An experiment to build an open source application for the Internet of Things as part of a software engineering course

Conference Paper · February 2024

DOI: 10.1109/SERP4IoT59158.2023.00008

2 authors:

Rareș Cristea
University of Bucharest
7 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Ciprian Paduraru
University of Bucharest
50 PUBLICATIONS **157** CITATIONS
SEE PROFILE

An experiment to build an open source application for the Internet of Things as part of a software engineering course

1st Rareș Cristea
Department of Computer Science
ICUB Research Institute of the University of Bucharest
University of Bucharest, Romania
rares.cristea@unibuc.ro

2nd Ciprian Păduraru
Department of Computer Science
University of Bucharest, Romania
ciprian.paduraru@unibuc.ro

***Abstract*—Software engineering education is a field in which innovation is constantly taking place, especially in terms of the methods used to achieve the competencies and learning objectives to be taught. The Internet of Things is a relatively new area that offers accessible but complex SE challenges that the authors believe are an excellent example for inclusion in the educational process. Over the course of two academic years, we have explored the integration between SE and the Internet of Things with the additional goal of building an open-source application repository to foster academic research and enhance industrial tools.**

***Index Terms*—software, engineering, education, Internet of Things, IoT, open source**

I. Introduction

Software engineering education (SEE) is an integrative set of competencies that are part of the skill set of any professional software developer. Software engineering (SE) skills are necessary to ensure the quality of the resulting software and to enable collaborative, planned, and effective development. Several strategies have been developed to facilitate the acquisition of these skills, and most involve a group application development project. The Internet of Things (IoT) is a rapidly growing paradigm for distributed systems. It is characterized by a swarm of interconnected devices with low computational power that focus on providing a limited number of functions that improve an existing device and generally provide data that enables better decision making. In both academic and industrial initiatives, a variety of device architectures and protocols make for a highly heterogeneous environment. The lack of a ”clear path forward” can be challenging for developers tasked with finding the best solutions to specific problems. The additional challenge presented by the current IoT landscape can be turned into a useful educational tool. Navigating through a real-world challenge alleviates the need to create an abstract or ”curated” learning environment and improves the applicability of the knowledge acquired [1].

A. Background - The Software Engineering Course

The informatics study program at the University of Bucharest has included compulsory courses in SE since the original design of the program. Students in this program have an initial course on software development methods in the second year, which focuses on teaching the tools needed to implement a successful software development life cycle (SDLC). The sequence of the SDLC isn't strictly enforced, but the emphasis is on understanding and being able to use at least individual artifacts, such as: planning tools, documentation software, Git, issue trackers, CI/CD pipelines, testing tools, etc. The mandatory SE course in the third and final year of the program focuses on the students' ability to correctly implement the SDLC phases. This course is about the ability to orchestrate and develop the use of the previously learned tools in a collaborative and efficient manner. During three years previous to the ones included in this experiment, this course accepted a broad range of project subjects, which students were required to ideate, and bring arguments towards the relevance their project has relative to the course criteria. These projects ended up being mostly web development projects.

B. Motivation for IoT

One of the primary goals of the SE course (SE) is to provide students with the analytical skills necessary to break down a customer's needs into software requirements, develop the solution, and deliver it. While generally all SE courses involve the collaborative development of a software product, SE instructors use different approaches in selecting the product to be developed. Some courses involve real-world stakeholders in the process of negotiating the final requirements for the product [2] [3], [4] which provides the most realistic experience, but also requires more resources to be gathered in the form of real-world stakeholders. Another approach is to allow students to determine their own project, but make sure that the requirements and work they want to do match the expectations of the course [5]. Another option is to propose a topic to which the project should relate and allow students to develop a well-defined product proposal on that topic.

In choosing the latter approach, we proposed IoT applications as the topic for the projects for several reasons: a) the course coordinators and assistants had academic experience with the topic, which made them reliable stakeholders; b) The relative unfamiliarity of the students with the technological particularities of IoT applications, enables the instructor to assess the student's ability to get familiar with a previously unknown topic.

