altexsoft
software r&d engineering

WHITEPAPER

Agile Project Management: **Best Practices and Methodologies**

1. The Art of Project Management

Regardless of industry, project management has proven to be a crucial element of a company's efficiency and its eventual success. In fact, projects are usually 2.5 times more successful when proven project management practices in place[1].

As defined by Gartner, project management is "the application of knowledge, skills, tools and techniques to project activities to meet the project requirements"[2]. Being an integral part of software engineering processes along with the business analysis and requirement specification, design, programming and testing, the project management has been a topic of considerable debate for years.

Regardless of the scope, any project should follow a sequence of actions to be controlled and managed. According to the Project Management Institute (PMI), a typical **project management process** includes the following phases:

1. Initiation
2. Planning
3. Execution
4. Performance/Monitoring
5. Project close

Used as a roadmap to accomplish specific tasks, these phases define the project management lifecycle.

Yet, this structure is too general. A project usually has a number of internal stages within each phase. They can vary greatly depending on the scope of work, the team, the industry and the project itself.

In attempts to find a universal approach to managing any project, humanity has developed a significant number of PM techniques and methodologies.

2. Traditional Project Management Methodologies

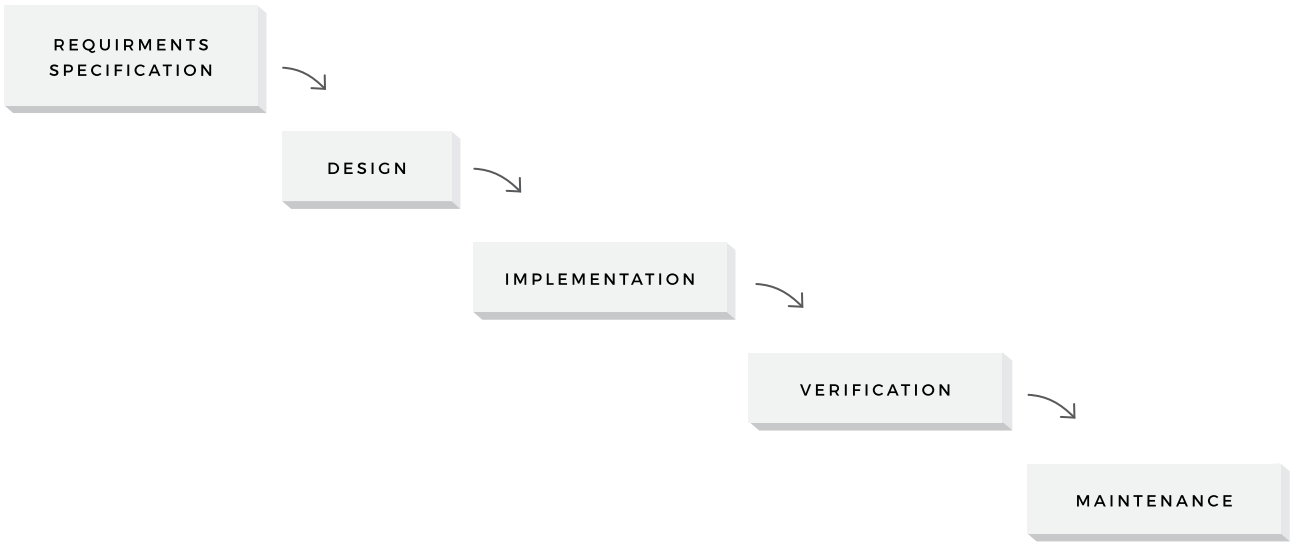
Based on the above-described classic framework, traditional methodologies take a step-by-step approach to the project execution. Thus, the project goes through the initiation, planning, execution, monitoring straight to its closure in consecutive stages.

Often called **linear**, this approach includes a number of internal phases which are sequential and executed in a chronological order. Applied most commonly within the construction or manufacturing industry, where little or no changes are required at every stage, traditional project management has found its application in the software engineering as well.

