AGILE PRODUCT MANAGEMENT

Vijay Dwivedi
Product Owner, PMI-ACP®, CSPO®

Mohammad Arsalan
Senior IT Consultant

Harshil Goyal
MS-MIS, University of Arizona, Arizona

ABSTRACT

Agile is a philosophy, an umbrella term, which defines a general approach to software development. It emphasizes teamwork, collaboration, frequent delivery of products and an ability to respond to change early and quickly. Agile is a lightweight software development concept that is intended to replace the document-driven, heavyweight software development processes (e.g. waterfall).

Agile offers various flavors of frameworks and processes to redefine how softwares are developed, monitored and delivered. Some of them are Scrum, XP, Kanban, Feature driven development (FDD), Dynamic Systems Development Method (DSDM) etc., which are nothing but agile ways of managing a project. In Agile, some of the processes of frameworks can be combined to form a single process. E.g. Kanban board, Kanban Work-in-Progress (WIP) and Scrum can be combined to have a single process of working, that is, sometimes called Scrumban. Some organizations combine the components of Extreme Programming (XP) such as test driven development (TDD) and pair programming with Scrum. The flexibility of the agile makes it so popular and widely used in the industry ranging from Fortune 500s to startups. This paper will essentially focus on Scrum.

Key words: Agile, Agile Framework, Product Management, Software Development, Burnup Chart, Burndown Chart, Agile Principles, Agile Myths, Agile Anti-Patterns
http://www.iaeme.com/ijm/issues.asp?JType=IJM&VType=7&IType=4

1. AGILE VALUES

As given in the agile manifesto, it provides us with four major values:

1.1. Individuals and interactions over processes and tools

Agile gives due diligence to the team members and believes that team possesses the capability of delivering and driving the product to success. Agile encourages an empowered and self-sufficient team capable of taking decisions in the direction of product success.

1.2. Working software over comprehensive documentation

This doesn’t mean that Agile does not want any documentation. We all know that prior documentation and knowledge repositories help in new team members’ onboarding. Documentation certainly helps, but it should be kept to a minimum. Team members can always go back to the tasks, user stories to refer back to what they built in the past.

1.3. Customer collaboration over contract negotiation

In Agile, collaboration is a key focus area. So unlike traditional models in which customer came into existence at the very end of the process, agile promotes close customer collaboration. It even says to involve customer in daily standup meetings.

1.4. Responding to change over following a plan

In Agile, all the work is done in small iterations called sprints. After each iteration, a demo of a small part of the work is given to the customers. If they don’t like the product they can reject or suggest some changes. Instead of following a frozen plan shaped at the start of the lifecycle, agile evolves over the period of time and incorporates changes.
2. AGILE PRINCIPLES

There are 12 agile principles which moreover align with 4 agile values.

<table>
<thead>
<tr>
<th>Regular Adaptation</th>
<th>Close Cooperation</th>
<th>Technical Excellence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self Organizing Teams</td>
<td>Sustainable Development</td>
<td>Simplicity</td>
</tr>
<tr>
<td>Face-To-Face Conversations</td>
<td>Working Software as progress measure</td>
<td>Motivated Individuals</td>
</tr>
<tr>
<td>Changing Requirements</td>
<td>Frequent Delivery</td>
<td>Customer Satisfaction</td>
</tr>
</tbody>
</table>

Figure 1 Agile principles

3. AGILE IN A BIGGER PICTURE

There are 3 most noticeable roles in Agile: Team (cross-functional), Product Owner and Scrum Master

3.1. Team

Agile team is a cross functional team of 3-8 people who are capable of performing development, testing, designing and requirement elicitation tasks. Ideally, any team member should be capable of playing any of the roles such as testing, coding and business analysis. But in real life it is not feasible. Hence, even agile teams which are expected to be cross-functional are not purely cross-functional. In real world agile, one will still find different roles played by different members of the team—Designer, BA, Developer and a QA.

3.2. Product Owner

Product Owner is a very challenging and new role; in the sense that it combines multiple roles into one. In traditional projects, there used to be 2 roles: Project manager and product manager. Project manager is a more inward facing role, which focuses on project deadlines, processes and people management to make sure the project is meeting the schedule and cost. Product management is more outward facing role, which focuses on product requirement & design, product ROI, product marketing and client issues resolution, etc. Product Owner to some an extent bridges these 2 roles in 1.