In this paper, we present the experience of organizing an IoT-focused SE course in two different academic years. We experimented with enriching an undergraduate SE course with technical knowledge about the Internet of Things. Our assumption is that the specific IoT challenges promote the accumulation of SE knowledge and prove that this is a good basis for further teaching SE knowledge through this method.

The main objective of this study is to identify the degree to which infusing specific technical knowledge in a project based SE course yields improvement of the specific knowledge, without sacrificing on the SE knowledge. Our research question is ***Does the addition of Internet of Things-specific software design requirements render a better SE course***?

II. Methods

Software Engineering courses have been widely used as fertile ground for empirical experimentation with innovative educational techniques. These courses ought to focus on the process of developing the right and correct software artefact, and on the collaborative aspect of software development. [6]. Generally SE courses center around a project based evaluation method, in which teams of developers have to develop a software application given the courses constrains. [7]. Improvements in this educational process is often achieved by experimenting with a variation in an established SE course. Previous research heavily focused on the employment of agile practices in the course project [8] [5] [9], and in particular on the interfacing between the developers and the rest of the stakeholders [10] [3] [1]. Some attention has been also given to improving the software product that outpus as part of the project. Surveys include [11]'s aim to evaluate commit metrics of the projects, and [12] survey of the tools chosen by the students. Lesser studies focused on the software product that is finally developed. Our study uses the previously proven research method of experimentation in improving SE educational courses, focuses it on the lesser explored area of the software product developed and particularly uses Internet of Things thematic as a litmus test for the approach.

A. Course Setup

The students had 12 weeks to develop an IoT application in 6 sprints. This course's expected workload was 60 hours per student, with teams consisting between 4-5 members. Their course defined goal was to create the software that would run an IoT device given its limited hardware computing capabilities. The instructor's initial assumption was that the apps would need to be developed in C/C++ to be as close as possible to the IoT paradigm [13]. However, this assumption was challenged after the first iteration of the experiment, and the second course also allowed the use of Python, a more familiar, and accessible programming language.

B. Project Assessment

The evaluation of the project was divided into two parts:

In the first part, students had to focus on the technical aspects of the apps they developed. For example, in the 2020 course, they had to implement the app in C/C++, they had to use at least the HTTP communication protocol and, for additional points, MQTT; the app had to include at least basic unit and integration tests that should provide ¿ 80% coverage, and provide JSON documentation of the API, which they represented with documentation designed in [14]. For the 2021 course, we built upon and improved upon the 2020 course evaluation criteria by also adding Python as an option for the languages in which the app could be developed, and replacing the requirement to use our own design of communication protocol with the use of OpenAPI and AsyncAPI for HTTP and MQTT, respectively.

The second part of the requirements focused on the SDLC phases other than development. For example, before starting the development of the app, the students had to create a software development plan. This included defining a set of requirements for the IoT app that they could obtain themselves or from other stakeholders, identifying a set of functional and non-functional requirements that they believed would be met, planning the development of the app over the six sprints, and creating developer and user documentation or a usage guide. Students were provided with templates for documentation to deliver. These elements are relatively common in all types of software projects, so no major changes other than slight improvements to the templates were required for the 2021 course.

III. Resulting application set

Students had to develop IoT apps considering various design criteria. The criteria that were critical for inclusion in the final app were as follows:

- Create or retrieve data from an external source
- Process the data and generate new information
- Provide an HTTP API
- Deploy an MQTT API
- Document API using a predefined JSON specification (or the OpenAPI, AsyncAPI specifications for the 2021-22 course)

