Reflection of the ECER Junior Botball Workshops

Muscalu Mihai-Cosmin
Department of Business Informatics
HTBLuVA Dornbirn
Dornbirn, Austria
muscalu.mihai@student.htldornbirn.at**

Fink Daniel
Department of Business Informatics
HTBLuVA Dornbirn
Dornbirn, Austria
daniel.fink@student.htldornbirn.at

Gantioler Nate
Department of Business Informatics
HTBLuVA Dornbirn
Dornbirn, Austria
nate.gantioler@student.htldornbirn.at

Ramsdorfer Leonard
Department of Business Informatics
HTBLuVA Dornbirn
Dornbirn, Austria
leonard.ramsdorfer@student.htldornbirn.at

Immanuel Ehe
Department of Business Informatics
HTBLuVA Dornbirn
Dornbirn, Austria
immanuel.ehe@student.htldornbirn.at

** *Email of the Author*

***Abstract* — Working in a chaotic environment is not a desirable way of programming or of communicating different features with colleagues. One way of solving this issue is by implementing Git-Flow into projects and sharing the code with project partners while describing features that can be implemented and that are planned to be added in the future to HTL's robots. The results of this research paper show that Git-Flow can help save time and facilitate management of code.**

I. Introduction

We at the HTL Dornbirn have learned that inefficient planning leads to a chaotic and hectic environment, so we have developed systems that help us order and communicate ideas for a better and more concrete way of communication. A planned course of action leads to an efficient and faster method of programming and implementation. With the implementation of Git-Flow into the team's coding environment we can reach higher goals, faster communication of ideas and faster implementation of features that will help us in the future of this project. Most IT companies have some form of communication implemented into their system to reach deadlines faster, so we here at HTL Dornbirn have learnt multiple systems that facilitate this. Systems that can help and assist HTL's students as well as others in sharing ideas in a team environment can be of service to companies.

This paper will analyze the efficiency of Git-Flow and its implementation into this project as a result. Furthermore, it will evaluate the way in which it was implemented and how it was used in this project.

II. How can one communicate clearly?

A. Definition and Methods

Communication is one of the most important aspects in a team and it can lead to either a chaotic and messy environment or a project done with time to spare. Like Nat Turner said: "Good communication is the bridge between confusion and clarity,", thus reinforcing that communication should be a high priority in every team and project, if one wants to accomplish their goals in a reasonable time.

Communication is defined as the ability to transfer one's information to another in order to produce greater understanding of a subject. The product of miscommunication is often the source of frustration-within a team and can lead to conflicts that will inevitably waste precious time in the scope of the project's timeline. The most effective way one can communicate is by first listening to and understanding their colleagues actively. Through the act of active listening one can better grasp the problems within the team and solve them with the appropriate solution instead of implementing unnecessary methods into the system. The act of actively listening by itself is an important step to improve one's communication skills. After listening, conveying one's message clearly and concisely is the next step of solving or implementing a feature effectively. Not using filler words and avoiding excessive speaking is the best way to broach the subject and suggest ideas. In this section this paper is going to evaluate and research the importance of communication within a team. ([4], n.d.)

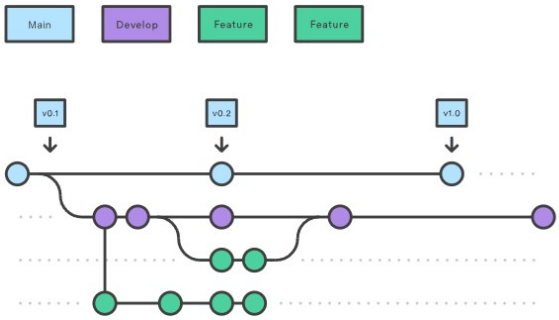
B. Importance within a team environment

With the rise of digitalization and automation, communicating ideas clearly is becoming more of a problem in the modern world. In a 2016 LinkedIn survey, communication was seen as one of the most important skills when working within a team environment and was extremely valuable to future employers. ([1], n.d.)

The team's way of leveraging communication potential is by implementing Git-Flow into the workflow of the team members and analyzing its benefits and average use by this team's members. The paper's goal is to not only reduce the programming time but to also improve the team's skills for the future to gain a more detailed understanding of communication and how it can be used effectively.

III. Concept and Structure

The concept and structure of Git-Flow is based on using GitHub along with VS Code to create a more ordered environment by separating different code and code versions into multiple branches. For example, this team can program code for an idea yet to be approved just to be able to test and demonstrate it but only do so on the "Features" branch instead of "Main" or "Implementation", like you can see in Error: Reference source not found This allows us to be able to separate and identify what the code should be used for, and which is of more importance during the testing or implementation phases of this project. This provides us with a clear and concise way of programming and also a better understanding of the teammate's intentions with their



code.

A. Structure and use of Git-Flow

By using Git-Flow this team is able to create a centralized workflow and a streamlined repository. This allows faster and more accurate communication of intentions between the team and avoids conflicts and confusion that can only lead to an unfinished product. The team will be using Git-Flow by writing the necessary code in VS Code, where Git-Flow will be present, and after it is finished, uploading it into the robot program. If any changes occur during the testing of the program, it is to be copied and then pasted into VS Code in order to be able to be accessed at a later date and viewed by the entire team. The team implemented this feature because the majority of the time schedules did not align and so the team was not able to work together or share ideas with colleagues.

