ACTION RESEARCH PROJECT REPORT

Agile Practices for

Software and Systems Engineering

Ashish Tiwari
Scientist-D
Electronics and IT Department
Bureau of Indian Standards

Project No: AR/0096
26 Aug 2021

Contents

1 Introduction

Organizations are required to face the continuously changing market requirements and need to be supported by the underlying software. The process of software development and maintenance require agile methods to fulfil the ever-growing needs of the changing market and at the same time, meet the characteristics of the software quality.

Since the birth of Agile Manifesto in 2001, Agile methodology has become increasingly popular over years. Agile implementations have brought in tremendous value to the customers such as high degree of collaboration between customer and project teams, transparency, focus on business value, allowance for frequent changes, early predictable delivery etc.

Over a period of time, a large number of frameworks have evolved around Agile Principles and Values. Also, while Agile methodologies revolve primarily around project execution, there are related software engineering processes such as Planning, Estimation, Budgeting etc. which need newer line of thinking and a level playing field among all stakeholders. Also, there are core engineering processes such as Software Architecture which also need more attention, given the nature of constantly changing Agile Requirements.

Various countries have taken steps to come up with guidance on different aspects of Agile implementations.

This action research explores the international standards and literatures available around various Software Lifecycle Processes and provides a brief guideline on how Standardizing Core Agile Practices will be useful for Software and IT Industry.

1.1 Why Agile?

All software systems have a lifecycle, irrespective of whether it is established formally or not. A lifecycle is a set of distinguishable phases or stages that the software goes through from its conceptualization until it ceases to exist. Lifecycle models are often considered as a framework of processes and activities that serve as the baseline for understanding and communicating about the software system. Software systems typically passes through its lifecycle as the result of tasks being performed by organizations and people, by instantiating a framework of processes and activities for their respective contexts. IS 16124 : 2020 (Identical to

ISO/IEC/IEEE 12207 : 2017) standard provides requirements related to a common process framework for describing the life cycle of software systems. These processes can be used by any organization to construct life cycle models for their product and service offerings.

The Agile way of software development has become a necessity due to ever changing nature of business requirements that defines the software. The Agile Manifesto and related values and principles have served as reference for agile practices across many organizations. However, there is no uniform adoption of agile practices in the industry. Different organizations exhibit different levels of maturity and diverse ways of interpreting, adopting and implementing these practices. The Agile practices offers a lightweight, adaptive, and collaborative development approach based on empiricism with the focus on rapid business value delivery.

2 Scope of work

- Study existing literature on Agile Practices, standards on Life Cycle Processes for existing models in Software Developments
- Identify the core practices involved in the agile methodology of IT related processes.
- Develop a brief guideline for implementing Agile in an organization

3 Methodology used in the work

- Studied the existing literature in the area of agile practices like ScRUM, Norwegian Standards etc.
- Held number of discussions with domain experts in the area through virtual meetings
- The discussions and literature study provided a better understanding of the common concepts in the area of systems (software) processes.
- Categorization of Agile Practices.
- Identifying Core Agile Practices which will be helpful for the industry.
- Identifying key aspects of Agile implementation in business scenarios
- Necessary recommendations for implementing the agile practices
- Based on the above findings, a recommended template for drafting standards on core agile practices was prepared.

4 References

- IS 16124 : 2020 /ISO/IEC/IEEE 12207 : 2017 Systems and software engineering — Software life cycle processes
- ISO/IEC/IEEE 24765 : 2017 Systems and software engineering — Vocabulary
- ISO/IEC/IEEE 26515 : 2018 Systems and software engineering — Developing information for users in an agile environment

- Collaborative Enterprise Architecture: Enriching EA with Lean, Agile, and Enterprise 2.0 practices" by Bente Stefan, Bombosch Uwe, Langade Shailendra (2012)

5 Summary Literature reviewed

As part of the action research, the following international standards were reviewed to understand the way each of the standards defines and describes reference architecture in the subject area:

a) IS 16124 : 2020 Systems and software engineering — Software life cycle processes
b) ISO/IEC/IEEE 26515:2018 Systems and Software engineering — Developing information for users in an agile environment
c) ISO/IEC TR 29119-6:2021 Software and systems engineering — Software Testing — Part 6: guidelines for the use of ISO/IEC/IEEE 29119 (all parts) in agile projects

6 Agile Practices – an Overview

The term Agile was coined in the *Agile Manifesto* (Beck et al., 2001). This set of four values and twelve principles was formulated to replace the inflexible and heavy-weight waterfall model for software development with a more lightweight, efficient and human-centric approach.