In the first year, 10 of the 54 student teams managed to meet all of the listed design criteria. Given the surprisingly large number of projects, we selected 5 of the best projects each year to include in the app materials. The selection was based on the evaluation criteria of the S course, which wasn't one of the decisive criteria for the app documents. The same principles were applied to the second iteration of the course, with details described in Table I. The number of projects remained relatively constant, as both years were aimed at the same group of students. The lower number of "valid apps" doesn't necessarily equate to lower quality of the apps developed. The use of OpenAPI and AsyncAPI in the second year allowed us to better filter the best candidates. We were

TABLE I
APPLICATION SET DEVELOPMENT YEARS

Years	Application Set		
	No. of projects	***Valid apps***	***Selected apps***
2020-2021	54	10	FlowerPower, Smarteeth, SmartKettle, SmartTV, WindWow
2021-2022	56	8	SeraSmart, SmartOven, Soundsystem, VacuumCleaner

able to include the same number of apps in the set in the second year. The final app set was made publicly available on GitHub [1]. It consists of 10 mock IoT apps, a hub app that connects to one or more apps depending on user configuration, and setup instructions for local and dockerized use.

Because of the opportunity to incorporate this project into a personal professional portfolio, some of the teams also contributed features to their projects that were outside the scope of the planned development. For example, one team developed an advanced graphical interface, and some teams chose to also deploy their app appropriate hardware device. On the one hand, this initiative demonstrates the variety of project goals that can be set in a course on IoT. On the other hand, it may point to the need to better define the field of IoT and determine the core learning objectives and appropriate competencies for this area of knowledge.

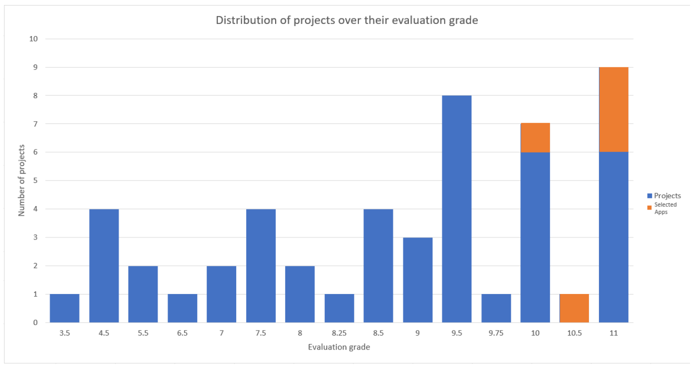

Fig. 1. The distribution of the 2021-2022 projects grades highlighting (with orange) the projects that were included as part of the final application set. The result also outlines that there is room for improving and growing the number of viable candidate applications.

IV. DISCUSSION

The expectations of the experiment were that students would gain a better practical understanding of SE practices and gain insight into a topic not yet adequately covered in the undergraduate curriculum. One of the expected externalities of the courses was that the student projects would eventually help create an open-source app that would have real-life apps - initially in academia and, after further enhancements, in industry.

[1]https://github.com/unibuc-cs/IoT-application-set

A. Technical challenge

Since we work with devices that have low computational power, we realised as an immediate consequence that the software developed should be focused on using the most efficient programming languages available. Our decision for the 2020 course was that all IoT projects must be built in C/C++. The success of this decision was rather mixed.

1) The use of C/C++ allowed students to explore the languages further. They already had introductory courses that focused on this language, but given the requirements of the IoT project, a greater focus on multithreading, dependency management, communication interfaces, etc.was needed.
2) C/C++ is considered a more difficult language to manage precisely because it allows flexibility such as memory management.
3) While C/C++ would be a better choice for introductory courses [15], higher level languages are generally preferred in this context because they can provide functionality faster.

B. Integration

One challenge in building open source software is to agree on similar software design principles. Our approach to this challenge was to encapsulate the device functionality. The apps were treated as gray-box software to minimize both current and future app customization needs and to ensure integration with the app set. Minimal to no changes were made to the source code of the app that was delivered as part of the project. The focus of our integration efforts was on the interface between the apps. The use of a custom JSON interface and later OpenAPI and AsyncAPI proved successful and warranted minimal code adjustments to the API definitions. 18 apps were declared valid in the sense that they satisfied almost completely the inclusion requirements, out of which 10 were integrated and tested withing the final set. 10 apps were selected to be included in the open-source app set in order to allow gradual introduction of further complexity.