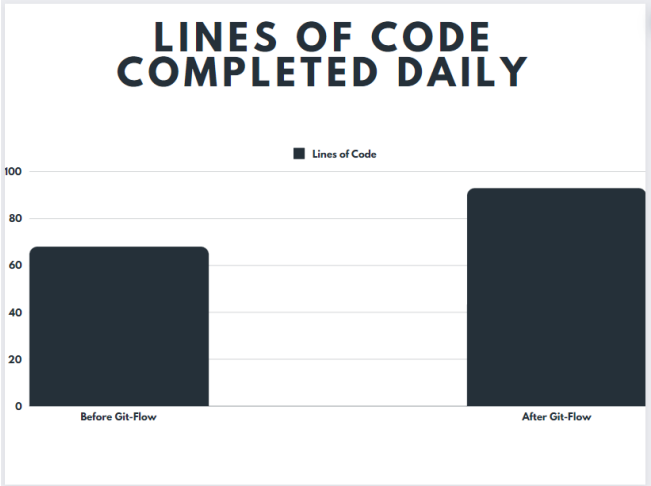
In the first session of using Git-Flow this paper will analyze how much time it takes for the team to get used to this method of code management. After that this paper will also analyze how much it improved the team's efficiency and how much time was saved by using this method over others. These will be analyzed in detail, and this paper will be reviewing and reassessing the effectiveness of this method daily by asking colleagues directly or by looking at the evolution of the code and how it was influenced by this method.

IV. Results

In this section one can read about the results of our implementation of Git-Flow and how it has affected the working environment along with what benefits it had in this project. Here one can also find the results of the daily analyzation and whether it has benefited this team in the long run or it did not yield the expected results.

A. First days of the implementation

In the first days of the implementation of Git-Flow most of our programming time had actually slowed down, but not considerably. Eventually the speed at which the team completed tasks was faster than before the implementation as you can see in Error: Reference source not found. The team's members, even still being new to this programming language, almost managed to reach the average daily lines of code that a professional writes. ([3],

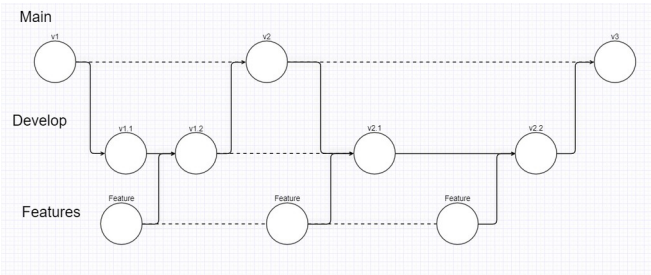


n.d.)

B. Benefits

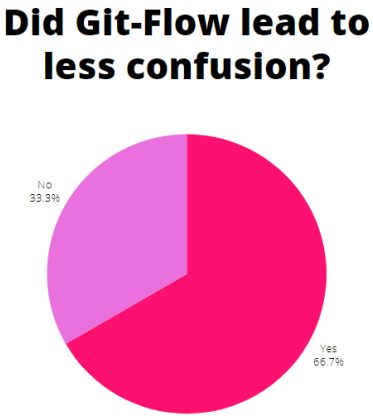
Through research this paper has found that the speed of this team had improved because the management and communication were facilitated with the use of Git-Flow. The member could look at the code that a colleague had programmed and not make the mistake of programming the same feature or a similar one. The improvement of communication resulted in the mitigation of duplicates withing the team.

The benefits of Git-Flow were not immediately clear, but in the long run it was useful. In Error: Reference source



not found you can see an example of the project structure.

As one can see, this project structure was simple to understand, and it is easier to plan for the future when understanding the current situation. This team's members reported less confusion and better understanding of their colleagues' ideas in contrast to teams that did not use Git-Flow as you can see in the survey in Error: Reference source not found.



The members could communicate their ideas and even their colleagues' ideas at a deeper level than before. It also improved relations between the members, as there were almost no conflicts in the team. Conclusion

Communication is one of the most important aspects in programming work and without it conflicts and confusion will thrive in your team environment. Git-Flow helped mitigate this issue but did not rid it entirely. Our planning withing Git-Flow is what actually cemented better communication within our team and what made working together an easier task than ever before. Git-Flow helped us greatly to implement some features that we are excited to present at the upcoming contest.

V. Acknowledgements

This paper would like to acknowledge Manuel Mayer for his guidance with the writing of this research paper. This team would also like to thank our professor Mr. Lenz Norbert for his guidance and help during this project.

VI. Works Cited

[1]. (n.d.). *LinkedIn survery; Top 10 most in demand soft skills*. Retrieved from LinkedIn: https://www.linkedin.com/business/talent/blog/talent-strategy/most-indemand-soft-skills

[2]. (n.d.). *Git Flow exaple*. Retrieved from www.atlassian.com: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

[3]. (n.d.). Retrieved from Medium: https://medium.com/modern-stack/how-much-computer-code-has-been-written-c8c03100f459

[4]. (n.d.). Retrieved from corporatefinanceinstitute: https://corporatefinanceinstitute.com/resources/management/communication/

[5]. (n.d.). Retrieved from Atlassian: https://www.atlassian.com/git/tutorials/comparing-workflows