Pair Programming Matcher Tool: Final Report

CS 3704: Intermediate Software Design

Prepared by
Caroline Joseph
Mason Gelletly
Quinn Anderson
Tatiana Monteiro

Epic Fellows

11/29/2023

1 ABSTRACT

One of the most important aspects of software development is its team. However, there are no efficient ways to form a team or to pair engineers. In a field where collaboration and continuous learning are crucial, an effective team-building tool is necessary. The Pair Programming Matcher's goal is to dynamically find a bridge between these objectives. The main objective of this tool is to employ a clever algorithm that matches software engineers for code reviews and pair programming sessions. It can also be used to build well-rounded and effective teams for software development projects. The tool will consider individual skill sets, project involvement history, current workload, and even past collaboration history while ensuring that the generated pairings are not only diverse but also beneficial to both parties. This will make sure that members of the teams are learning from each other, there are fewer code silos, and the overall code quality increases. Beyond matching and team building, the tool will serve as a hub for feedback by allowing participants to share their experiences and preferences for future matches. This adaptable and responsive system will be an essential tool in not just companies but also classrooms. Our goal is to refine the way teams collaborate, learn, and ultimately create an asset for building superior software products.

2 INTRODUCTION

Software Engineering involves a lot of collaboration in order to develop and maintain efficient and working software. Many aspects go into a successful team, like communication, teamwork, productivity, and learning. Pair programming is a method for Software Engineers to improve their code while working collaboratively, thus being able to receive direct feedback and tips when it comes to coding [1]. However, it can be difficult for programmers to find the correct person to pair with, which could be someone who uses the same editor, has similar coding styles, and can give constructive feedback in a manner they are comfortable with. Our proposed solution would be to create a matching tool that would take in input preferences from the user and generate an ideal and realistic matching for them. Furthermore, this tool can also be used to match teams together. For example, if a team of 3-4 software engineers wanted to pair with another team in order to discuss and compare code, this tool would be able to match them with a team that fits their requirements.

3 RELATED WORK

In the software engineering field, there have been related projects and processes to the Pair Programming Matcher Project. In regards to new hires and internships, many companies have adopted programs where they pair these new hires or interns with mentors. Furthermore, interns work with teams of experienced professionals. Companies in software engineering utilize teams

to solve issues and they match these teams in order to complete the task in the most efficient manner.
One example in the software engineering field of this is formation. Formation is a tool for engineers to get highly customized training, and land a dream role in a specific company [2]. The technology matches people with highly trained engineers, in hopes that these engineers can teach the person how to succeed in the field. The training platform is AI allowing for improvement in skills and better matching with these professionals.
Another similar program and one related to the potential Pair Programming Matcher is Google's CS Research Mentorship Program. This program matches students usually from lower-income places with mentors in Google to guide them and help them learn more computer science skills [3]. This program is very helpful as it can help these students gain skills through peer-to-peer networking, career mentorship, and displaying different pathways in the field. Overall there is related work to the Pair Programming Matcher, which can inspire the project to include certain elements. Furthermore, there are parts of both of these programs and other projects, that are not included, that the Pair Programming Matcher can include in order to be more efficient and useful.

4 APPROACH

High-Level Design Decisions:

We will be using a combination of Client-Server to handle the distributed nature of our application and a Model-View-Controller (MVC) to structure the application's internal design. This will allow our system to balance scalability, maintainability, and a clear separation of crises, which are critical for a robust and efficient team-building tool like our Pair-Programming Matcher

Our system requires a robust backend to handle the complex logic of matching engineers, processing feedback, managing user profiles, and more. A client-server architecture will help us achieve this separation, where we have the server handle the data processing and storage, while the client will provide an interactive and user-friendly interface. This architecture will make it easier to achieve scalability because the server can handle multiple requests from many clients simultaneously. This is important as we wanted our tool to be used by both companies and universities, which means it will need to potentially support a large number of concurrent users.

