Agile practices definition

The following agile practices have been extracted from the categories of agile principles defined by Palopak and Huang in 2024 [1]. This set, like others, was created to interpret how agility benefits teams in software project development and to establish guidelines for decision-making in the industry.

Palopak and Huang define six categories of agile principles. In each of them, we define a set of practices by reviewing the Agile Manifesto [2], the questionnaire Palopak and Huang used in their study [1], and the most popular agile methodologies for software development, e.g., Scrum [5] and XP [3]. Some of the original principles associated with each set have been modified. Consider the following definitions for each of the agile practices studied in the retrospective analysis.

1. Team proactivity (principles 5, 11 and 12).
    - 1.1. **Work motivated regarding the purpose of the project:** They show enthusiasm for achieving the objectives of the project, going beyond just fulfilling the obligations of the subject [2].
    - 1.2. **Improve team processes:** They carry out activities to solve problems or propose improvements regarding the way of developing software, relating to each other or overcoming technical or team difficulties [1, 2].
    - 1.3. **Check whether completed tasks meet acceptance criteria:** Define acceptance criteria or a definition of “done” for each task, which is reviewed by the team [2, 5, 6].
2. Team cohesion (principle 6).
    - 2.1. **Use a pair programming approach:** They carry out project activities, both development and others, by meeting in groups of two people and working simultaneously, supporting each other to solve tasks [3].
    - 2.2. **Report the status of tasks:** They report the status of the tasks that each member performs, whether related to software development or project management [5, 6].
    - 2.3. **Communicate frequently and effectively:** They constantly communicate with each other regarding important issues of the project or the situation of each member, seeking to be effective in their conversations regarding the way they transmit information [2, 4].
    - 2.4. **Meet face-to-face with the team:** They organize occasions where they can meet in person, either to work or discuss aspects of the project, focusing on having direct, face-to-face communication with the members [2, 5].
    - 2.5. **Daily meetings:** Organize and attend daily team meetings to learn about the status of the project and make decisions about its direction [5].
3. Customer collaboration (principles 2 and 4).
    - 3.1. **Seek feedback from teachers:** They consult the opinion of teachers or assistants to adjust the direction of the project, check the current quality of the product or resolve doubts regarding technical or project management aspects [2, 8].
    - 3.2. **Seek customer feedback:** They consult customer opinion to check the current quality of the product and their satisfaction with it at regular intervals [2, 5].
4. Process simplicity (principles 8 and 10).

4.1. **Choose a simple design first to develop software:** They do not use a complex and complete design of the entire system and its components, focusing first on completing the desired functionalities and checking their correctness [3].

4.2. **Maintain a constant and sustainable pace:** They work on the project in a sustained manner over time, completing the assigned tasks in the same way, overcoming blockages, avoiding time lapses in which the project is not being worked on and ensuring that the team is not overloaded [2].

4.3. **Avoid tasks that are not valuable to the client:** Focus the development of the project on carrying out those tasks that are important to the client, avoiding those that do not contain direct value for the client or for the completion of the Sprint [1, 2].

4.4. **Assign simple tasks to multifunctional members equitably:** They define simple tasks to be performed and assign them equitably to the team members, who have the ability to perform in any area within the development of the project [2, 3, 4, 5].

4.5. **Avoid excessive technical debt through refactoring:** They perform a restructuring (or refactoring) of the code or the work done in order to solve the possible problems that the technical debt produced by moving quickly with simple designs can bring [1, 3].

5. Regular delivery (principles 1, 3 and 7):

5.1. **Prioritize tasks based on their value to the customer:** They know the customer's needs and prioritize performing the tasks that provide the greatest value to him or her first [1, 2].

5.2. **Establish a suitable scope for the Sprint:** They estimate the scope of the Sprint considering the team's capacity and the adjusted effort they estimated for the functionalities to be developed, learning from the effort previously made when developing other functionalities [2, 3, 5].

5.3. **Follow a continuous integration approach:** Integrate the work done by each of the members into a common production branch, making the progress in the project tasks immediately reflected in the progress of the software product to be developed [2, 3].

6. Technical excellence (principle 9 and 11):

6.1. **Obtain the technical and theoretical knowledge necessary for the tasks:** They report having the technical and theoretical knowledge to tackle the assigned tasks or, if they do not have it, they seek to obtain it through training or research on the subject [7].

6.2. **Write simple and understandable code:** They write the system code, ensuring that it is simple in terms of cyclomatic complexity and understandable for the rest of the team in terms of its documentation and clarity in the description of functions and variables [1].

6.3. **Follow integration and versioning standards:** Define and follow standards or guidelines for the integration of each member's work into the team's version system, preventing it from being a non-systematic process [1, 3].

6.4. **Follow implementation standards:** Define and follow standards or guidelines for writing system code or performing tasks, so that each member works in a similar way and can more easily understand the functionalities that others develop [1, 3].

6.5. **Have a collective ownership of the code:** They know all of the code or the details of the functional aspects of the implemented software system, even those they did not work on [3].

6.6. **Apply integration tests:** They test that the entire software system works as expected after the work of a member is integrated into the productive version of the system, verifying at the same time that the previously implemented functionality has not been affected [1, 3].

6.7. **Apply unit tests:** Apply unit tests to each of the pieces of code implemented when completing a task, seeking to review the functional correctness of the smallest components of the system [1].

References:

[1] Palopak, Y., & Huang, S. J. (2024). Perceived impact of agile principles: Insights from a survey-based study on agile software development project success. Information and Software Technology, 176, 107552.

[2] Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001, February). The agile manifesto.

[3] Shrivastava, A., Jaggi, I., Katoch, N., Gupta, D., & Gupta, S. (2021, July). A systematic review on extreme programming. In Journal of Physics: Conference Series (Vol. 1969, No. 1, p. 012046). IOP Publishing.

[4] Lindsjørn, Y., Sjøberg, D. I., Dingsøyr, T., Bergersen, G. R., & Dybå, T. (2016). Teamwork quality and project success in software development: A survey of agile development teams. Journal of Systems and Software, 122, 274-286.

[5] Schwaber, K., & Sutherland, J. (2011). The scrum guide. Scrum Alliance, 21(1), 1-38.

[6] Ahmad, M. O., Markkula, J., & Oivo, M. (2013, September). Kanban in software development: A systematic literature review. In 2013 39th Euromicro conference on software engineering and advanced applications (pp. 9-16). IEEE.

[7] Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. Computer, 34(11), 131-133.

[8] Masood, Z., Hoda, R., & Blincoe, K. (2018). Adapting agile practices in university contexts. Journal of Systems and Software, 144, 501-510.