Teaching ROS efficiently to mixed skill classes
A blended learning approach with gamification elements

Staehle Benjamin and Ertel Wolfgang

Institute for Artificial Intelligence University of Applied Sciences
Ravensburg-Weingarten, Germany staehle@hs-weingarten.de

Abstract. In this work in progress report, we illustrate how to efficiently teach a mixed skill class the foundations of ROS within one semester in a single course. The goal is to equip students with the basic knowledge and tools to join our Robocup@Home team, start a scientific project or thesis. To achieve this in a mixed skill setting we combine blended learning with gamification elements supported by a code versioning system. We believe that this approach is not only the most efficient way to teach this kind of matter but also bridges the gap between academic and industry working concepts.

1 Introduction

Developing Service Robots for over a decade, we experienced how frustrating it can be for students having to deal with multiple barriers before being able to start working with their actual field of interest. Many students came to our lab with big visions but then had to spend months understanding the depending software frameworks and setting up their development environment. We noticed that such tasks can consume a lot of the initial motivation which can even lead to a dropout. This also reflects back on researchers and employees of the university as they have to invest a serious amount of time explaining and assisting novice students. In order to leverage this problem and to increase efficiency inside our institute, we decided to start a practical oriented course to create a balanced foundation for students interested in robotics. This also gives us the opportunity to include students from non-pure computer-science fields such as electrical engineering and mechatronics whose curriculum usually covers merely basic education in complex software development.

1.1 Robotic Frameworks

[HC11] describe the most relevant toolkits and frameworks for robotic development. The diversity in the late 2000s made it difficult to build applications upon other research group's software. In many cases, this led to the development of isolated solutions resulting in redundant applications even among different departments inside the same university. Established frameworks like ROS[1], their

[1] Robot Operating System http://www.ros.org/

vivid communities and broad support of robotic hardware have improved the collaboration and communication among robotic research groups drastically over the past years. Yet the complexity of hardware and software on robot systems remains extraordinary challenging but the foundation laid down by ROS makes entering the world of robotics much easier and therefore interesting for teaching.

1.2 Teaching Concepts

The vision of this work is motivating students to transform their already acquired, but mostly theoretical, knowledge into solutions for real-world problems. Service Robots are an ideal platform for this transition as they incorporate aspects of various study fields and therefore can be used to address a broad audience. Already evolved courses like [Yim+08], [CWC13] and [Cap13] state that the most effective and fun way to acquire this kind of knowledge is a practical approach that involves interaction and experimenting along learning the theoretical concepts. Classic lectures, that separate theory from practical exercises, cannot meet this requirement as each student has to pass an individual point of understanding which cannot be forced from outside. This requires an asynchronous teaching method as proposed in [Kel+06]. Methods like flipped classroom likewise demand that the student can freely choose when to learn and at which speed. As the name indicates the method flips the lecture to the students free time and homework or exercises to the original class session. Still, the method emphasizes a strict separation between teaching and exercising [Chr16]. Blended learning, however, transforms lecture times into hybrid teach and exercise sessions in which students can choose how to approach their current problems while benefiting from group discussions in a workshop-like atmosphere. In this approach, the teacher is no longer the singular source of knowledge but a mentor to whom the students can directly address specific questions. A very important component in this rather loose way of teaching is the strict definition of deadlines and a consequent reaction if they are not respected. Another way to achieve this is the usage of gamification elements such as live exercise rankings during the sessions to trigger a competitive motivation among the students. Also, the course can be separated into major parts that act similar to an achieved goal in a computer game. In this work, we combine the mentioned concepts into a practically oriented course which we describe in the following section.

2 Method

2.1 Code Versioning and Issue Based Working

Being familiar with a code versioning system such as git and platforms that are building functions around it is an essential skill for any software developer these days. Popular platforms for code versioning are github[2], bitbucket[3] or source-

[2] https://github.com
[3] https://bitbucket.org

forge[4]. We still experience that sometimes even high semester students are not familiar with these tools. Our approach is centered around a university-hosted version of gitlab[5] but there are no functions used that an online platform such as github could not easily replace. A core functionality of such platforms aside from supporting the development process are issue trackers. In general, these mechanisms are used for bug reports or feature requests. They can have rich descriptions including code samples or images and have a comment function to discuss specific topics in the scope of the current issue. We use these mechanisms to distribute course materials and exercises. Additionally, it is an effective communication channel to our students as they can ask questions individually and in relation to their current exercise. Each repository also contains an own wiki area which the students can use for personal notes. The built-in CI[6] pipelines, similar to the popular jenkins[7] and travis[8] used in many open-source projects, evaluate the work in progress of the students at each commit and provide instant feedback that is visualized in the web frontend as shown in Fig. 1a.

