[**Homework 3-4**](https://github.com/hendraanggrian/IIT-ITM511/blob/assets/assignments/hw3-4.pdf)**: Agile and requirements engineering**

**Problem 1**

*At the end of their study program, students in a software engineering course are typically expected to complete a major project. Explain how the agile methodology may be very useful for the students to use in this case.*

Agile methodology is suitable for the final project of software engineering study because of its shorter release cycle that allows the student to produce a fast – albeit flawed – prototype. Being fast is the key here, as students can collect early feedback from the instructor. By subscribing to Agile, the software development planning is not bound to a fixed schedule and may change according to given feedback.

**Problem 2**

*Explain how the principles underlying agile methods lead to the accelerated development and deployment of software.*

Through the principle of incremental development, agile methodology breaks down the project into smaller and more frequent release cycles (Sommerville, 2016). The compact size of each part enables faster testing and targeted code refactoring. In this environment, new features may arrive quicker by delivering unpolished results swiftly, with the understanding that ongoing iterations will refine them based on user feedback and evolving needs.

**Problem 3**

*Extreme programming expresses user requirements as stories, with each story written on a card. Discuss the advantages and disadvantages of this approach to requirements description.*

Extreme programming methodology is a subset of agile development where the customer is more involved in the requirements-gathering process. Instead of standard specifications, extreme programming encourages the use of user stories for an entertaining engagement between stakeholders. The simplicity of user stories makes it easier for the customers to illustrate their needs without going into much detail (Cohn, 2022). However, this approach may not be appropriate for large projects because of its ambiguity in determining time and cost.

**Problem 4**

*In test-first development, tests are written before the code. Explain how the test suite may compromise the quality of the software system being developed.*

Since test-first development revolves around testing, how we conduct the tests is indispensable to the subsequent processes. For example, flawed tests could lead to misunderstanding of what the project represents. Moreover, test updates would also be necessary once the underlying code becomes available.

**Problem 5**

*Suggest four reasons why the productivity rate of programmers working as a pair might be more than half that of two programmers working individually.*

1. **Quality over quantity:** Produced solutions are of the highest quality as every line of code is peer-reviewed by both programmers. The extra attention to detail would make the code less likely to be defective.
2. **Shared knowledge:** By pairing programmers to work together, they form a collective intelligence and combined proficiency. The constant exchange of information makes pair programming ideal for mentoring junior developers.
3. **Emotional support:** Frustration is common when encountering complex problems or unforeseen issues, potentially disrupting overall progress. When it happens, it is crucial to have proper support in navigating emotional challenges. Although I suspect the worker's affection and personality play a huge role in the effectiveness of the support.
4. **Sense of ownership:** Collaboratively authored solutions by multiple programmers promote the idea that the entire engineering team is accountable for the work. With the current procedure, a single programmer is unlikely to be blamed for a sloppy performance.

**Problem 6**

*Explain why agile methods may not work well in organizations that have teams with a wide range of skills and abilities and well-established processes.*

Agile methods prioritize speed delivery, which sometimes comes at the expense of other aspects of the project. For example, an engineering team may deliberately suppress tests or documentation because resolving them would mean extending the deadline. The lack of structure and clarity puts the project at a disadvantage and unfit for a team of diverse expertise and pre-determined processes.

**Problem 7**

*One of the problems of having a user closely involved with a software development team is that they ”go native.” That is, they adopt the outlook of the development team and lose sight of the needs of their user colleagues. Suggest three ways how you might avoid this problem, and discuss the advantages and disadvantages of each approach.*

1. **Define non-negotiable requirements:** Expect customers to set boundaries when discussing needs and goals.

* **Pro:** Build a trusting relationship by empowering users with the authority to make their own choices.
* **Con:** Unprepared customers may find it uncomfortable, stalling the discussion.

1. **Find a second opinion:** Occasionally invite other users or engineers to the negotiation table.

* **Pro:** Prevent personal biases by gathering more perspectives.
* **Con:** Requires additional employee allocation and training, chipping resources away from other departments.

1. **Use requirements versioning:** Track customer demands to compare preferences over time.

* **Pro:** Identify recent trends as well as previous requests.
* **Con:** Collecting more data complicates the documentation.

# Bibliography

Cohn, M. (2022, October 5). *Advantages of User Stories over Requirements and Use Cases*. Retrieved from Mountain Goat Software: https://www.mountaingoatsoftware.com/articles/advantages-of-user-stories-for-requirements/

Sommerville, I. (2016). Software Engineering. In *Agile development techniques* (10 ed., pp. 77-78). Pearson Education.