DOI: 10.15439/2020F46
ISSN 2300-5963 ACSIS, Vol. 21

Towards a Better Understanding of Agile Mindset by Using Principles of Agile Methods

Necmettin Ozkan
Information Technologies Research and Development Center
Kuveyt Turk Participation Bank
Kocaeli, Turkey
necmettin.ozkan@kuveytturk.com.tr

Mehmet Şahin Gök
Department of Business
Gebze Technical University
Kocaeli, Turkey
sahingok@gtu.edu.tr

Büşra Özdenizci Köse
Department of Business
Gebze Technical University
Kocaeli, Turkey
busraozdenizci@gtu.edu.tr

Abstract— **The right way to agility should start with a proper agile mindset instead of applying Agile methods directly. However, apart from the manifesto, it is unlikely to find a comprehensive set of Agile principles that can serve for an improved agile mindset. Our study intends to fulfill this gap in a systematic way: providing a list of the Agile methods along with their principles within a single source, in the way of providing a better understanding of the concept of agility from a wide and exhaustive perspective. To do so, the collected 105 principles were content-analyzed in order to group them into 32 categories for a higher-level abstraction. These categories then were subsumed into two main categories. The whole grouping process was reviewed by one expert and the list was adjusted accordingly. Then, based on the consolidated list of the categorized principles, analysis and evaluations were made by the authors. As a part of the evaluations, semi-structured interviews with two experts were conducted to evaluate the categorized principles in general, especially in terms of their contribution to agility.**

Keywords— agile mindset, agility, agile methods, agile frameworks, principles

I. Introduction

Effective agile individuals, teams and organizations require a particular attitude, way of thinking and behavior so called as agile mindset, beyond the given set of procedures, techniques and rituals [12]. By applying a set of Agile practices of (a) particular Agile method(s), there is no guarantee to utilize the agile mindset properly [1]. The practices offered by the Agile methods have some fundamental limitations in nature; they are not adequate to cover possible agility capabilities fully and also they are very static in providing the ability of adaptation to changing situations. Indeed, the right way to agility should start with a proper agile mindset instead of applying Agile methods directly. Principles, as "a basic belief, theory, or rule that has a major influence on the way in which something is done" (macmillandictionary.com), support any mindset more effectively than practices. This emphasizes a proper understanding and locating the principles first and foremost, before the practices.

In terms of providing Agile principles, the Agile Manifesto is the most well-known set in supporting the Agile mindset in a formal sense via its values and principles. Apart from the manifesto, it is possible to find different sets of Agile principles scattered in the literature, which makes it hard to reach a comprehensive list. The existing literature has been reviewed in our study to find out the most possible comprehensive list of the Agile methods. It shows that there has been no study aiming to disclose the principles or principle-like features (referred to as "principles" across the study, if not stated otherwise) of the methods. Our study in particular intends to fulfill this gap in a systematic way: providing the most possible complete list of the methods along with their principles within a single source from a wider and exhaustive perspective, in the way of providing a better understanding of the concept of agility.

In addition to exhibiting the known attributes of the methods, differently, our study aims to reveal some analysis through the consolidated list of the principles (Level 1/L1) such as grouping them into categories (Level 2/L2) along with the classification of these categories at a higher level (Level 3/L3), an analysis on the methods and their principles (L2), the contribution of principles (L2) to agility and more, with referring to expert opinions to get more sound basis. In particular, taking advantage of reaching out to such a scope, the principles provided by the methods (L2) and the principles of the Agile Manifesto are mapped, in terms of coverage. Consequently, the research questions are formed as follows:

RQ1: What are the Agile methods in the literature?

RQ2: What are the principles offered by these Agile methods?

RQ3: Is there a match between the principles of the methods with the manifesto principles or not?

RQ4: What do the principles (L2) represent in general, and also about their contribution to agility in particular?

The rest of this paper is organized as follows: Section 2 delivers the background for the methods and the definition of agility for software solution development. Section 3 elaborates related works and Section 4 depicts the research design. Section 5 delivers findings and analyses made for the set of the principles. Section 6 evaluates findings and analysis with the consideration of the feedback from the interviewees. Finally, Section 6 presents conclusions and future work.