3.3. Scrum Master

Scrum Master is a unique and a different role. Scrum master acts as a guide and a coach who helps the team in day-to-day operations. Scrum Master protects the team from external influences and also helps in conflict resolution among the team members, between different teams, etc. Scrum Master acts as a guardian, and usually follows a servant-leadership approach instead of an authoritative approach. The overall goal of the agile is to create self-empowered and self-driven team that can take decisions on its own
with minimal to no supervision, and Scrum master promotes that culture. The scrum master’s role is not necessarily a technical one. Any person with some good experience in handling teams can transition easily to this role.

4. PARADIGM SHIFT

In agile, the well-known iron triangle is reversed, that is, time and cost are fixed but the scope is derived. It is a marginal shift from conventional product management approach where Scope (requirements) are fixed and time and cost are derived based on that scope. But, in Agile things have been evolved to accommodate changing requirements (30-45% of project volatility can easily get absorbed)

![Figure 2 Paradigm shift](image)

5. AGILE IN ACTION

5.1. Project/Product Kickoff Activity

Even in agile projects during the initial phases, a lot of things are discussed between the stakeholders, sponsors and the higher management. Based on that discussion, a vision statement is chalked out for the project which delineates what the project is, who the project is for, how it is unique and what the project does. Once the vision statement of the project is written down, it is the responsibility of the product owner to share that vision with the team and get everyone excited, enthusiastic and on-boarded with project/product vision and mission. This marks the kick-off for any project and paves way for actual product development.

5.2. Iterations/Sprints

As stated earlier, in Agile with Scrum, work happens in iterations. These iterations are also called sprints. Ideally, these sprints are of 1-4 weeks duration. During these iterations, entire product lifecycle is ‘somewhat’ followed; ‘Somewhat’ because some activities like *product planning, vision and goal identification, product roadmap are done outside these iterations*. During a normal agile iteration, only requirements and designs are discussed, implemented, tested, delivered and accepted.
5.3. MVP

Agile supports incremental and continuous delivery of products. So at the end of each iteration, which is also called a sprint, we have a potentially shippable product increment (PSPI) or Minimum Viable Product (MVP).

![Figure 3 Incremental delivery in agile](image)

6. AGILE IN A NUTSHELL

Agile iteration starts with requirement gathering and analysis which is done in parallel with design discussion. It is then closely followed by coding and testing. Some organizations follow somewhat waterfall approach within an agile sprint but it is highly recommended that design, coding and testing all happen in parallel. A developer can go back and forth with the designer to get the design right and then implement it by testing his code intermittently throughout the iteration. In some cases, a developer can even follow Acceptance Test driven development (ATDD) approach of working too - writing failing test cases first and then writing code that meets acceptance criteria.

Agile SCRUM diagram depicts this entire agile process in a more succinct way. At the core of Agile, we have product backlog which contains entire product items. Based on the team discussion, discussion between Stakeholders and product owner things get added to the product backlog. It is the responsibility of the Product owner to keep the product backlog updated.

In Agile, product items are progressively elaborated (rolling wave approach) which means the items on the top (about to get worked upon) contains more information, and have more clarity compared to the items in the bottom of the backlog.
In Agile too, things go through proper planning and there are pre-defined phases in which products are developed and managed. In the **product planning** phase, PO and stakeholder sit down and write down the vision of the product. They should possibly discuss the strategy of the product as well at this time. Vision of the product should be clear, crisp and concise. Ideally vision should be so simple and comprehensible that it should pass the Elevator test --possible to deliver the summary of vision in the time span of an elevator ride, approximately thirty seconds to two minutes (*widely credited to Ilene Rosenzweig and Michael Caruso*)

Next Inner phase is **release planning**, during the release planning what is to be accomplished in a release is discussed with the team at a higher level. Ideally, a release comprises 4-6 sprints worth of work. During this meeting, a collective release vision is shared with the team and potential high level blockers and impediments are identified. This phase outlines when the release will happen, and it also communicates to everyone as what all is expected in the release. If any external help is required to achieve the
release goals, those are also outlined and a plan is created to mitigate those risks.