This architecture will make it easier to manage and update the application by dividing it into three interconnected components. This will allow us to develop well-structured and maintainable code, as well as parallel development of the user interface and the business logic. The Controller layer will handle user requests and update the Model. The Model is where our data and logic will be handled. This layer will manage our user profiles, skill sets, workload, project involvement, and things they've collaborated on. The View is where the UI/UX interaction will be

handled. It will represent what the users interact with such as how to input skills, and preferences and give feedback, as well as being updated by the model.

Implementation Process:

We have elected to utilize the Waterfall model as the process for our Pair Programming Matcher project. This model represents the best for our project as it allows our relatively small team to gather and come up with a plan upfront, empowering the group to split and have members individually contribute to the project on their own terms. In the event that a conflict arises with the agreed-upon design, then it will be a simple conversation with the other members as to whether we need to shift back to the design stage. However, due to the nature of this process, this potential situation should never become a reality. Overall, this process will grant our team the structure to complete this project in an organized, timely, and complete manner.

When focusing on cost, it is important to understand our cost is very dynamic as that algorithm for matching and software to implement it can be started by us in the team as we are currently doing. With our current insights on our design, we can keep costs lower for now. We feel that by using free programs like Figma to continue to design, we can keep operating at our own cost. The other aspect to consider for current costs is the matching algorithm itself. We can most likely use free open-source machine learning software to match people, but we feel that this can be more difficult. This could have a cost of more time to figure out or even a cost to hire and invest money into programs that could help us with this. The same cost will still hold for the future, as we continue to grow and when we get to the part where we want to have our application available to be used nationally, we will have to invest in advertisement, more hires, and the infrastructure to make sure it works at a bigger scale.

Right now we are assuming a combination of Client-Server and a Model-View-Controller (MVC). We hope that in regard to our stakeholders, the Client-Server high-level architecture will allow multiple clients to use our product while our server supports that. This is important as we want the people using our app to be supported in doing so regardless of other users. Furthermore, we assume that our server will be able to support multiple requests at the same time, and linking back to cost we need to ensure we invest in a good and efficient server to support the high-level architecture we chose.

Focusing on our risk and success metrics, our product will be on the path to success if stakeholders want our application. Choosing a low- and high-level design for our product also incurs some risk if either design fails, if stakeholders dislike the design, or if it incurs too much cost. That being said this also affects the success metrics as continuing to narrow down the specification like our design is important in our ability to succeed. In the future, we want to define some more success metrics for our group as well.

When considering the timeline, our project has been developed throughout our school year and as the semester wraps up we want to discuss the project in the grand scope of things. We are

happy so far with our planning and completing the low- and high-level design. Our plan is to then be at a good breakpoint at the end of the semester so that we have the ability to break and potentially come back to it in an effective manner.

Testing Approach:

For our black box testing plan, we will focus on creating test scenarios that test many functionalities of our tool. We want to test the user preference input, the effectiveness of our matching algorithm, team assembly capabilities, and the feedback system. Each scenario will have multiple test cases, including normal cases for standard inputs, boundary cases that limit acceptable inputs, and error cases for inputs that we expect will generate errors. The process will involve executing these test cases, and documenting the inputs, expected outcomes, and actual outcomes so that we can determine if it was a pass or a fail. Then we will conduct a thorough review of the test result to identify any discrepancies and reasons it might've failed. Afterward, we will prepare a detailed report outlining the testing process, results, and any discovered bugs or issues, along with recommendations for improvements.

For a user preference input test example, we might input a set of user preferences like an intermediate coding skill level and preference for Python, and expect that the tool will correctly store these values. For a matching algorithm test example, we might input two users with preferences that complement each other and see whether the tool matches them as expected. To ensure that the pair programming matching tool works as intended from the user's perspective and adheres to the desired quality standards we will follow this structured black box testing approach. We will also make periodic reviews and updates to the test plan as necessary to adapt any tool changes or enhancements.

