Evaluation of Pair Programming

Praanto Samadder and Rushil Garg

May 16, 2024

Abstract

Hello World

1 Introduction

This report aims to assess the efficacy of pair programming within a group comprising individuals with varying levels of programming expertise. It delves into the outcomes and obstacles encountered during the development of a fact-checking tool utilizing natural language processing. Our evaluation of pair programming effectiveness is conducted through practice observation and comparison with findings from other researchers.

2 Background

Traditional development methods follow a sequential, linear approach with distinct phases like requirements gathering, design, implementation, testing, and maintenance. Examples include Waterfall and the V-Model, where each phase must be completed before moving on, leading to longer development cycles and less flexibility compared to Agile.

Agile methods prioritize flexibility, collaboration, and customer feedback in iterative software development. They emphasize adaptive planning, continuous improvement, and delivering frequent, small increments of usable software through methodologies like Scrum, Kanban, and Extreme Programming (XP). Agile fosters responsiveness to change, customer involvement, and efficient delivery of value to stakeholders.

The key differences lie in planning, flexibility, and iteration. Agile methods emphasize adaptability, collaboration, and delivering frequent increments of usable software, while traditional approaches favor a sequential, linear process with less room for change once a phase is complete. Agile encourages continuous improvement and customer involvement, while traditional methods focus on following a predefined plan. Agile suits dynamic projects, while traditional methods may be better for stable, well-defined requirements.

2.1 Pair programming

Pair programming is a collaborative software development technique where two programmers work together at one workstation. One programmer, known as the driver, writes the code, while the other, called the navigator, reviews each line of code as it is written. This dynamic duo continuously switches roles, enhancing code quality through real-time feedback, improved problem-solving, and knowledge sharing. Pair programming promotes teamwork, reduces errors, and fosters learning and creativity within the development process.

2.2 The project itself

This report encapsulates insights and learnings gleaned from an agile project focused on creating a fact-checking solution utilizing natural language processing (NLP). The objective of the project was to develop either a website enabling users to submit claims or a browser extension allowing users to scan articles for potential claims, with subsequent fact-checking functionality.

When scanning for articles, the backend would first use the LLM API to do claim extraction, then send a request to the retrieval API to retrieve relevant sources. Finally, the backend sends off those sources to the LLM API in order to run fact checking on them and returns the result to the front-end. The user sees these results as text highlights on the article that they ran the scan on.