Should Robotics Engineering Education Include Software Engineering Education?

Milda Zizyte*
milda@brown.edu
Computer Science Department
Brown University
Providence, Rhode Island, USA

Trenton Tabor*
ttabor@andrew.cmu.edu
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

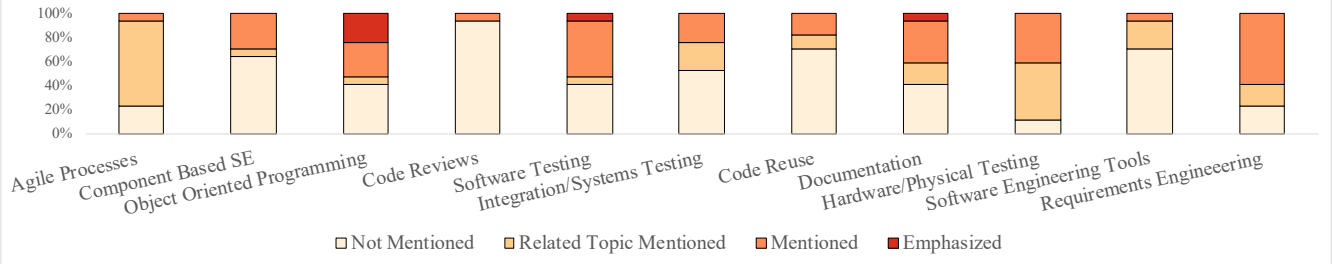

Figure 1: Summary of prevalence of Robotics Software Engineering Practices [8] in Robotics Bachelors of Science Programs.

ABSTRACT

Multiple universities across the United States now offer bachelor's degrees in robotics, which aim to prepare students to work in the robotics industry. To judge how well these programs are providing software engineering training, we evaluate whether these programs teach the software engineering practices that are required for robotics software engineering. We compile an updated list of robotics bachelor's degree programs and measure whether the curriculum of each program claims to teach a specific practice. We find that some of these practices are not mentioned in the curricula, and that some are only taught implicitly in long-term project courses. These project courses vary in scope, guidance, and structure. This implies that robotics bachelor's degrees may not be preparing students to engage with the practices in the workforce.

CCS CONCEPTS

• **Computer systems organization** → *Robotics*; • **Social and professional topics** → **Software engineering education**.

KEYWORDS

Robotics Education, Robotic Software Engineering, University Curriculum

*Both authors contributed equally to this research.

RoSE'22, May 9, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9317-1/22/05.
https://doi.org/10.1145/3526071.3527514

ACM Reference Format:
Milda Zizyte and Trenton Tabor. 2022. Should Robotics Engineering Education Include Software Engineering Education?. In *4th International Workshop on Robotics Software Engineering (RoSE'22), May 9, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3526071.3527514

1 INTRODUCTION

In 2007, the first "Robotics" Bachelors of Science was launched in the US [9]. After 15 years, by our count, there are at least 16 universities in the US offering a robotics degree. These programs claim to prepare students for industry, research, or academia after graduation. Developing robots in the real world does not simply require technical and theoretical skills, but the ability to apply engineering (including software) practices.

As researchers and practitioners in Robotics Software Engineering (RSE), we were curious how these programs chose to prioritize and emphasize the Software Engineering (SE) concepts students needed in robotics industry. Generally "Engineering" and "Engineering Technology" programs are ABET-accredited for a "general engineering" category and are given flexibility to emphasize or teach different topics depending on institution[1]. Contrast these with the SE degree, for which IEEE and ACM release curriculum guidelines every 10 years [1].

Previous work, by Garcia, et al., has studied which RSE practices are central to robotics, finding trends in common practices, such as using Agile methodologies and implementing object-oriented code [8]. We sought to evaluate the degree to which existing RSE programs actually teach these practices. To do so, we developed an up-to-date list of robotics programs and then compared a previously published list of RSE practices to the publicly available curricula for each program. We find that many of these programs omit formal training in these common practices, which may lead to underprepared robotic software engineers.

[1]Carnegie Mellon University's "Robotics Major" does not publicly list an accreditation

2 METHOD

To examine how much focus Robotics BS programs put into each of Garcia's practices [8], we developed a process of counting mentions in course descriptions. These course descriptions are often publicly available and short, providing an efficient window into the topics for a degree. We built a sufficient corpus of descriptions, by performing a census of Robotics BS programs offered in the USA.