Known as the **waterfall model**, it has been a dominant software development methodology since the early 1970s, when formally described by Winston W. Royce:

> *There are two essential steps common to all computer program developments, regardless of size or complexity. There is first an analysis step, followed second by a coding step ... This sort of very simple implementation concept is in fact all that is required if the effort is sufficiently small and if the final product is to be operated by those who built it - as is typically done with computer programs for internal use.*

Waterfall Model



Waterfall model has a strong emphasis on planning and specifications development, which takes up to **40% of the project time and budget**. Another basic principle of this approach is the strict order of the project phases. A new project stage does not begin until the previous one is finished.

The method works well for clearly defined projects with a single deliverable and fixed deadline. The Waterfall approach requires thorough planning, extensive project documentation and tight control over the development process. In theory, this should lead to on-time, on-budget delivery, low project risks, and predictable final results.

However, when applied to the actual software engineering process, Waterfall method tends to be slow, costly and inflexible due to numerous restrictions. In many cases, its inability to adjust the product to the evolving market requirements, often results in a huge waste of resources and the eventual project failure.

3. Defining Agile Principles

As opposed to traditional methodologies, the A**gile approach** has been introduced as an attempt to make software engineering more flexible and efficient. With 94% of the organizations practicing agile in 2015[3], it has become the industry standard for project management.

The history of agile can be traced back to 1957: at that time Bernie Dimsdale, John von Neumann, Herb Jacobs, and Gerald Weinberg were using incremental development techniques (which are now known as Agile), building software for IBM and Motorola. Although, not knowing how to classify the approach they were practicing, they realized clearly that it was different from Waterfall in many ways [4].

However, the modern-day agile approach was officially introduced in 2001, when a group of 17 software development professionals met to discuss alternative project management methodologies. Having a clear vision of the flexible, lightweight and team-oriented software development approach, they mapped it out in the Manifesto for Agile Software Development.

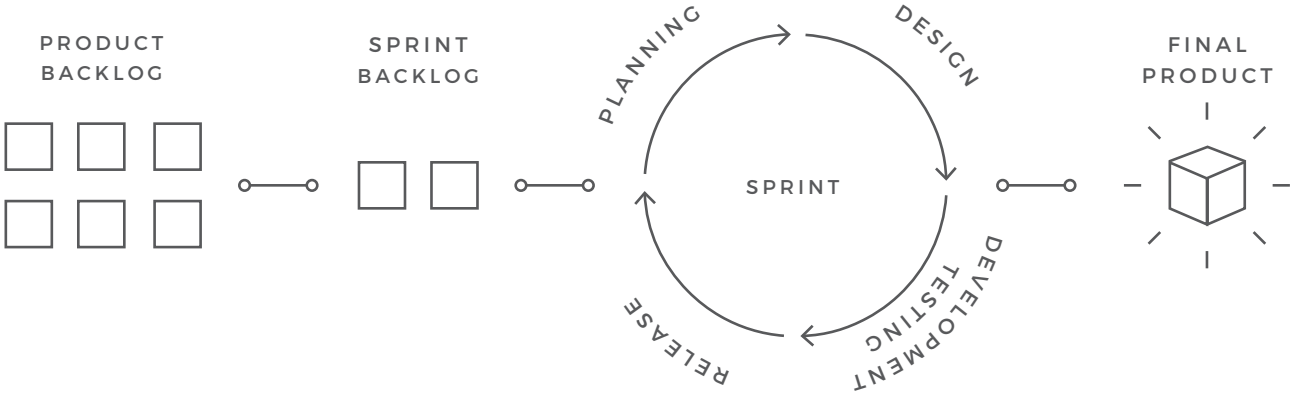
Aimed at "uncovering better ways of developing software", the Manifesto clearly specifies the fundamental principles of the new approach:

> *Through this work we have come to value:*
>
> ***Individuals and interactions*** *over processes and tools*
>
> ***Working software*** *over comprehensive documentation*
>
> ***Customer collaboration*** *over contract negotiation*
>
> ***Responding to change*** *over following a plan* [5].

Complemented with the Twelve Principles of Agile Software, the philosophy has come to be a universal and efficient new way of managing projects.

Agile methodologies take an iterative approach to software development. Unlike a straightforward linear waterfall model, agile projects consist of a number of smaller cycles - sprints. Each one of them is a project in miniature: it has a backlog and consists of design, implementation, testing and deployment stages within the pre-defined scope of work.