II. Background

A. A Brief History of the Agile Methods

As a cornerstone of the Agile methods, iterative, evolutionary, and incremental development roots go back decades [2]. It grew from the 1930s' work proposing a series of short "plan-do-study-act" cycles for quality improvement and was involved in software projects such as NASA's Mercury in the early 1960s, with practices like time boxing, test-first development [2]. One of the early traces of the Agile principles were also witnessed in the work of the Tavistock Group, which conducted research on the self-organizing teams of British coal miners in the 1950s [3]. It is worth mentioning that, in 1976, Tom Gilb introduced evolutionary project management as the first clear flavor of the Agile methods [2].

One of the early known works defining the self-organizing teams is Takeuchi's study [4], inspired by the Toyota production system. In the study of Morgan [5], he argues that an organization can improve its ability to self-organize through the holographic brain metaphor. In 1988, Gilb published a new book, Principles of Software, which describes the Evo method (chronologically the first method in our study).

Apart from these prominent milestones, throughout the 1970s and 1980s, there are some other publications and specific projects partially integrating the Agile practices. While people from the previous decades incorporate a preliminary major specification stage with the teams utilizing iterations with minor feedback, differently in the 1990s, the mainstream of Agile initiatives became preferring less early specification work, rather a stronger evolutionary analysis approach [2]. In this decade, unlike the previous ones, the agile mindset and practices started to take a form within certain formal methods/frameworks (hereinafter and heretofore referred to as "method"), such as Scrum, Dynamic Systems Development Method (DSDM), Rational Unified Process (RUP), Extreme programming (XP), Feature-driven Development (FDD), which later referring collectively to Agile Software Development Methodologies.

The quest for a full-fledged agile mindset ended up in the meeting in a ski resort in Utah, in the year 2001, where the well-known techniques from some "Agile Software Development Methodologies" were combined within the manifesto for the Agile Software Development [6]. Those well-known methods that had an influence on the manifesto include DSDM, TDD (Test-Driven Development), ASD (Adaptive Software Development), D3 (Design Driven Development), Scrum, Crystal, XP, Pragmatic Programming, FDD [6]. From this period of time to today, the interest in Agile has continued increasingly and various other methods have been presented.

B. Back to the Basics: the Definition of Agility in Software Development Domain

The understanding of the word "Agile" varies [7], even among prominent Agile pioneers. For instance, Alistair Cockburn defines it as "being effective and maneuverable" [8]. Kruchten's [9] definition is "the ability of an organization to react to changes in its environment faster than the rate of these changes". Conboy and Fitzgerald [10] state agility as "the continual readiness of an entity to rapidly or inherently, pro-actively or reactively, embrace change, through high-quality, simplistic, economical components and relationships with its environment the continual readiness of an entity". Highsmith defines agility as "the ability to both create and respond to change in order to profit in a turbulent business environment; it is the ability to balance flexibility and stability [11]. Instead of using such existing definitions, we would rather like to present a revised definition of agility inspired from the definition in [42], to communicate a better understanding of the background of our mentality used throughout this study.

We see "responding to change" as the widely recommended feature of agility. At this point, questions arise: change of "what"; inconsistent customer requirements, analysis documentation or changes in the environment? Therefore, it is better to define agility based on the closest point to the source of the change, which is the reality itself, instead of from the view of customers, for instance. Users or customers are a kind of proxy of reality and the same as documentation as a proxy of the system being developed, not the reality itself. From another point of view, the definitions similar to of Kruchten ("the ability of an organization to react to changes in its environment faster than the rate of these changes") take us to a passive position of re-acting. Although information technology has traditionally taken a passive position throughout its history, as it has been seen as a business-driven body, it is not a common rule beyond ages. Thus, these two points bring us to a new definition of agile; "the ability to move quickly and easily" (where Cambridge, Oxford and Macmillan dictionaries achieve consensus for this part of the definition), to adapt to changes of the reality or to create changes becoming the reality, let us say in the domain of software solution development.

III. Related Work

There are plenty of studies reviewing the Agile methods, comparing them with their characteristics, strengths, weaknesses, similarities and differences, providing criteria to choose them according to the context of development, generally provided in an informative way. Our study rather intends to provide a complete list of the methods along with their principles. In addition to exhibiting the known attributes of the methods such as their principles, we aim to reveal some patterns through the consolidated list of the principles such as by grouping them into categories, the classification of categories, the frequency of principles, the contributions of principles to agility and more. Expert opinions are involved in interpretation-intense sections to get more sound determinations. Hence, this study provides a wider perspective to the concept of agility by revealing all possible Agile methods and their principles in a single picture along with analysis, resulting in inputs for a better understanding of the agile mindset.

