Proposal and AI Adoption Plan for Software Development

Background and Hypotheses

- **AI is a tool, not an autonomous intelligence.**
  It works as an advanced search and generative engine, providing the most relevant and popular solutions based on the analysis of large amounts of data. For maximum effectiveness, AI requires clear task setting, intermediate control, and validation of results (manually or via tests).
- **Knowledge and skills are the foundation of productivity with AI.**
  To work effectively with AI, it is necessary to have:
  - Deep knowledge of the frameworks and best development practices in use.
  - Prompt engineering skills (the ability to formulate queries for AI).
  - An understanding of AI's limitations and principles (AI is a language model, not an expert).
- **The golden mean.**
  It is important to strike a balance between independent work and using AI: AI is an assistant that needs to be given goals, tasks, and whose every step should be checked.
- **The more complex the task, the higher the risk of an incorrect result.**
  For large and complex tasks, AI should be used as an auxiliary tool, not as the main executor.
- **Fundamental preparation is the key to success.**
  Before adopting AI, it is important to:
  - Bring the codebase up to modern standards.
  - Identify and close knowledge gaps in frameworks and best practices within the team.
  - Teach all developers the basics of prompt engineering.
- **Knowledge sharing is the foundation of long-term success.**
  It is necessary to create a culture of sharing prompts, cases, challenging situations, and examples of unsuccessful AI usage. This will not bring instant results, but will lay a solid foundation for future growth.

What Has Already Been Implemented

- **GitHub Copilot licenses** have been issued to all developers, and the tool is integrated into IDEs.
- **General AI training sessions** have already been conducted for the entire team.
- **Individual work is planned** with each developer, including practical sessions on prompt engineering and effective AI usage.

AI Adoption Plan

Stage 0: Fundamental Preparation

- **Skills and codebase audit:**
  Assess the team's knowledge of frameworks, best practices, identify gaps and technical debt.

- **Prompt engineering training:**
  Conduct training and practical sessions for all developers.
- **Bringing code to standards and refactoring:**
  Carry out refactoring, not only eliminating technical debt but also improving architecture, readability, and maintainability. With AI, these tasks are now much easier and faster to accomplish.

Stage 1: Integrating AI into Processes

- **Pilot wave:**
  First, train and involve key and lead developers so they become influencers and set the trend for the rest of the team.
- **Integration of AI tools:**
  Implement the best solutions (e.g., GitHub Copilot) and configure them for the technologies in use.
- **Gradual expansion:**
  After a successful pilot, scale up to the entire team.

Stage 2: Developing a Knowledge Sharing Culture

- **Creating an internal knowledge base:**
  A platform for sharing prompts, cases, challenging situations, and examples of failures.
- **Regular discussions:**
  Meetups, workshops, and reviews of new AI capabilities.
- **Mentorship:**
  Appoint those responsible for supporting and training colleagues.

Stage 3: Continuous Improvement

- **Monitoring and feedback:**
  Regularly collect feedback, assess the impact of AI on productivity and quality.
- **Skills update:**
  Conduct periodic training and keep the knowledge base up to date.
- **Experimentation:**
  Constantly test new approaches and share results.

Key Focus Points

- **AI is a tool whose effectiveness depends on the developer's level and team culture.**
- **Fundamental preparation and knowledge sharing are the foundation of long-term success.**
- **AI adoption is not a one-off event, but a systematic process requiring leader involvement, training, and continuous improvement.**
- **Thanks to the integration of AI and the best tools on the market, such as GitHub Copilot, tasks that were previously complex and time-consuming (refactoring, technical debt elimination, training, and knowledge assessment) are now much easier and faster to accomplish. This allows you to quickly bring the codebase and processes in order, laying the foundation for further updates and improvements to the entire development process.**
- **The main focus is on improving what already exists and using proven solutions, rather than creating your own AI tools with a high risk of being unclaimed.**

- **AI is in demand at all stages of work: from identifying and closing knowledge gaps, to freeing up capacity for key developers and refactoring the codebase.**
- **Instead of building another 'cool tool', it is important to focus on preparing for AI adoption at scale across the entire team.**
- **Yes, the effect may not be immediately noticeable, but this is the foundation for a 'multi-story building' that will emerge in the next stages of team and process development.**