Agile Development Cycle



At the end of each sprint, a potentially shippable product increment is delivered. Thus, with every iteration new features are added to the product, which results in the gradual project growth. With the features being validated early in the development, the chances of delivering a potentially failed product are significantly lower.

Prioritizing flexibility and rapid turnaround, **the** Ag**ile approach offers the following benefits**, according to the recent research:

- Ability to manage the changing priorities (87%)
- Increased team productivity through daily task allocation (84%)
- Better project visibility due to the simple planning system (82%)[3]

4. Agile Methodologies

Agile is an umbrella term for a vast variety of methodologies and techniques, sharing the principles and values described above. Each of them has its own areas of use and distinctive features.

4.1 Scrum: Roles, Sprints and Artifacts

Scrum is a dominant agile methodology. It's used exclusively by 42% of organizations while another 54% of the companies combine it with other techniques[6]. First described in 1986 by Hirotaka Takeuchi and Ikujiro Nonaka in the New Product Development Game, this approach is based upon the systematic interactions between the three major roles: Scrum Master, Product Owner, and the Team.

- A **Scrum Master** is a central figure within a project. His principal responsibility is to eliminate all the obstacles that might prevent the team from working efficiently.
- A **Product Owner**, usually a customer or other stakeholder, is actively involved throughout the project, conveying the global vision of the product and providing timely feedback on the job done after every sprint.
- **Scrum Team** is a cross-functional and self-organizing group of people that is responsible for the product implementation. It should consist of up to 7 team members, in order to stay flexible and productive.

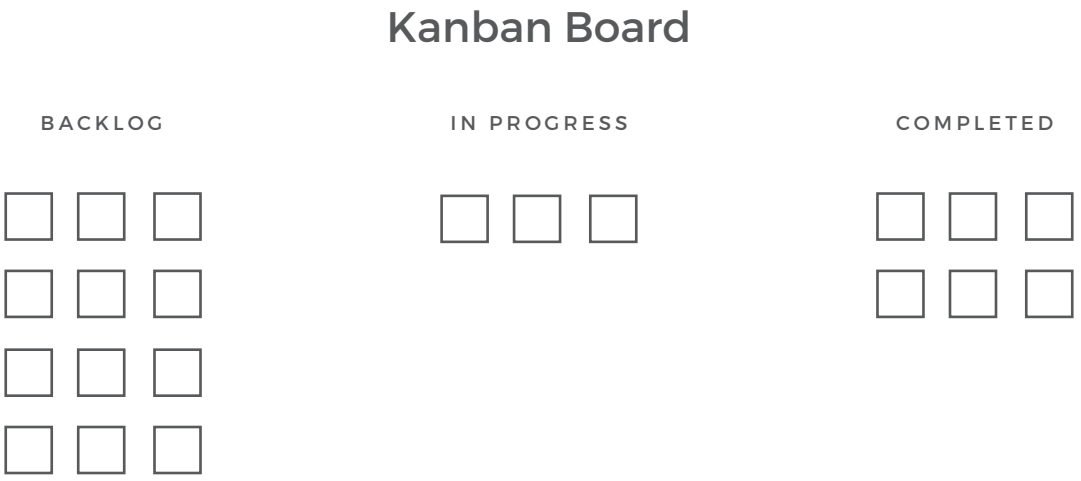
A basic unit of work in scrum - **sprint** - is a short development cycle that is needed to produce a shippable product increment. A sprint usually is between 1 and 4 weeks long: More lengthy iterations lack the predictability and flexibility that are scrum's fundamental benefits. Having no standard duration (as long as it is less than 4 weeks), all the sprints within a project should have a fixed length. This makes it easier to plan and track progress.

Scrum relies on **three main artifacts** which are used to manage the requirements and track progress - Product backlog, Sprint backlog, Sprint burndown chart. The process is formalized through a number of recurring meetings, like Daily Scrum (Standup), Sprint Planning, Review and Retrospective meetings.