2.1 University programs studied

We chose our study universities by performing an independent search followed by reviewing the literature. We only considered USA-based programs. Initially, we performed a web search for Robotics BS programs. Some programs were in our top results while others appeared in aggregator lists. We found that our web search was able to find more programs if the query included the acronym "ABET", an accreditation board responsible for certifying "Robotics Engineering" programs. After completing our search, we compared our list of Bachelor of Science Robotics programs to a less recent survey [6], providing two more programs. For the Engineering Technology programs in our study, we chose to include them alongside our ten "Robotics Engineering" and one "Robotics" program, since industry often considers them equivalent [10]. [2]. The programs included in our analysis are summarized in Table 1.

2.2 SE Concepts Used in Robotics Development

We based our list of common SE practices in robotics on Garcia et al. [8], who conducted 18 semi-structured interviews of robotics experts and also collected 156 responses to an online survey. While they were also able to gather data on both the perception of the difference between SE and RSE and the challenges faced in RSE, we direct your attention to their **RQ1**: "*What practices are applied in SE for service robots?*". This question was examined in both the survey and interviews; by comparing both, the authors were able to discover that practitioners were likely using different terminology for the same activities. The survey is available and uses SE-specific terms; e.g. the options for "Which of these software engineering processes do you apply in your projects?" were:

- Waterfall
- Hybrid (e.g., V-Model, Spiral)
- Agile (e.g., SCRUM, Extreme Programming)
- Other (please specify below)

These options, while they contain examples, do not contain explanations. Many RSE practitioners may lack formal training in these terminologies and techniques.

From Garcia's work, we selected the SE practices that were reported to be used by over half of the industry practitioners surveyed. While Garcia focused on Service Robotics, our experience leads us to believe that the practices studied are applicable to all RSE.

2.3 Evaluation of coursework

Two authors of this work independently reviewed the coursework of each program to evaluate to what extent the SE practice was mentioned in course descriptions.

[2]We excluded robotics BS programs that did not include a major programming component. These programs were from Central Connecticut State University, Gannon University, and Pennsylvania College of Technology.

Table 1: Bachelor of Science programs analyzed.

Major	University
Robotics Engineering	Lawrence Technological University
Robotics Engineering	Worcester Polytechnic Institute
Robotics Engineering	University of Detroit Mercy
Robotics Engineering	Lake Superior State University
Robotics Engineering	University of Michigan
Robotics Engineering	University of Hartford
Robotics Engineering	Arizona State University
Robotics Engineering	Widener University
Robotics Engineering	UC Santa Cruz
Robotics Engineering	Miami University
Mechatronics and Robotics Engineering	Trine University
Mechatronics and Robotics Engineering	Southern Illinois University Edwardsville
Robotics and Control Engineering	United States Naval Academy
Robotics Engineering Technology	Purdue University
Robotics & Controls Systems Technology	Millersville University
Robotics and Manufacturing Engineering Technology	Rochester Institute of Technology
Robotics (Second major only)	Carnegie Mellon University

Our categories were:

Not Mentioned - no mention of concept or any related concept in any course required by the major.

Related Topic Mentioned - concept in question was not mentioned, but we determined that a related topic was, in at least one course description. For example, if they discussed hardware debugging, we considered that to be related to *Hardware/Physical Testing*.

Mentioned - concept is mentioned by name in at least one course description, with some exceptions described below.

Emphasized - concept is highlighted in more than one course description, or the course description makes clear that the concept is the primary focus of the course.

Because course syllabi were often not publicly available for these courses, we used only public-facing course descriptions when performing our analysis. Every school provided such course descriptions in a relatively consistent format and level of detail.

To ensure that this methodology would be sufficient for recognizing SE practices if they appeared in a program, we individually examined 6 SE programs with the same criteria. With the exception of the *Hardware/Physical Testing*, in this calibration we found all of these programs mentioned or often emphasized these practices.

Disagreements of classification were resolved using deliberation. We ignored courses that were optional or may not be taken by all students going through a program.

In some situations, we refined our classification criteria based on an agreed-upon understanding of SE concepts and/or curricula. These criteria were:

- If a course had a long-term, structured team project component, we assigned "related topic mentioned" to the evaluation of Agile Processes. This was based on the idea that, throughout a semester or longer, a team would perform some sort of team-based iterative management and design towards a final product.
- If a course description uses "modular" as a term or includes education about the Robotics Operating System (ROS), we assigned "mentioned" to the evaluation of Component-Based SE. ROS abstracts robot behavior into separate processes or components that communicate via specific message-passing interfaces, and therefore requires thinking about software development in terms of components.
- If a course had a requirement for writing reports for a long-term, structured project, we assigned "mentioned" to the evaluation of Documentation. If this description went in-depth with the sorts of materials expected in these reports, such as safety considerations and evaluations of alternatives, we assigned "emphasized."
- If a course mentions finding customer/industry needs as part of conceptualizing a project, we assigned "mentioned" to the evaluation of Requirements Engineering, as we considered this synonymous with requirements elicitation.