2.2 Course Overview

Our prototype lecture *Introduction to Autonomous Mobile Robots* is a 5 ECTS[9] course with two lab sessions per week. It is separated into four tiers with a rising degree of difficulty. Also, it is a mixed skill course that masters and bachelor students of different study fields can attend in parallel. The exercises and course materials cover the same topics but masters have more challenging exercises and also have to dig deeper into the theoretical background than bachelors. Each session is started with a 10min wrap up of the current progress stating how many students have already reached which level and how the overhaul class performs. All statistics are anonymized to protect the student's privacy.

Tier1. The entry part of the course aims to balance the basic skills among the participants necessary to work in a robotic environment. This includes an introduction to the Linux shell, git code versioning and Python. For this purpose we use the well-known, free online learning platform codecademy[10] which provides comprehensible tutorials at a beginner level and live code evaluation. In parallel, the students get an introduction to our gitlab platform and their tier1 repository. Here they have to solve additional exercises covering relevant topics from codeacademy and with extra tips and tricks for daily use which are not part of the online-tutorials. Students that claim to have sufficient skills are allowed to skip the codecademy courses and only solve the gitlab exercises but in return have to work more autonomously than other students in this tier.

[4] https://sourceforge.net
[5] https://about.gitlab.com
[6] Continous Integration
[7] https://jenkins.io
[8] https://travis-ci.org/
[9] European Credit Transfer and Accumulation System for students
[10] https://www.codecademy.com

Tier2. The next level consists of plain python exercises and explains the fundamentals of a robotic system such as actuators and sensors. This tier's primary goal is to familiarize students with the gitlab platform and python. All course materials and exercises are organized inside the gitlab platform from this point on. The students have to do basic calculations on a given set of simulated laser data as seen in Fig. 1b and are introduced to creating own tests for the CI engine.

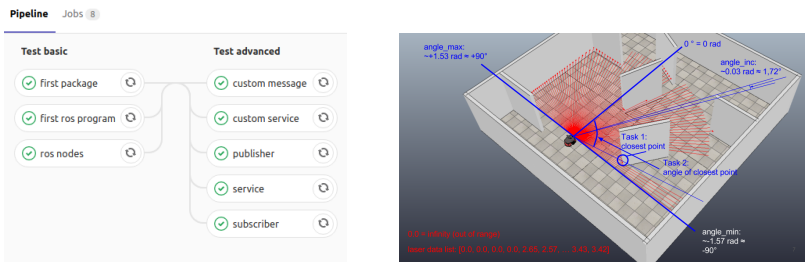
Tier3. In this chapter, the students learn about the most important ROS concepts such as topics, services, actions and helper tools. Beforehand the students are introduced to a virtualized development environment. We provide virtualbox[11], docker[12] and kvm[13] images from which the students can choose based on their personal preference. All exercises can be conducted inside this environment using gazebo[14], a popular simulation framework. In this environment, the students only have to do minor adjustments such as generating ssh keys to start working. The degree of difficulty rises constantly while progressing to encourage the students to discuss the materials rather than just solving the tasks. This tier ends with a mini project where the students have to navigate a turtlebot robot through a gap in a wall. The project is evaluated individually and acts as a midterm exam.

Tier4. The course ends with the so-called maze challenge in which the students get a laser-equipped turtlebot and a training maze which they can use to test their algorithms. Before, we reflect common mistakes that occurred during the tier3 mini project and point out what could be improved. After this, the students get a clean repository which they have to organize and document on their own. The final grading does not only depend on the performance of their robot in the challenge, but also on the quality of code and development process (e.g. using issues, commit messages, tests). The best students of this course are given the opportunity to become a member of the universities RoboCup@Home[15] Team alongside with project and thesis offerings.

2.3 Comparison to Online Platforms

Flipped and blended learning approaches strongly depend on online teaching resources and exercise frameworks. Before starting to run an own infrastructure which has to be set up and maintained it is advisable to check the currently available online platforms. When we started the course 2016 we enrolled our students at the *robotIgniteAcademy*[16] offered by the company *The Construct*.