C. Open Source Project

In [16], it was observed that aligning students' work with an open source software (OSS) project increased students' confidence in their technical and communication skills. Qualitative feedback from students confirmed that working on a collaborative OSS project increased their motivation. Students have a strong tendency to create a portfolio of software projects to which they have contributed during their studies. They believe that this portfolio provides an advantage in later employment in industry. By contributing to a joint OSS project, they feel that their portfolio is not only enhanced by an additional project, but also by a project that is a joint contribution to a larger project, and that the success of the overall app package leads to a better appreciation of their own contribution. Further studies can be made on the impact of this type of projects and the resume-driven motivation that students might have [17].

D. Agile Challenge

Our course implemented the agile process, as do most SE courses. The two reasons for choosing this process are: a) The process is guided by deadlines that can be accommodated with the time frame of the course; b) It is easier to implement and less bureaucratic for the instructor, and it is less formal, more adaptable, which helps students manage their own schedule [4]. Teaching with the agile process has its prerequisites [18]: on the one hand, implementing agile processes requires that stakeholders value agile processes. Usually, students are open to this approach, but at the time of development, they may not have the necessary know-how on how to implement the process correctly. Therefore, further guidance from instructors is needed to keep Agile "on track." It is noted that students who were aware and asked for recurring feedback were better able to implement the Agile process. On the other hand, the SE curriculum of their program does not allow students to learn about other development processes and presents the perspective of Agile as a "silver bullet." Startups are reported to have difficulty implementing the Agile methodology and its variants, opting to select elements from different methodologies [19].

V. Conclusions

Integrating two seemingly distant areas of computer science into a single course can prove challenging, and in attempting to be successful in both areas, knowledge areas of one of the topics may not be properly addressed. Our course continued to focus on SE as the primary source of knowledge, but provided enough context on the Internet of Things to encourage students to explore the topic further. Focusing the course projects on a single type of app,didn't hinder the SE content, and delivered students with knowledge they may have otherwise haven't interacted with. Therefore it can be concluded that the addition of IoT requirements was a net positive for this SE course.

Acknowledgement

This research was supported by European Union's Horizon Europe research and innovation programme under grant agreement no. 101070455, project DYNABIC.

References

[1] M. Shaw, J. Herbsleb, and I. Ozkaya, "Deciding what to design: closing a gap in software engineering education," in *Proceedings of the 27th international conference on Software engineering*, ser. ICSE '05. New York, NY, USA: Association for Computing Machinery, May 2005, pp. 607–608. [Online]. Available: https://doi.org/10.1145/1062455.1062563

[2] M. Marques, S. F. Ochoa, M. C. Bastarrica, and F. J. Gutierrez, "Enhancing the Student Learning Experience in Software Engineering Project Courses," *IEEE Transactions on Education*, vol. 61, no. 1, pp. 63–73, Feb. 2018, conference Name: IEEE Transactions on Education.

[3] J. Vanhanen, T. O. A. Lehtinen, and C. Lassenius, "Teaching real-world software engineering through a capstone project course with industrial customers," in *2012 First International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex)*, Jun. 2012, pp. 29–32.

[4] M. R. Marques, A. Quispe, and S. F. Ochoa, "A systematic mapping study on practical approaches to teaching software engineering," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, Oct. 2014, pp. 1–8, iSSN: 2377-634X.