5 DEPLOYMENT & MAINTENANCE

For our Pair Programming Matcher Tool, it is very important we discuss how we would both deploy and maintain our application. First, we will focus on our deployment. Deployment is the delivery of software to users, and after working hard on the Pair Programming Matcher Tool, the goal is to deliver our program to users so they can benefit from what we have built. We want to follow the key deployment concepts in our deployment: staging, building, CI/CD, and test automation. Staging is building confidence in our project in a production-like environment, and waterfall staging is more intensive with more and more expensive testing at each stage. We have started in the staging process, by making a working prototype and demoing it, and then explaining it. However, our prototype is very simple and not super close to our ideal finalized product. This does mean we are in the process of waterfall staging as in the future we can invest more money into our tests and spend more time to develop a better prototype to then test. We would hopefully have a better test and try to have people in a paid environment test the product. For build, we want a version of our app in pre-release, that our development team can use. We are a ways away from this but we will definitely carry through on this step. Next, we will

focus on continuous integration and continuous development. Essentially, with every change we want to make sure that we test it effectively, and then furthermore we want to ensure we are constantly improving. Especially regarding the simplicity of our first prototype, we want to continuously improve and integrate new software changes and features. If we stay committed to making changes, even in deployment, we can help our users' best interests. Focusing on test automation for deployment is also important. We will mostly use fuzz testing and mutation testing because these fault-based tests will make sure we deal with any errors. Especially with an application that we want to be used by many users, there can be issues we might not think of to test, and these fault-injecting tests will ensure our application can stand up to other issues and is safe. Overall, with respect to deployment, we need to make many changes before deploying our product, but once we do we will work to have a rolling deployment, are our target environments will be incrementally updated. This will lead to less risk and support for multiple versions, and while it is slower we believe we have the time.

After deployment, it is essential that we keep good maintenance so that we keep up with our customer needs. We want to make sure that the code that we have written is maintainable, readable, reusable, and bug free, so that as we maintain our application we can actually keep it working and running while dealing with errors. Consequently, one of the best ways we can have good maintenance is to simply write good code that is well thought out and makes sense. Obviously this is easier said than done, but being cognizant that what we are writing will be reused can help us later. Following that, the maintenance we want to focus most on is preventive software maintenance. We believe that predicting changes to keep software working for a long time is essential for our application. We want to make sure that if we find any issues, even if it is a latent fault, we work to make an upgrade to prevent bugs that can arise from that later. We also note that corrective software maintenance is very common and we will also use that technique as well. As we are working we will also refactor portions of our code, to increase readability and reusability, along with reducing the code size overall. Also, within our development team, we will have code review, so we can manually inspect source code changes, to ensure requirements are ment, consistent design, and consistent implementation. Overall, we believe that following these maintenance steps will allow our application to continue working effectively and keep working for the customers needs.

6 FUTURE WORK

In the future, additional new features and enhancements can be added to improve the Pair Programming Matcher Tool. To improve accuracy in matching, we want our matching algorithm to leverage a broader range of user data like coding habits, project histories, and even using AI to observe soft skills. Another significant area of focus is our reviewing system. In the future, we can implement a more robust system that alerts users about potential negative interactions based on past reviews, in light of fostering a more positive and productive community. Since programmers have different needs, it is a good idea to integrate built-in IDE support. This will allow our users to collaborate within the coding environment they prefer, and allow pairs and teams to code in real-time, share resources, and even conduct code reviews, all within the same platform.

7 REFERENCES

[1] K. -W. Han, E. Lee and Y. Lee, "The Impact of a Peer-Learning Agent Based on Pair Programming in a Programming Course," in IEEE Transactions on Education, vol. 53, no. 2, pp. 318-327, May 2010, doi: 10.1109/TE.2009.2019121.D
[2] Formation, https://formation.dev/programweek (accessed Sep. 22, 2023).
[3] “CS Research Mentorship Program,” Google Research, https://research.google/outreach/csrmp/ (accessed Sep. 22, 2023).