Extreme Programming (XP) Article

Arturas Bulavko

April 2017

Introduction to System Development methods

System Development method is a widely recognised framework to structure and manage the project in order to develop an information system. There are several methods and each of these methods come with advantages and disadvantages which help the people in charge decide whether specific methodology is appropriate for their project.

Introduction to XP as an Agile approach

Agile methodology has evolved from Waterfall primarily by introducing iterations as demonstrated in *Figure 1*. Such approach helps limit the cost of change by having strong communication within the team in order to deliver high value in short periods of time. Agile strongly focuses on rapid delivery of working product components in order to evaluate and improve functionality during the next timebox.

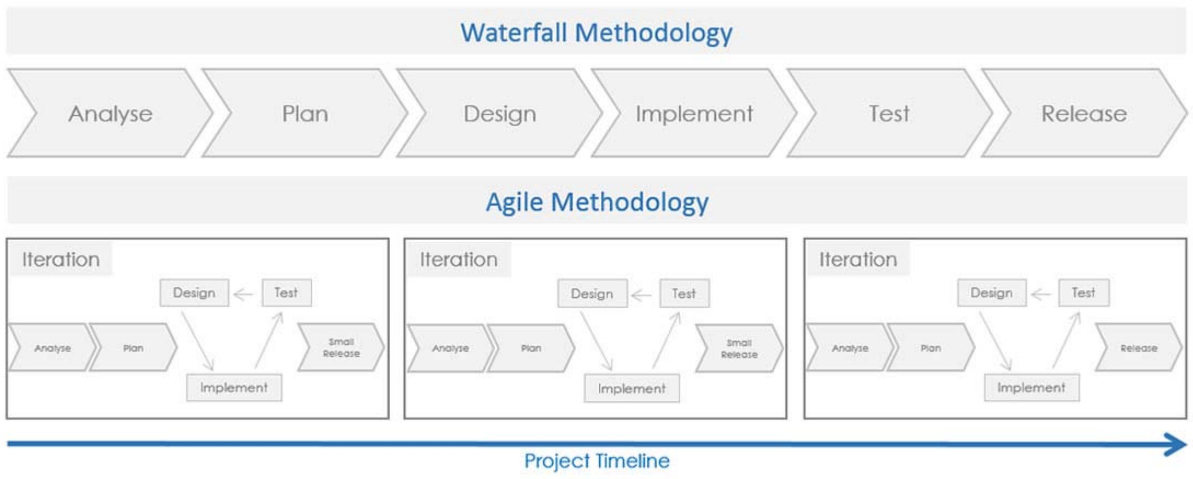

Figure 1 – Waterfall vs Agile Methodology

Explanation of life cycle stages

XP is an Agile methodology which strongly focuses on interactions between the team members to achieve maximum efficiency. Generally, all software development methodologies follow a linear approach where each phase is completed in order and upon completion initiates the next one. However, XP mixes all of the phases throughout the development lifecycle and each person involved undertakes phases in various order. Such approach is very valued due to the flexibility it brings. XP incorporates the following phases in the development lifecycle.

- ✓ Planning – at this stage the development team and customers meet to create user stories.
- ✓ Designing – creating brief solutions to the outlined problems.
- ✓ Coding – converting designs into actual code considering collective code ownership.
- ✓ Testing – testing and eliminating bugs, as well as ensuring Customer Acceptance tests are done.
- ✓ Listening – obtaining feedback about specific components and discussing potential improvements with the team to maximise the efficiency.

Project Roles

According to Beck and Andres (2005, p.82) the roles of a team are not “fixed and rigid” enabling each team member to contribute towards the project with their expertise ensuring its success. The following fundamental roles are required in order to make the project work.

- ✓ Customer – sets the project goals and makes the business decisions. The customer is also in charge of analysing risks and providing precise user stories to aid developers.
- ✓ Developer/Programmer – converts customer stories into code by following the team’s guidelines/standards. It is also vital for the developer to have good communication with the customer to avoid making certain decisions on their own.
- ✓ Tracker – ensures the project is on schedule using velocity metrics.
- ✓ Coach – guides the team and gives expert advice to introduce better practices.
- ✓ Tester – creates and runs tests. Feedbacks to the team with the test outcomes.
- ✓ Manager – organises meetings and ensures they are productive.

Five Values

XP is built upon five fundamental values which dictate success but seem ambiguous due to differences in point of view. Such values keep the workplace in harmony making everyone very productive. Below are the five XP values.

- ✓ Simplicity – emphasises on developing only today’s problem today and keeping it as simple as possible. Do not spend precious time on features which are not needed currently.
- ✓ Feedback – it is vital to test all code as soon as you implement it. Similarly, it is always best to ask customer for the feedback to ensure problems are identified as soon as possible.
- ✓ Communication – XP makes all teams members communicate as much as possible to ensure everyone knows what they are doing and thus remain on schedule.
- ✓ Courage – guarantees that mistakes will be admitted and resolved. Furthermore, XP expects all team members to tell the truth at all times because this can potentially eliminate problems further down the development lifecycle.
- ✓ Respect – all team members must be respected and treated equally independently of the role they undertake. No matter how big or small the impact is, everyone contributes.

Five Fundamental Principles

Another main feature of XP is principles which act as a measurement of values. They are considered to be binary and can be easily checked; they are either done or not. Below are the five principles.