The four values of the Agile Manifesto are stated as:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The twelve principles of the Agile Manifesto outline ways to implement the four key values.

There is no such thing as the Agile practices, although many practices ascribe to Agile – Scrum, XP, Feature-Driven-Development, and so on. The Agile Manifesto simply expresses their essence in four key values and twelve principles.

There are certain common characteristics across all Agile flavors:

1. **Iterative, Time-boxed Approach**: Software is developed in cycles of a fixed length duration, most often, between two and four weeks long. Goal of an iteration is to deliver a feature that is completely "done" i.e. a piece of software is fully tested and can be deployed. An iteration is never prolonged.
2. **Working Software**: The short development cycles are accompanied by strict governance on quality. Working Software implies keeping the system always in a deployable state. Therefore, daily build, test automation and continuous integration are central concepts in Agile projects. The ultimate goal is to have the software in a fully deployable state after each iteration. Furthermore, a feature is complete if it has fully

satisfied its "definition-of-done" and is available to the customer for demo (having visual results) at the end of the iteration.

3. **Welcoming Change**: The iterative approach helps in shaping the users and stakeholders' requirements over time. A customer representative is present at regular Agile meetings. It helps in seeking customer input for each iteration planning. It is also a way to respond to the accelerated changes in the business and technology environment and reprioritize development activities. This enables development projects with a time horizon of two or three years to catch up with these changes along the way at much shorter time intervals.
4. **Self-organized Teams**: An Agile project must always be executed by a cross-functional and self-organized team. The team takes common responsibility for delivering the software on promise. Each team member must fulfill this responsibility in an end-to-end fashion. The team organization is flat. There are only two predefined, exposed roles: a) The scrum master, responsible for moderating (not managing!) the team and protecting it from outside disturbances; b) The product owner, acting as a representative of the project customer. The specialist roles like an architect are subject to some debate in the Agile community.
5. **Customer Collaboration**: Agile favors direct communication over written documents. Flat organization, mutual respect, open communication characterizes Agile teams. This is more natural as humans communicate most effectively and efficiently when coming together in a trustful atmosphere, and expressing their insights and opinions – not only through words, symbols and drawings, but also through gesture, facial expression and voice modulation. The customer plays a critical role in the success of the project, more specifically, due to his continuous engagement for a) providing insight and input in backlog prioritization and iteration planning; b) reviewing customer demo and its visual results at the end of each iteration; c) providing overall user feedback and advice

6.1 Lifecycle processes

Every software product or service has a lifecycle IS 16124 : 2020 (Identical to ISO/IEC/IEEE 12207 : 2017) describes this life-cycle as an abstract functional model that represents the conceptualization of the needs, realization by addressing the needs, subsequent utilization after realization, evolution based on change in environment or evolving needs and retiring or disposing the software product or service at the end of its useful life. IS 16124 : 2020 proposes 4 interacting process groups :

a) Agreement processes,
b) Organizational project-enabling processes
c) Technical management processes
d) Technical processes,

covering the activities that are typically performed during the software systems lifecycle.

The agreement processes aid organizations to agree on the responsibilities of the different functions related to software during its conceptualization, realization, utilization, evolution and subsequent disposal. The organizational project-enabling processes enable an organization to acquire and supply software through the initiation, support and control of projects. The technical management processes aid organizations to manage resources and assets provided by

the organization to meet the agreements and other organizational commitments. The technical processes aid organizations to understand requirements, transform them into appropriate software products or services, reproduce software as and when needed, utilize the software in different environments, sustain the software till they are decommissioned, and retire or dispose the software at the end of its useful life.

6.2 Agile in a Project Context

Figure 1 positions Agile Development within the context of an end-to-end project lifecycle. This is how IT folks (i.e. the supplier or service provider) would look at Agile Development from an IT delivery perspective.

