Dependable Robots in Human Environments*, 2005.

[5] R. Alami et al., "Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006.

[6] A. De Santis and B. Siciliano, "Safety issues for human-robot cooperation in manufacturing systems," in *Tools and Perspectives in Virtual Manufacturing*, ser. VRTest, 2008.

[7] G. W. Clark, M. V. Doran, and T. R. Andel, "Cybersecurity issues in robotics," in *IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*. IEEE, March 2017.

[8] D. Brugali and B. A. MacDonald, "Editorial to the Special Issue on Robotic Computing," *Journal of Software Engineering for Robotics*, vol. 8, no. 1, pp. 1–2, Dec. 2017.

[9] ISO, "IEC/EN 61508-3 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements," 1998.

[10] IEC/EN, "IEC/EN 62061: Safety of machinery: Functional safety of electrical, electronic and programmable electronic control systems," 2005.

[11] ISO, "ISO 12100:2010 Safety of machinery – General principles for design – Risk assessment and risk reduction," 2010.

[12] ——, "ISO 13849-1:2015 Safety of machinery – Safety-related parts of control systems," 2015.

[13] ——, "ISO 10218-1:2011 Robots and robotic devices – Safety requirements for industrial robots, Part 1: Robots, and Part 2: Robot systems and integration," 2011.

[14] ——, "ISO/TS 15066:2016 Robots and robotic devices – Collaborative robots," 2016.

[15] ——, "ISO 13482:2014 Robots and robotic devices – Safety requirements for personal care robots," 2014.

[16] ANSI/RIA, "R15.06-2012 American National Standard for Industrial Robots and Robot Systems - Safety Requirements," 2013.

[17] E. C. for Electrotechnical Standardization, "EN 50128: Railway applications – Communications, signaling and processing systems – Software for railway control and protection systems," 2011.

[18] R. T. C. for Aeronautics (RTCA), "DO-178C Software Considerations in Airborne Systems and Equipment Certification," 2012.

[19] ISO, "ISO 26262-1:2011 Road vehicles – Functional safety," 2011.

[20] L. Garber, "Robot OS: A New Day for Robot Design," *Computer*, vol. 46, no. 12, Dec. 2013.

[21] F. Ciccozzi, D. Di Ruscio, and I. Malavolta, "Engineering the software of robotic systems," in *IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2017.

[22] G. Liebel, N. Marko, M. Tichy, A. Leitner, and J. Hansson, "Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice," *Software & Systems Modeling*, vol. 17, no. 1, pp. 91–113, Feb. 2018.

[23] S. Russo and F. Scippacercola, "Model-based software engineering and certification: Some open issues," in *IEEE International Symposium on Software Reliability Engineering Workshops*, Oct. 2016, pp. 237–240.

[24] M. Sabetzadeh, S. Nejati, L. Briand, and A. H. E. Mills, "Using SysML for Modeling of Safety-Critical Software-Hardware Interfaces: Guidelines and Industry Experience," in *IEEE 13th International Symposium on High-Assurance Systems Engineering*, Nov. 2011, pp. 193–201.

[25] H. Giese et al., "Living with Uncertainty in the Age of Runtime Models," in *Models@run.time: Foundations, Applications, and Roadmaps*, ser. LNCS, vol. 8378. Springer, 2014, pp. 47–100.

[26] D. Schneider and M. Trapp, "Conditional Safety Certification of Open Adaptive Systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 8, no. 2, July 2013.

[27] F. Scippacercola, R. Pietrantuono, S. Russo, A. Esper, and N. Silva, "Integrating FMEA in a Model-Driven Methodology," in *DASIA 2016 - Data Systems In Aerospace*, ser. ESA-SP, vol. 736, 2016.

[28] F. Scippacercola, R. Pietrantuono, S. Russo, and A. Zentai, "Model-in-the-Loop Testing of a Railway Interlocking System," in *Model-Driven Engineering and Software Development*, ser. Communications in Computer and Information Science, vol. 580. Springer, 2015, pp. 375–389.

[29] J. Whittle, J. Hutchinson, and M. Rouncefield, "The state of practice in model-driven engineering," *IEEE Software*, vol. 31, no. 3, pp. 79–85, May 2014.

[30] G. Carrozza, M. Faella, F. Fucci, R. Pietrantuono, and S. Russo, "Engineering Air Traffic Control Systems with a Model-Driven Approach," *IEEE Software*, vol. 30, no. 3, pp. 42–48, May 2013.

[31] J. H. Weber and M. Price, "Certification of Medical Information Systems: Towards a Foundational Framework and Methodology," in *27th IEEE International Symposium on Software Reliability Engineering Workshops*. IEEE, Nov. 2016, pp. 241–248.