Innermost phase is sprint planning during which a sprint or iteration worth of work is discussed with the team. During this meeting, PO comes up to the team with a list of highly prioritized work item. Team discusses those work items among themselves, clarifies issues/doubts with the PO and then creates task/plan of attack (POAs) for each work item. Post that team collectively enters into an estimation activity. In the estimation, only team members are involved -PO and Scrum master are not involved. Team sizes/estimates the story based on the comparative complexity of the story.

\[
\begin{align*}
A & \text{ (Point } = 1) \\
B & \text{ (Point } = 3) \\
C & \text{ (Point } = 5)
\end{align*}
\]

Figure 6 Estimating related items (B and C) keeping A as the baseline

With comparative we mean, if a most simple story (adding a name field on the webpage) is given 1 story point, and then all other stories will be given points keeping the complexity of 1 Pointer story as a baseline. E.g. Here work item B and C are somewhat related to A, and A is 1 story point worth. Then based on team’s expertise they can estimate B and C to be 3 and 5 point respectively. Generally, it is seen team following Fibonacci series \((1, 2, 3, 5, 8, 13, 21\ldots)\) to estimate stories. Though, one can use any other method of estimation to size the story.

If the story is very complex and the team rates it very high, say, 13 or 21 points then it is recommended to slice that story into smaller stories. If the story asks to implement some new concept then create separate research stories to research that concept: basically, separate learning of a new concept from implementation of that concept. This will help to slice down such an unknown story.

7. AGILE PRODUCT MANAGEMENT

Product owner manages the product and is also responsible for the ROI of the product. 7.1 For managing the product well, PO needs to talk to the customer, follow through product analytics, perform product research, do trend and competitive analysis to make the product market-ready.

7.2. PO also acts as a glue between the team and the customer/user. It is also the responsibility of the PO to funnel the feature requests he/she is getting from the team and stakeholders.
7.3. Product Owner decides which feature the team should build to gain the maximum advantage in the market. Based on his/her research, PO will be able to weed out trivial feature requests of stakeholders and Team.

*Remember agile says - Simplicity--the art of maximizing the amount of work not done--is essential.*

7.4. Successful agile product management is a collective team effort with Product Owner being in the steering seat taking inputs from multiple channels, projecting the product in the direction of success. And PO & team should do so iteratively and continuously using PLAN, DO, ACT and CHECK method.

### 8. AGILE INFORMATION RADIATORS

Information radiators are a big visible display that communicates to people the progress of the project. As people walk by the information radiators, they can easily see the progress without any interruptions. There are basically two types of chart that are widely used as information radiators to track and communicate the progress. A burndown chart, which shows how much work is remaining in the project, and a burn up chart that shows how much work has been completed.
Figure 8 Burnup and Burndown chart

Burnup chart, in simplest form, consists of a line which tracks the work completed and shows what amount of work is left to be completed. Whereas burndown chart tracks the work that needs to be completed. The vertical axis, in both these charts, defines the amount of work, which might be defined as number of tasks, estimated hours or simply the story points. The horizontal line which shows the time scale may be defined in terms of hours, days or week. Other information radiators can be storyboards, Kanban boards etc.

Why use a burn up chart?

A burn up chart clearly shows work completed and overall project scale. It is an effective tool for reassuring stakeholders and clients that a good progress is being made. Burn-up communicates to them how any additional feature requests they make will affect the deadline. A burn up chart can make clients reconsider whether they really need an extra feature.

Why use a burndown chart?

Burndown charts are simple to understand and are straightforward model to see that the number of tasks must reach zero by a defined date. They are easy to determine whether the goal will be reached in time or not. Therefore it is often better to create a burndown chart for use in presentations and demonstrations to clients and management.
9. AGILE ANTI-PATTERNS
These guidelines will help in identifying when agile is not being followed. It is good that we tweak the process to suit our needs but we need to do that carefully.

- **Following a one week sprint by cutting down few meetings such as sprint retrospectives and demo**
  Cutting down meeting will help in releasing the product early. But soon one will realize that the overall team’s velocity is coming down after 4-5 sprints. Since agile absorbs project’s volatility, it does give rise to discontent within the team over the period of time. Sprint retrospectives and reviews help in venting out discontent to some degree and re-establishing good practices.

- **Having same Product owner and Scrum master**
  As discussed earlier, Product Owner and Scrum Master are two very distinct roles and each play a very different and crucial part for the success of the product/project. These roles if combined will be a lot of responsibility for same individual. More than work, it is about the skillset. Each role requires a different skillset and expertise.