A capstone or capstone-style course is a dedicated design or project course that involves a long-running group project taken from conception to implementation. These courses often had "capstone" in the title, but we also included others that fit our definition. For such courses, we measure the length (in terms) of the capstone-style course, whether the course involves a final presentation to an outside audience, and what terms come up most frequently in the course descriptions, in aggregate.

3 RESULTS AND DISCUSSION

We discovered several patterns in the program curricula, including which practices are taught and which are overlooked, and how capstone-style courses are an opportunity for structured SE learning.

3.1 SE practices taught in robotics programs

We summarize our results in Figure 1, showing an anonymized distribution of the emphasis given to each of Garcia's practices in each program. As evidenced in the figure, there was a wide variety in coverage for these topics. Most programs had at least some mention of *Object Oriented Programming*, while no program mentioned *Code Reviews*. Of our study programs, only one required an explicit course in SE, but all of them mentioned at least one of these practices in another course.

3.2 Under-emphasized Practices

We noticed several practices that were not explicitly mentioned in the descriptions in our study. In particular, *Agile Processes*, *Code Reuse*, and the QA practices of *Integration/Systems Testing*, and *Code Reviews*, were not included by name. It is possible that some of these were overlooked in our review, due to terminology mismatch. Garcia found that many practitioners were performing *Requirements Engineering* but claimed not to be. We similarly found that many programs taught the concepts of *Requirements Engineering* without ever using the phrase.

We are concerned about the omission of these practices, since students expect programs to reflect the activities and skills needed to succeed in the workplace. If they're not emphasized in the course descriptions, students may discount their importance.

3.2.1 Agile Processes. In Section 2.3, we describe counting any long term structured project as a "related topics mentioned" for Agile practices. While one description mentioned "cyclic design iteration" directly, most programs do not describe how they encourage students to structure their own engineering practice. Garcia's survey showed a clear preference for an Agile development structure. Recent quantitative research demonstrates that utilizing a formal SCRUM structure can help students develop these skills [11]. However, we can not tell from course descriptions if students are required or even encouraged to follow any such process.

3.2.2 Reuse. Garcia identified three important bottlenecks to further reuse of code. The first is the code itself, including a number of interface issues. Then, there is a lack of documentation for reusable components. Finally there are licensing issues from the available tools from the community. However, there is still significant reuse in spite of these difficulties.

Recognizing these issues, it seems that there is a need in robotics software education to prepare students to overcome and prevent these issues. We found very little discussion of reuse in the studied course descriptions. In Figure 1, we show that no programs emphasized and only two programs even mentioned code reuse. Those two mentioned reusable objects, but there was no mention of design for reusable libraries or frameworks. It does not appear that students are, for example, being exposed to designing idioms and patterns for whole libraries, which may perpetuate the issues in industry. An added benefit to consciously designing for reusable libraries and frameworks is that the design documents are themselves useful, widening the second bottleneck as well.

None of the course descriptions we examined discussed analyzing licenses in the SE process, although these issues are discussed in capstone courses, as students are integrating a larger system from many parts. We also didn't see any mention of the common engineering decision of build vs. buy for custom hardware or software components, which can have a large impact on a project scope [7].

3.2.3 Quality Assurance Practice. According to Garcia, the two most common quality assurance practices in RSE are Integration Tests and Code Reviews. Again, these practices were not emphasized in the examined course descriptions, but may be taught or learned independently by students or experientially learned over the duration of another course. If these practices are not covered, however, this mismatch may be critical. We found a lot of mention and emphasis in courses on software design, but much less on demonstrating that the software does what it was designed to do. While it can be difficult to build a lesson around code review, we're excited to see new techniques in this area [2].

3.3 Capstone-style courses

All but one of the programs required a capstone-style course. Most capstones were two terms long, one was one semester long but a follow-on to a professional design course, one was two or three semesters long depending on if a student did a senior project, and one had three 2-semester sequences spread out over three years.

3.3.1 Audience. Capstone courses all involved presenting the final project in some structured way. In terms of audience,

(1) one school required a capstone presentation made to a panel of engineering faculty;
(2) three schools had capstone-style courses that specifically mentioned industry representatives at final presentations;
(3) one school had a capstone that was sometimes done with industry partnership, depending on the project;
(4) one capstone sequence included a co-op practicum, where students worked part-time off-campus in an industry job; and
(5) the other ten schools that offered a capstone-style course did not indicate a specific audience for capstone presentations.

We found that several schools used the capstone-style courses to expose students to industry partners. The prevalence of industry partners for these specific courses indicates that they are a way that schools facilitate an interface between students and industry. Students have an chance to demonstrate their engineering skills and get direct feedback from industry.