- ✓ Rapid Feedback – implement the obtained feedback into the system promptly, ensuring it is correctly understood.
- ✓ Assume Simplicity – keep all work as simple as possible and implement complex tasks only when they are needed. YAGNI and DRY principles can be followed to ensure this.
- ✓ Incremental Change – break big problems into smaller tasks which can be implemented quickly rather than investing time into developing a complex solution which might not work.
- ✓ Embracing Change – accept the modification even if it looks incorrect and continue working without going back to the problem.
- ✓ Quality Work – high quality standards are always met by the team members which makes them proud of their work, enabling them to keep motivated working further.

Four basic activities

XP is established upon four activities which drive the project, outlined below.

- ✓ Coding – the outcome of the project is a software; therefore, this activity is considered to be top priority whereas other activities are there to support it.
- ✓ Testing – all features must be thoroughly tested and documented to prevent costly problems towards the end of the project.
- ✓ Listening – it is vital for the programmers to understand the business side in order to develop a solution to match a specific problem provided by the customer.
- ✓ Designing – helps organise the logic of the system once it starts to become too complex to keep track of all dependencies.

Other XP practices

Extreme Programming is also built upon the following practices which are vital to understand in order to be on the peak of effectiveness during the development lifecycle.

- ✓ Planning Game – main planning process occurring once a week which consists out of Release Planning and Iteration Planning. It enables development team to sketch a brief plan which can be amended during the project.
- ✓ Small Releases – focusing on developing small releases which build up into a complex system. Jeffries et al. (2000, p.50) point out that majority of failed projects could have been successful if they were to adapt small releases. This is proven by XP's low failure rates.
- ✓ Metaphor – making the team follow naming conventions and set team standards to ease finding specific variables, class and document names.
- ✓ Simple Design – keeping the design as simple as possible and making it achieve the story card, pass all tests and satisfy the business need.
- ✓ Testing – ensuring that each small release is thoroughly tested and all bugs are fixed.
- ✓ Refactoring – each team member working concurrently to keep the code clean and non-repetitive. In some cases, rewriting the code to a better standard.
- ✓ Pair Programming – all programming must be done in pairs. According to Bryant et al. (2008) pair programming is beneficial because one might understand the problem better and create a better mental solution to it.
- ✓ Collective Ownership – everyone in the team working on the code, thus no individual is responsible for it.
- ✓ Continuous Integration – merging the code written by different programmers into main repository. Chromatic (2003, p.34) emphasises that this task must be automated as much as possible to be able to spend more time on the actual development of the software.
- ✓ 40-hour week – to keep all developers enthusiastic, a strict schedule is followed which permits them to work 40 hours a week keeping them motivated at all times.
- ✓ On-site customer – keeping customer on site allows rapid feedback to be obtained as well as elicit further user stories.
- ✓ Coding standards – the code should ideally follow same style for easier integration. All metaphors must be consistent.

Importance of capturing the 'story'

Story cards are considered as an artefact and are written by the customer. They are primarily used to estimate how long it will take to implement a desired feature. A story card must represent one feature and must not be long but should provide enough information to ensure the developers understand what needs to be done. Furthermore, story cards are directly used with Acceptance Tests to confirm that the requirement was correctly implemented. Story cards are important because they help plan iteration cycles and prioritise functionality.

Phases within the planning game and iteration planning

Planning game takes place in a form of an Iteration Planning and Release Planning, where Iterations focus on the developer and Releases focus on the customer. It can be seen as a meeting with all roles to understand what features are required and estimate how long each will take. Postmus et al. (2011) point out that in XP, the primary idea is to "plan features to implement rather than the development tasks" which are required to be carried out in order to implement these features. As such, the following phases are done to ensure business value is met in the shortest amount of time.

- ✓ Exploration – Customer writes a story card with the business' problem, the development team estimates how long it will take to implement a solution to this story. If the story card is too complex, the customer will be asked to split it into few smaller story cards.
- ✓ Commitment – At this phase the business sorts user stories by business values and development team sorts them by risk. The development team also determines velocity of the project and picks users stories which will be implemented for first release.
- ✓ Steering – An opportunity for programmers and business people to adjust the plan.

Conclusion

It is clear that XP is a very effective system development methodology. However, from personal experience it is challenging to form a team which can adopt all these practices to work effectively together. Considering that XP is focused around people, sometimes such methodology might not be appropriate because the business customer is not prepared to sit in the office with the developers to provide constant feedback. Otherwise, when all team members are eager to work, XP makes the project a pleasant journey, ensuring a correct solution is delivered on time.

References:

Beck, K., Andres, C. (2005) *Extreme programming explained : embrace change*. 2nd edn. Boston, Mass.: Addison-Wesley.

Bryant, S., Romero, P., Du Boulay, B. (2008) 'Pair programming and the mysterious role of the navigator', *International Journal of Human – Computer Studies*, 66(7), pp.519-529. Doi: 10.1016/j.ijhcs.2007.03.005.

Chromatic (2003) *Extreme Programming pocket guide*. Beijing ; Cambridge ; O'Reilly.

Jeffries, R., Anderson, A., Hendrickson, C. (2000) *Extreme programming installed*. Harlow : Addison-Wesley.

Postmus, D., van Valkenhoef, G., Tervonen, T., de Brock, B. (2011) 'Quantitative release planning in extreme programming', *Information and Software Technology*, 53(11), pp. 1227-1235. Doi: 10.1016/j.infsof.2011.05.007.