[11] https://www.virtualbox.org
[12] https://www.docker.com
[13] https://www.linux-kvm.org
[14] http://gazebosim.org
[15] http://www.robocupathome.org
[16] http://www.theconstructsim.com



(a) (b)

Fig. 1: CI result and Python laser exercise

Starting from scratch this is one of the most comfortable ways to set up a class. Besides the great tutorials, the web-based coding and simulation environment is the key benefit of this approach, as students only need a web browser to work. Nevertheless, we decided to run our own setup in the end due to flexibility and cost reduction reasons. Also, we experienced connection issues from time to time and other small problems that tarnished the experience for our students. In Table 1 we compare the two approaches. The results only reflect our personal experience and may depend on external factors, such as connection speed, which the provider cannot influence.

Table 1: Comparison of online platform and self hosted solution

Criteria	**Online platform**	**Self hosted**
Setup effort	++	-
Maintenance effort	++	o
Flexibility (e.g. self defined exercises and challenges)	-	++
Costs	- -	+
Reliability	o	+

++ very good, + good, o neutral, - bad, - - very bad

3 Conclusion and Outlook

At the time of writing this work is still in progress and will be evaluated at the end of 2018. Nevertheless, we can already state that those students which have completed this course during the last semesters were able to achieve a homogeneous skill level in the context of ROS basics regardless of their background. This is also reflected by the composition of our current RoboCup@Home Team where the amount of mechatronic and computer-science students is equally distributed.

Currently, we are investigating evaluation instruments such as [CWC13] to document and compare the learning quality among multiple classes and study fields. Also, we are continuing to improve the instant feedback mechanisms and want to apply the proposed methods in other lectures to continuously enhance the learning quality at our university. Further, we plan to establish follow up classes that are focused on specialized topics of robotics such as manipulation, object recognition or navigation.

4 Acknowledgements

The Authors would like to thank Martin Preussentanz, Jochen Weissenrieder, Joerg Wendorff, Benjamin Kathan, Christopher Bonenberger and Markus Schneider for the inspiring discussions. In addition, we want to thank Sashidhar Reddy Kanuboddi, Simon Bucher and Igor Chernov for supporting the lecture as tutors and coworkers. Special thanks go out to Steffen Pfiffner for helping to run this course in the start phase. This work was conducted within a practically oriented curriculum development program named WILLE[17] founded by the MWK[18] State Department of Baden-Wuerttemberg Germany.

References

[Cap13] D. J. Cappelleri. "A Novel Lab and Project-Based Learning Introductory Robotics Course". In: *IEEE Transactions on Education* 56.1 (2013), pp. 73 –81.

[CWC13] N. Correll, R. Wing, and D. Coleman. "A one-year introductory robotics curriculum for computer science upperclassmen". In: *IEEE Transactions on Education* 56.1 (2013), pp. 54–60. ISSN: 00189359. DOI: 10.1109/TE.2012.2220774.

[HC11] A. Harris and J. M. Conrad. "Survey of popular robotics simulators, frameworks, and toolkits". In: *Conference Proceedings - IEEE SOUTHEASTCON*. IEEE, 2011, pp. 243–249. ISBN: 9781612847399. DOI: 10.1109/SECON.2011.5752942. arXiv: 0412052 [cs]. URL: http://ieeexplore.ieee.org/document/5752942/.

[Kel+06] J. O. Kelly et al. "A non-prescriptive approach to teaching programming". In: *ACM SIGCSE Bulletin* 38.3 (2006), pp. 217–221.

[Yim+08] M. Yim et al. "A practice-integrated undergraduate curriculum in mechanical engineering". In: *ASEE Annual Conference and Exposition, Conference Proceedings* (2008). ISSN: 21535965.

[Chr16] Christopher Pappas. *Blended Learning vs Flipped Learning: Can You Tell The Difference? - eLearning Industry*. 2016. URL: https://elearningindustry.com/blended-learning-vs-flipped-learning-can-tell-difference (visited on 02/19/2018).

[17] Wissenschaft lernen und lehren, trans: learning and teaching science

[18] Ministerium fuer Wissenschaft, Forschung und Kunst, trans: state department for science, research and art