3.3.2 Capstones as SE practicum. One commonality that we noticed across programs was that much of the coverage of agile, documentation, requirements engineering, and testing practices in the curricula we studied came from capstone or capstone-style courses.

We also noticed some common themes in their curricula. In particular, these courses start with identification of needs, then have students go through a process of specification, design, prototyping, and final product. Documentation is often emphasized. When evaluating common language across the descriptions of these courses by looking at the most frequent words across all descriptions, we found mentions of the following categories:

- Teamwork ("teams", "management")
- Forms of communication ("presentation," "report," "proposal")
- Design ("design," "planning," "constraints")
- Implementation ("development," "prototype")
- Evaluation ("testing," "performance," "validation")
- Technical skills ("robotics," "knowledge")

This indicates that students are learning and applying SE in their capstone-style courses. What we cannot determine from course descriptions alone is whether they are given frameworks for learning these concepts, or if they are tasked to figure out the concepts by "learning by doing." Because these courses vary in scope, guidance, and structure, it is hard to judge how much SE students are being taught deliberately.

3.3.3 Capstone opportunities. Work in SE education suggests that a capstone course is an opportune place for students to build "soft skills," as they are more confident in their technical skills by that point [3]. For robotics, we see an opportunity to emphasize SE education in these programs by injecting more structured SE frameworks in these courses or their prerequisites. If students get the chance to interface with industry in their capstone-style course, this is also an opportunity to get real-world feedback on the applicability of the practices taught.

4 RELATED WORK

Five years ago, a subset of the programs we studied were benchmarked for general concepts covered, faculty affiliation, and accreditation [6]. That benchmark also examined other degree and minor programs. Recently, a survey was conducted of *Software Development* skills (among hardware, professional, etc. skills) [4]. Those skills include some overlap with the practices we examined; for instance, "Do agile program design" is a subset of the *Agile Processes* we examined. Generally, however, these practices are orthogonal to their skills. There has also been recent analysis in how closely general SE education aligns with industry trends [5], which found similar, but smaller gaps between education and practice.

5 FUTURE WORK (FOR OUR COMMUNITY)

From these results, it is clear that the RSE research community needs to engage with robotics educators. We need to help them understand the value of formal SE training for their students. This will require, on top of research in best practice, research into pedagogy and advocacy for the importance of our work.

ACKNOWLEDGMENTS

Thanks to our study programs for publishing their descriptions.

REFERENCES

[1] Mark Ardis, David Budgen, Gregory W Hislop, Jeff Offutt, Mark Sebern, and Willem Visser. 2015. SE 2014: Curriculum guidelines for undergraduate degree programs in software engineering. *Computer* 48, 11 (2015), 106–109.
[2] Barış Ardiç, Irem Yurdakul, and Eray Tüzün. 2020. Creation of a Serious Game for Teaching Code Review: An Experience Report. In *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE T)*. 1–5. https://doi.org/10.1109/CSEET49119.2020.9206173
[3] María Cecilia Bastarrica, Daniel Perovich, and Maira Marques Samary. 2017. What can students get from a software engineering capstone course?. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. IEEE, 137–145.
[4] Carlotta A. Berry, Michael A. Gennert, and Rebecca Marie Reck. 2020. Practical Skills for Students in Mechatronics and Robotics Education. *ASEE annual conference exposition proceedings* (2020). https://par.nsf.gov/biblio/10184531
[5] Orges Cico, Letizia Jaccheri, Anh Nguyen-Duc, and He Zhang. 2021. Exploring the intersection between software industry and Software Engineering education-A systematic mapping of Software Engineering Trends. *Journal of Systems and Software* 172 (2021), 110736.
[6] Joel M Esposito. 2017. The state of robotics education: Proposed goals for positively transforming robotics education at postsecondary institutions. *IEEE Robotics & Automation Magazine* 24, 3 (2017), 157–164.
[7] Kim Fowler. 2004. Build versus buy. *IEEE Instrumentation & Measurement Magazine* 7, 3 (2004), 67–73.
[8] Sergio García, Daniel Strüber, Davide Brugali, Thorsten Berger, and Patrizio Pelliccione. 2020. Robotics software engineering: A perspective from the service robotics domain. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 593–604.
[9] Michael A Gennert and Craig B Putnam. 2018. Robotics as an Undergraduate Major: 10 Years' Experience. In *2018 ASEE Annual Conference & Exposition*.
[10] Ronald E Land. 2012. Engineering technologists are engineers. *Journal of Engineering Technology* 29, 1 (2012), 32.
[11] Christoph Matthies, Johannes Huegle, Tobias Dürschmid, and Ralf Teusner. 2019. Attitudes, beliefs, and development data concerning agile software development practices. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 158–169.