[5] A. Bollin, E. Reçi, C. Szabó, V. Szabóová, and R. Siebenhofer, "Applying a maturity model during a software engineering course – How planning and task-solving processes influence the course performance," *Journal of Systems and Software*, vol. 144, pp. 397–408, Oct. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121218301407

[6] CC2020 Task Force, *Computing Curricula 2020: Paradigms for Global Computing Education*. New York, NY, USA: ACM, Nov. 2020. [Online]. Available: https://dl.acm.org/doi/book/10.1145/3467967

[7] T. Clear, S. Beecham, J. Barr, M. Daniels, R. McDermott, M. Oudshoorn, A. Savickaite, and J. Noll, "Challenges and Recommendations for the Design and Conduct of Global Software Engineering Courses: A Systematic Review," in *Proceedings of the 2015 ITiCSE on Working Group Reports*, ser. ITICSE-WGR '15. New York, NY, USA: Association for Computing Machinery, Jul. 2015, pp. 1–39. [Online]. Available: https://doi.org/10.1145/2858796.2858797

[8] M. Polmear, E. Volpe, D. R. Simmons, N. Clegorne, and D. Weisenfeld, "Leveraging faculty knowledge, experience, and training for leadership education in engineering undergraduate curricula," *European Journal of Engineering Education*, vol. 0, no. 0, pp. 1–20, Feb. 2022, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/03043797.2022.2043243. [Online]. Available: https://doi.org/10.1080/03043797.2022.2043243

[9] J. Khakurel and J. Porras, "The Effect of Real-World Capstone Project in an Acquisition of Soft Skills among Software Engineering Students," in *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE T)*, Nov. 2020, pp. 1–9, iSSN: 2377-570X.

[10] B. Bruegge, S. Krusche, and L. Alperowitz, "Software Engineering Project Courses with Industrial Clients," *ACM Transactions on Computing Education*, vol. 15, no. 4, pp. 17:1–17:31, Dec. 2015. [Online]. Available: https://doi.org/10.1145/2732155

[11] S. Hamer, C. Quesada-López, A. Martínez, and M. Jenkins, "Measuring Students Source Code Quality in Software Development Projects Through Commit-Impact Analysis," in *Information Technology and Systems*, Rocha, C. Ferrás, P. C. López-López, and T. Guarda, Eds. Cham: Springer International Publishing, 2021, pp. 100–109.

[12] R. C. Palacios, C. Casado-Lumbreras, T. Samuelsen, and X. Larrucea, "Students' selection of teamwork tools in software engineering education: Lessons learned," p. 11.

[13] S. M. Alnaeli, M. Sarnowski, M. S. Aman, A. Abdelgawad, and K. Yelamarthi, "Vulnerable C/C++ code usage in IoT software systems," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Dec. 2016, pp. 348–352.

[14] C. Păduraru, R. Cristea, and E. Stăniloiu, "RiverIoT - a Framework Proposal for Fuzzing IoT Applications," in *2021 IEEE/ACM 3rd International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT)*, Jun. 2021, pp. 52–58.

[15] N. Alzahrani, F. Vahid, A. Edgcomb, K. Nguyen, and R. Lysecky, "Python Versus C++: An Analysis of Student Struggle on Small Coding Exercises in Introductory Programming Courses," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: Association for Computing Machinery, Feb. 2018, pp. 86–91. [Online]. Available: https://doi.org/10.1145/3159450.3160586

[16] J. Pereira, "Leveraging Final Degree Projects for Open Source Software Contributions," *Electronics*, vol. 10, no. 10, p. 1181, Jan. 2021, number: 10 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2079-9292/10/10/1181

[17] J. Fritzsch, M. Wyrich, J. Bogner, and S. Wagner, "Résumé-Driven Development: A Definition and Empirical Characterization," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, May 2021, pp. 19–28.

[18] V. Devedžić and S. R. Milenkovic, "Teaching Agile Software Development: A Case Study," *IEEE Transactions on Education*, vol. 54, no. 2, pp. 273–278, May 2011, conference Name: IEEE Transactions on Education.

[19] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, "Software development in startup companies: A systematic mapping study," *Information and Software Technology*, vol. 56, no. 10, pp. 1200–1218, Oct. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584914000950