The majority of the works on the agile mindset are satisfied by only mentioning the term as a "fixed concept" without actual descriptions, details, explanations or definitions [1]. As witnessed by the study of Mordi and Schoop [1] conducted in 2020, there are also relatively few studies on the agile mindset. Among these few papers, [1, 12, 43] aim to come with a list of the elements of agile mindset. Miler and Gaida [12] conducts a survey with 52 Agile practitioners who evaluate the importance of 26 selected elements of the agile mindset to the effectiveness of an Agile team. Miler uses the literature review to identify relevant elements, consisting mainly of books, web sites and hardly of peer-reviewed papers. By using a similar way to define the characteristics of the agile mindset, the study [1] conducts a review of the existing literature, including both scientific as well as practitioner publications, and interviews with practitioners. Study [43] identifies factors that affect the expansion of agile development in large organizations positively or negatively using interviews within multiple case studies then groups them in two categories: "agile mindset" and "contextual dependencies". When it comes to the difference between these types of studies and our study, our work focuses on the principles specifically that may contribute directly or indirectly to the understanding of the agile mindset, with their possible elements.

IV. Research Design

The well-known methods that had an influence on the manifesto include DSDM, TDD, ASD, D3, Scrum, Crystal, XP, Pragmatic Programming, FDD [6]. From this set of methods Scrum, XP, Crystal and FDD were used to form the search phrase, as these are better known than others do. Thus, the search was done with the keyword of "scrum 'Extreme Programming' crystal 'feature driven development'", in the dates between 28/01/2020 and 05/02/2020, without any specific filter in the year range, within the full text, in the libraries of IEEE Xplore, Web of Science, Science Direct and Springer, respectively. A total number of 368 works that are peer-reviewed and in English were returned from the search results. The researcher could not reach the full text of the 58 of them. The rest 310 works were examined through their full text to find and extract the methods mentioned. Considering that any new method name not encountered since after 83% of this search indicates that the search result set is sufficient in terms of the coverage.

After reaching the list of the methods, explicitly listed principles or principle-like attributes of each method along with their descriptions were extracted from the formal books (as indicated in Table 1) or from the formal web-site of the methods. Primarily, the principles were sought, if not found, the principle-like attributes were used. The principle-like attributes include philosophy, value, pillars, characteristics, and properties, respectively. To reach to this list of attributes (philosophy, value, pillars, characteristic, and properties), the concepts with a close relationship with dictionary meanings of "principle" were sought in multiple dictionaries. The relationships between these attributes are shown below. It demonstrates that the meaning of the principle, philosophy, value and pillars are related to each other by means of shared words in the descriptions of their meanings. By definition, properties and characteristics of a concept serve for effectively defining phenomenon under consideration [14], which make "characteristic" and "property" a proper candidate for inclusion.

- Principles: "a **basic** belief, theory, or rule that has a major influence on the way in which something is done" (macmillandictionary.com)
- Philosophies: "a system of **beliefs** that influences someone's decisions and behavior" (macmillandictionary.com)
- Values: "the **principles** and **beliefs** that influence the behavior and way of life of a particular group or community" (macmillandictionary.com)
- Pillars: an important idea, **principle**, or belief (macmillandictionary.com)
- Characteristics: "a typical or noticeable feature of someone or something" (dictionary.cambridge.org)
- Properties: "a quality or characteristic that something has" (oxfordlearnersdictionaries.com)

In the chain of the "reality-values-principles-practices", even though principles have a close relationship with practices in a way of influencing the pattern of practices done, the practices were not included in the set because of that they can/should be diverse and varying, with no limitation. Even though Agile methods share some common practices such as short time boxed iterations with adaptive and evolutionary refinements of plans, specific practices of the methods still vary [15], in this sense, it makes it difficult to collect all of them from all of the methods. Thus, this study internationally prefers to exclude practices from the list.