- **Different Product backlogs for different teams (Leading to component teams)**
  Many organizations have a different backlogs for different teams. This is an agile anti-pattern and it leads to breaking up of teams into component teams. If one has 2 teams - a frontend team working on UI and a backend team working on associated server side scripting, then agile suggests to have a common backlog for both the teams. Probable solution of this problem: Create a single story for both these teams and that story should have tasks for each team. So, when the story is complete only then it is available for testing, otherwise testers will be testing 2 different stories or will be waiting for both the stories to finish. A single story and a common backlog helps in better progress tracking, maintaining good agile cadence and better product management.

- **Improper estimation of stories (False Burndown charts)**
  Team should be given ample time to go through the stories before coming to the sprint planning meeting. Team should not be rushed through the requirements. This will lead to skewed estimation and team will never be able to get to know their true sprint velocity* (*number of story points completed in a sprint)

- **Changing priorities within the sprint (Making agile fragile)**
  Every team member wants to see the product to see the day of light - move to production. If the priorities keep on changing, it not only affects the team’s velocity but also the motivation and momentum of the team. No change is bad but a perpetual loop of re-work can affect the team greatly and can also lead to disgruntled employee - leading to employee turnover.

- **Keeping QA members in the silos**
  Although, Agile suggests to have well versed cross functional teams but still the roles are not yet diluted at least in Indian IT sector. So there are still QAs, Developers, UI/UX team members. QA’s are one of the most important team asset and they should be involved in every product discussion so that they know
the overall product picture. Otherwise they will only be testing the product from a very narrow point of view.

- **No value creation with the story**
  Story should follow INVEST (independent, negotiable, Value, Estimable, Small, Testable) approach. In industry terms, story needs to be vertically sliced, that is, story should be complete and should add value to the user. Having a login field that doesn’t validate the data against the database has no value to the end user. A vertically sliced story will be ‘Having a username field functionality that verifies the username’. This story will touch base with entire system (UI-->Server-->Database-->Server-->UI). This approach is analogous to slicing the cake vertically which lets you enjoy all the flavors of the cake.

10. COMMON MYTHS ABOUT AGILE

- **Agile means that one needs to follow a set of best practices**
  One needs to take into account whether it fits into a project or not. It is generally good practices and not the best practices

- **An iterative waterfall is Agile**
  Agile focuses on providing valuable or shippable features after every sprint and not just a chunk of work optimal for the development team

- **Agile is a new concept**
  The roots of agile goes back to late 1980’s, but the Agile Manifesto was published in 2001 by a team of 17 software developers

- **Agile can be understood by reading books/articles**
  Practical experience is essential to enable an agile mindset

- **Agile means SCRUM**
  Agile is a mindset and some of the frameworks that use Agile are Scrum, Kanban, Extreme Programming, etc.

- **A user story is a mini-specification document**
  User story is a placeholder for future conversation and consists of acceptance criteria which when met leads to the success of a user story.

- **Reducing the scope of User story to pass it**
  If a story is not completed in a sprint, it should be marked as incomplete and a new user story should be created for it with resizing of that user story

- **There is at least a team lead, or/and a developer, or/and a quality analyst in Agile**
  There are only 3 roles in Agile i.e. Scrum Master, Product Owner and Team Members. When the size of a team is considered, Scrum Master and Product Owner are not counted in it. The reason for not having roles like developer or quality analyst is that agile advocates cross functional team where depending on business requirement, a team member can work in any role
**User story should be sized similarly by all teams**

Sizing of user stories is a relative concept and different teams size user stories differently. Also, some teams size stories using Fibonacci series (1, 2, 3, 5, 8, 13 and so on), some size using T-shirt sizes (extra small, small, medium, large and so on), some size using relative mass valuation, etc. It does not matter which method for sizing is chosen as long as there is consensus in a team.

**A team cannot miss user stories in a sprint**

It is alright to miss user stories/ story points in Agile, if justified by valid reasons. Example- If one of the team member goes for an unplanned leave, then there is no way that his/her planned sprint work can be accomplished in that sprint.

**Individual is responsible for an incomplete user story**

In Agile, a team has collective ownership, so if a story is not completed then it’s the team that misses story points and not one or two team members.

**REFERENCES**

**Books:**

[2] Roman Pichler’s Agile Product Management with Scrum: Creating Products that Customers Love
[5] Mike Cohn Succeeding with Agile
   http://www.agilemanifesto.org/
[8] Agile Alliance
   https://www.agilealliance.org/
[9] Agile Software Development
   https://en.wikipedia.org/wiki/Agile_software_development#The_Agile_Manifesto
[10] Mountain Goat Software
    https://www.mountaingoatsoftware.com/