By setting customer needs and on-time, on-budget delivery as the highest priority, Scrum has **overall project success rate of 62%** [6]. Thus, the list of companies using this approach is impressive. In fact, there is a public spreadsheet with such organizations, including Microsoft, IBM, Yahoo, and Google.

4.2 Kanban: Comprehensive Solution to Handling Work in Progress

Another most common project management framework is **Kanban**. It is reportedly used in some form by 43% of Agile organizations[6]. Originating from a visual system of cards used in Toyota manufacturing as a production control method, Kanban is simple, yet powerful, approach to developing software products.



Kanban prioritizes the **work in progress (WIP)**, limiting its scope to match it effectively to the team's capacity. As soon as a task is completed, the team can take the next item from the pipeline. Thus, the development process offers more flexibility in planning, faster turnaround, clear objectives, and transparency.

No standard procedures within the process, as well as the fixed iterations, are required in Kanban, as opposed to Scrum. The project development is based on the workflow visualization through a **Kanban board**, usually represented by sticky notes and whiteboards or online tools like Trello.

Companies like Spotify and Wooga (leading mobile games development company) have been using this approach successfully over the years. Yet, 43% of organizations combine Scrum with Kanban techniques, using the so-called **Scrumban** rather than the original methodologies [6].

4.3 Lean: Eliminating Waste in Software Engineering

Lean is the third most widely used agile approach, adopted by 21% of organizations[6]. Having the same origins as Kanban, the approach started as a technique applied to physical manufacturing. It stemmed from Toyota Production System as a management approach aimed at "making the vehicles ordered by customers in the quickest and most efficient way, in order to deliver the vehicles as quickly as possible"[7].

The application of Lean principles to software development was initially introduced by Mary and Tom Poppendieck in their book Lean Software Development: An Agile Toolkit. It includes the **7 basic principles**:

- Eliminate waste
- Amplify learning and create knowledge
- Decide as late as possible
- Deliver as fast as possible
- Empower the team
- Build integrity/quality in
- See the whole [8]

Basically, these fundamentals perfectly describe Lean philosophy: Its aim is to deliver more value through less effort, investment and time. In terms of a project, a term "waste" refers to anything that is not adding the value to the project and thus should be eliminated. In software engineering, this can be idle time, unnecessary features or defects.

Lean software development is an **iterative and incremental methodology**. Therefore, as in any other Agile approach, the working product increment is delivered at the early stages of the development process. The further progress depends largely on the product owner's feedback.

What differentiates Lean approach is that the team is not restricted to use any formal processes, such as recurring meetings or thorough task prioritization. However, its principles are often combined with other Agile techniques: 21% of teams **combine Scrum with Lean** [6].

Being effectively adopted by a vast number of manufacturing companies, like Nike, Ford and Intel, Lean principles are widely used in other industries. Startups and successful companies, e.g. Corbis, PatientKeeper, and Xerox, apply Lean software engineering practices to their processes.

5. Agile Software Engineering Best Practices: Extreme Programming

While the methodologies listed above focus mostly on the project management process and its formal aspects, there are a vast number of agile practices that are solely technical. Extreme Programming (XP) combines the most essential, providing agile teams with a number of tools to optimize the engineering process.

Extreme Programming is a set of certain practices, applied to software engineering in order to improve its quality and ability to adapt to the changing requirements. Kent Beck, one of the initial signatories of the Agile Manifesto, was the first to document these practices in his book Extreme Programming Explained: Embrace Change.

The most commonly used **XP practices** are:

- Test-Driven Development (TDD),
- Refactoring,
- Continuous Integration,
- Pair Programming.

Test-Driven Development is an advanced engineering technique that uses automated unit tests to propel software design process. As opposed to the regular development cycle, where the tests are written after the code (or not written at all), TDD has a test-first approach. This means that the unit tests are written prior to the code itself.

According to this approach, the test should fail first when there is no code to accomplish the function. After that the engineers write the code, focusing on the functionality to make the test pass. As soon as it's done, the source code should be improved to pass all the tests. These three steps are often referred to as the **Red-Green-Refactor cycle** [9].

TDD has proven to provide the following benefits:

1. The tests are used to capture any defects or mistakes in the code, providing constant feedback on the state of every software component. Thus, the quality of the final product is increasingly high.