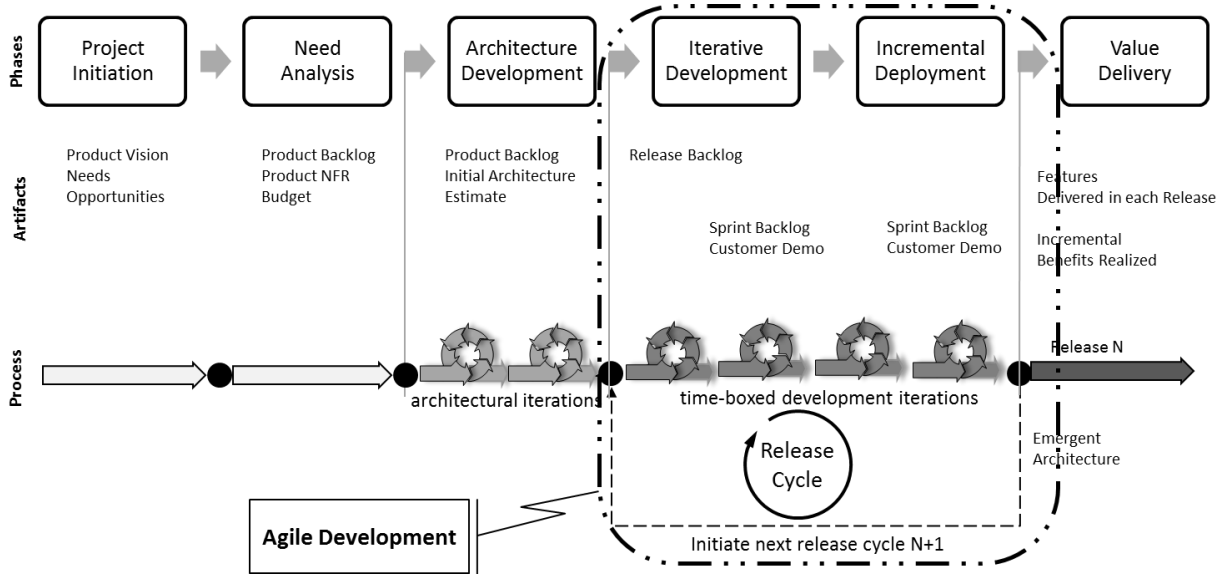

Figure 1 Agile in a Project Context

7 Areas for Standardization

Agile Manifesto, which a set of four values and twelve principles, is the foundation for Agile Development. Many different methods and techniques help in practicing the values and principles set forth by Agile Manifesto, but there is no such thing as Agile practice or Agile methodology.

In Agile literature till date, there are a number of terms, concepts, techniques, methods and artefacts which are being used, referred and found relevant. Nonetheless, the interpretations thereof are still broad and generic. The following sections highlight some of such areas which are worth considering for standardization.

7.1 Agile Glossary of Terms

A number of terms are being used in Agile with or without a formal definition or notion. It is largely left to individual interpretation as to what they mean. For example, Continuous Integration, Epic, Minimum Viable Product and so on.

It therefore becomes necessary to establish and expand the Glossary of Terms used in Agile Environment. It may further become essential to supplement certain terms with a few illustrative examples.

7.2 Key Artifacts

A number of artifacts are used in Agile Environment, however there is no mention as to what the content should be, how the content should be structured, what meta-data about the artefact should be captured, and how the lifecycle of an artefact be managed. A few illustrative examples of artefacts which are largely vague are as follows:

1. **Business Case**: what a business case should contain, in what format and at what degree of details; how to make it lightweight; who prepares it; who consumes it; how it will help in product/project evaluation, budgeting, pricing, costing, and other related downstream activities.
2. **Product Vision**: what is the purpose of this artefact, who should prepare it, who should use it and for what reason, what it should contain and in which structure or format, what constitutes a good product vison.
3. **Product Backlog**: how the backlog can be split, decomposed, merged and synchronized at different stages of development cycle e.g. product backlog vis-à-vis release backlog vis-à-vis sprint backlog; how each of the backlogs be characterized in terms of content, content type, ownership for e.g. Epic, Feature, User Story, Task; how different backlogs should be mapped or linked together.
4. **Product Non-Functional Requirements**: how the NFR for the product be defined; At what different levels of architecture the NFR can be imposed, for e.g. product-level, component-level, function-level, task-level.
5. **User Stories**: how the user-stories should be organized in the backlog and at what levels of granularity (e.g. Epic, Feature, User Story, Task), what meta-data about the user-stories be maintained; how the user stories be categorized, how user stories are detailed from a high-level abstract description to a precise implementable description of the requirement. How the conditions of satisfaction and definition of done for user storied be maintained in the backlog, how the lifecycle of user stories be managed within the product backlog.
6. **Contract**: what different types of contracts are relevant in Agile Environment; when to use what kind of contract; what an agile contract should contain; who should negotiate the contract and who should sign it (i.e. the roles accountable for contract execution); how the risks are balanced between the customer and the supplier; standard and specific terms and conditions in an Agile contract