After reaching the list of principles of the methods, which is a straightforward process, the first author content-analyzed the principles' descriptions (L1) and grouped them into 32 categories (L2) based on his knowledge for a higher-level abstraction. These categories then were subsumed into two main categories (L3), by the same author. The whole grouping process was reviewed by one expert in Agile Software Development having both academic and sector background for 5 years in Agile Software Development particularly, and the list was adjusted accordingly (%18 of the items updated after two iterations). Then, over the consolidated list of the principles with their grouping (L2), analysis and evaluations were made by the authors. As a part of the evaluations, the first author conducted semi-structured interviews with two experts to evaluate the principle categories (L2), especially in terms of their contribution to agility. The notes taken were then reviewed by the interviewees and necessary corrections were made accordingly.

V. Findings and Analysis

A. Methods (RQ1)

The search mentioned in the Research Design Section to find out the Agile methods in the literature has ended with 28 methods listed in Table 1.

Regarding these methods, as one of them, **Evo**, the first Agile method in the list, provides a baseline for many Agile initiatives. As the most used method, **Scrum**, is designed for small self-organizing teams breaking their work into smaller parts that can be completed within time-boxed iterations that are no longer than one month. **DSDM** was initially proposed to build quality into RAD (Rapid Application Development). In the recent version, DSDM fixes time; functionality varies according to the need of stakeholders. It resembles Scrum in terms of practices such as time-boxing, iterative development, taking the customer in, staying mainly on the development layer, covering the world of a single team and increasing roles via the proxies. In a different way, DSDM adds a project layer on top with planning activities, encourages visualization through the concept of modeling and makes an emphasis on the quality aspects of the development.

DAD brings discipline in the implementation of Agile approaches and builds on the many practices from Scrum, AM, LSD, and others yet with the aim of moving beyond Scrum, which makes it a scaling framework as well. **DevOps** proposes a set of practices that combine software development (Dev) and operations (Ops) which aims to provide a continuous stream of integration and delivery. **FDD** focuses on the feature aspect of a project and development is organized based on the feature concept, posing a position located mainly on the first parts of the development pipeline. **XP** proposes software development engineering practices. **Crystal** family is a collection of the Agile methods proposing different sub-methods based on the individual project complexity and the team size that are measured mostly by the quantitative properties. Then, it recommends the implementation of certain roles and artifacts accordingly, representing a plan-driven approach to development to some extent.

An ancestor of considerable methods on the list, **RUP** draws attention with its object-oriented modeling, numerous roles and artifacts offering a descriptive and obsolete approach for today in terms of agility. **OpenUP** preserves the essential characteristics of RUP that include iterative development, use cases and scenarios driving development and architecture-centric approach yet adding some Agile aspects such as iterative development with feedback loops. Like RUP, **ICONIX** uses UML based diagrams turning to use case text into working code. **AM** was introduced to adapt modeling practices using an agile mindset and it covers only modeling. Sharing the same inventor, **ADM** focuses on the data aspects of development. Coined by the same inventor, **AUP** proposes a simplified version of RUP and, in 2012, was superseded by DAD.

ASD comes with some basic principles, lacking with implementation details, as a more iterative and shorter-interval version of the RAD. **ASP**, with an image of extinction with very few resources, describes concurrent development processes in the Japanese software industry, which already includes practices like dividing software into smaller parts, a time-fixed interval of delivery, close customer relations, and incremental construction of the system. Although **MSF** was not designed with a full Agile perspective at the first stage, it brought in an Agile template into the tool in 2005. **PSP&TSP** offers suggestions for individuals and teams to manage their own works and determines their competencies with a focus on measurement that they need to develop.

LSD focuses on optimizing the entire development process and reducing waste. **Kanban** focuses on continuous flow and continual delivery of work instead of iterating. **Scrumban** offers a structure that combines selected features of Kanban and Scrum.

OSSD is hardly to count as a pure Agile method, yet it can be considered similar to the Agile approach with sharing code freely, faster development cycles and such. **ISD** proposes development with small teams working in parallel and dependency management by using a combined spiral /waterfall model with daily builds aimed at developing a product with high speed.

TDD provides a set of practices for testing. **BDD** is an extension of test-driven development with a set of practices for testing. **PP** introduces a set of programming best practices in the form of the collection of short tips. These three (TDD, BDD and PP) are excluded from the list for further stages as they focus deeply on programming practices. **D3**, suffering from lack of sufficient resources, uses design as a part of processes to learn and better define requirements whereby design and user experience drive the development. For **APM**, there is similarly no sufficient resource for further investigation and thus, these two (D3 and APM) are excluded from the list for further stages.