2. The unit tests can be used as an always up-to-date project documentation, changing as the project evolves.

3. Being deeply involved in the product development, the team needs to be able to critically analyze it and foresee the planned outcome in order to test it properly. This keeps the team motivated and engaged, contributing to the product quality.

4. With a thorough initial testing, the debugging time is minimized.

Apart from being used within the TDD cycle, **code refactoring** is a common practice in agile software development. Basically, it's a process of a constant code improvement through simplification and clarification. The process is solely technical and does not call for any changes in software behavior.

Extending the source code with each iteration, agile teams use refactoring as a way to weed out code clutter and duplications. This helps prevent software rot, keeping the code easy to maintain and extend.

Continuous Integration (CI) is another practice agile teams rely on for managing shared code and software testing. Tools like CruiseControl or Jenkins are used to verify the quality of the software and automate its deployment. In addition, CI helps maintain the shared code, eliminating the integration issues. Thus, the product's mainline is robust and clean and can be rapidly deployed.

Pair Programming, or "pairing", is considered to be a very controversial agile practice. This technique requires two engineers working together. While one of them is actually writing the code, the other one is actively involved as a watcher, making suggestions and navigating the through process.

Being focused on both code and more abstract technical tasks, this team of two is expected to be more efficient, creating better software design and making fewer mistakes. Another benefit of this approach lies in spreading the project knowledge across team members.

However, this practice has often been accused of having a negative impact on the team's short-term productivity. The research shows that each task usually requires 15-60% more time[10], which is a major drawback of the approach. Yet, there are some opinions that the extra time is easily compensated in the long term through the overall higher quality of the software [11].

Conclusion

The Agile approach is often mistakenly considered to be a single methodology. Yet, there are dozens of methodologies and certain practices that have not been touched upon in this research.

Regardless of the exact methodologies and techniques they use, Agile teams have proven to increase profits 37% faster and generate 30% more revenue than non-agile companies[12]. Higher speed, flexibility, and productivity achieved through such approaches are the key drivers which motivate more and more organizations to switch to Agile.

Software engineering, being an extremely fast-paced industry, calls for flexibility and responsiveness in every aspect of project development. Agile methodologies allow for delivering cutting-edge products and cultivating innovative experiences, while keeping the product in sync with the market trends and user requirements.

However, there is always a place for diversity. Depending on your business requirements and goals, you might still benefit from using the Waterfall model or the combination of the two.

References

1. http://www.pmi.org/~/media/PDF/learning/pulse-of-the-profession-2016.ashx
2. http://blogs.gartner.com/it-glossary/project-management/
3. https://www.versionone.com/pdf/state-of-agile-development-survey-ninth.pdf
4. https://www.cs.umd.edu/~basili/publications/journals/J90.pdf
5. http://www.agilemanifesto.org/
6. https://www.scrumalliance.org/scrum/media/scrumalliancemedia/files%20and%20pdfs/state%20of%20scrum/scrum-alliance-state-of-scrum-2015.pdf
7. http://www.toyota-global.com/company/vision_philosophy/toyota_production_system/
8. http://www.poppendieck.com/
9. https://msdn.microsoft.com/en-us/library/aa730844(v=vs.80).aspx
10. http://www.cs.utah.edu/~lwilliam/Papers/ieeeSoftware.PDF
11. https://www.ibm.com/devops/method/content/code/practice_pair_programming/
12. http://www.pmi.org/~/media/PDF/learning/pulse-of-the-profession-2015.ashx

About AltexSoft

AltexSoft is a Technology & Solution Consulting company co-building technology products to help companies accelerate growth. The AltexSoft team achieves this by leveraging their technical, process and domain expertise and access to the best price-for-value Eastern European engineers. Over 100 US-based and 200 worldwide businesses have chosen the company as their Technology Consulting Partner..

US Sales HQ

701 Palomar Airport Road,
Suit 300, Carlsbad, CA 92011
+1 (877) 777-9097

Global HQ

32 Pushkinskaya Str.,
Kharkiv, Ukraine 61057
+38 (057) 714-1537