7.3 Methods and Techniques

A number of methods and techniques are being used to address specific concerns in Agile Environment. There is however not much guidance or formalization to help in bringing consistency in practicing them. These methods and techniques are candidates for standardization:

1. **Estimation**: In Agile, estimation is done primarily on the basis of user story or size of the user story – story points. There are a few popular estimation methods such as T-Shirt Sizing, Planning Poker and Affinity Diagram. For a longer-term, the estimation is ball-park (e.g. T-Shirt Size) whereas for a near-term, it is more precise (in story points). It is necessary to formalize which method of estimation should be used at what stage of the development, and how. It is also important to understand the degree of variance achievable and acceptable at each stage of the development.
2. **Budgeting**: In contrast to traditional software development, Agile takes a different view on Budgeting. It assumes schedule and budget to be fixed upfront while leaving the scope of the deliverable changing. It therefore becomes necessary to allow the Agile teams to flexibly consume the allocated budget in an attempt to maximize the value delivered. It is somewhat hard for traditional organizations to allocate the budget without clear commitment requirements to be implemented and/or activities to be performed.
3. **Architecture Development**: Agile is averse to upfront architecture development exercise and presumes that the architecture shall evolve during the development – code is the architecture. The idea of emergent architecture is not scalable and often fails in large projects. It therefore becomes necessary to formalize what initial architecture should be in place, when it can be changed and how it will evolve over time without impacting the already released software much.
4. **Continuous Prioritization**: The work to be done – the user stories, the tasks – is continuously reprioritized to better align with the user expectations and to maximize business value for the customer. The prioritization can be for a list of projects or for the work-items in a backlog. It is necessary to formalize the periodicity of prioritization; the basis for prioritization; the stakeholders who will participate in prioritization; the methods and techniques to be used for prioritization; overall governance for effectiveness of prioritization on the achievement of strategic goals and benefits.
5. **Continuous Integration**: An Agile project should maximize automation to accelerate development process. In particular, continuous integration implies a tool-supported automation of software build and test execution, which is repeated in continuous loops. This way, code that causes compilation errors or test failures, is noticed with only a minimal delay after checking it into the source code management system. The methods, techniques and technologies available for continuous integration, test automation, release automation and DevOps in general need some degree of technology-agnostic formalization.
6. **Contracting**: The process of contracting between the customer and supplier should be tailored and formalized to effectively work in Agile Environment. The risks should be mitigated at the best or well-balanced between the customer and the supplier at the least.

The processes followed, the stakeholder involved, the terms and conditions need to be standardized based on the type of project work being contracted out. This is especially tricky in Agile Environment for the lack of upfront clarity about the scope, the requirements and the acceptance criteria for the contracted project work.

7. **Benefit Quantification and Governance:** There is a need for strategic governance in Agile software development. It implies an ongoing monitoring and control of Agile projects and programs on four dimensions: a) solution maturity (architectural excellence), b) delivery assurance (process excellence), c) organizational readiness (people excellence), and d) business value creation (business excellence)
8. **Scaling Agile**: While Agile methods and techniques work pretty well in small setup such as innovation labs, and startup companies, managing Agile at a scale is always a challenge. Many different approaches are suggested for scaling Agile projects like scrum of scrums. It is necessary to study different scaling options and assessing them for operational feasibility, management overhead, cultural conflict, geographic spread, and so on. Based on the study, optimal models for Scaling Agile can be provided.

8 Proposed structure for standards describing Core Agile Practices

1. Foreword
2. Overview
3. Need for the standard
4. Purpose
5. Scope
6. References
7. Terminology
8. Conceptual foundations
   a) Lifecycle processes
   b) Agile practices
   c) Criteria for practices
   d) Description of practices
9. Agile practices
   a) Practices under agreement processes
   b) Practices under organizational project-enabling processes
   c) Practices under technical management processes
   d) Practices under technical processes