While determining the "Obsolete" field in Table 1, three different parameters were looked at: 1- whether the main subject (such as UML modeling, object-oriented approach, spiral model) on which the method is based becomes obsolete in the Agile world for today, 2- whether superseded by another method, 3- no appearance in the VersionOne reports [13] from 2006 to 2019 (Obsolete) or disappearance towards the recent years (Nearly Obsolete), 4- the resources found during the authors' review on the methods belong to the far old years. These reasons for being obsolete as coded from 1 to 4 accordingly are also delivered in the list. For instance, AUP, ADM and AM are superseded by DAD and the ones using RUP as the foundation stone including OpenUp, ICONIX and AUP are out of date as RUP is so, at least for the Agile communities of today. For the rest of the methods that are referred to as "Alive", it implies that their ideas are still valid and their names are included in the reports of Version One for at least the recent three years (2017, 2018, and 2019).

TABLE I. LIST OF AGILE METHODS

Method	Abb.	Release year	Vitality	Reasons of Being Obsolete	Principle Related Attribute	Main Reference
Evolutionary Project Management	Evo	1981	Obsolete	3,4	Principles	[17]
Dynamic Systems Development Method	DSDM	1995	Nearly Obsolete	3	Principles	[18]
Scrum	Scrum	1995	Alive	-	Pillars	[19]
Rational Unified Process	RUP	1996	Obsolete	1,3,4	-	[20]
Agile Software Process	ASP	1997	Obsolete	3,4	Characteristics	[21]
Open Source Software Development	OSSD	1997	Obsolete	3,4	-	[22]
Crystal	Crystal	1998	Obsolete	3,4	Properties	[23]
Adaptive Software Development	ASD	1999	Obsolete	3,4	Characteristics	[24]
Extreme Programming	XP	1999	Alive	-	Values	[25]
Feature-driven Development	FDD	1999	Nearly Obsolete	3	-	[26]
Internet Speed Development	ISD	1999	Obsolete	1,3,4	-	[27]
Pragmatic Programming	PP	1999	-	-	-	-
Agile Modeling	AM	2002	Nearly Obsolete	1,2,3	Values	[28]
Agile Data Method	ADM	2003	Obsolete	1,2,3	Philosophies	[29]
Lean Software Development	LSD	2003	Alive	-	Principles	[30]
Agile Unified Process	AUP	2005	Nearly Obsolete	1,2,3	Philosophies	[31]
Microsoft Solutions Framework	MSF	2005	Obsolete	3,4	Principles	[32]
Open Unified Process	OpenUP	2006	Obsolete	1,3,4	Principles	[33]
Behavior-Driven Development	BDD	2009	-	-	-	-
DevOps	DevOps	2009	Alive	-	Principles	[34]
Scrumban	Scrumban	2009	Alive	-	-	[35]
Kanban	Kanban	2010	Alive	-	Principles	[36]
Disciplined Agile Delivery	DAD	2012	Alive	-	Principles	[37]
Design Driven Development	D3	-	-	-	-	-
Personal Software Process & Team Software Process	PSP&TSP	1996	Obsolete	3,4	Principles	[38]

Agile Portfolio Management	APM	-	-	-	-	-
ICONIX	ICONIX		Obsolete	1,3,4	-	[39]
Test-driven development	TDD	-	-	-	-	-

B. Agile Principles (RQ2)

After reaching the list of the methods, the principles of each method (L1) were collected as described in the Research Design section and 105 (101 distinct in names) principles were achieved. These principles were then grouped into the high-level principles that are 33 in number (L2). During this stage, it is seen that some original principles (L1) can serve for multiple high-level principles (L2) then they are duplicated under different high-level L2 principles, yielding 114 L1 principles in total (duplicated ones are marked with a number inside the corresponding principle box). The L2 principles are also classified as People or Process-Relevant (L3), according to their descriptions. At this stage, if an L3 item includes both Process and People Relevant L2 item(s) then it was taken of those with a higher number (there was no equality encountered). As a note, this hybrid distribution was seen in the 5 of 33 L2 principles. All trees are depicted as below bearing principle name, relevant method(s), and the unique numbers if duplicated.

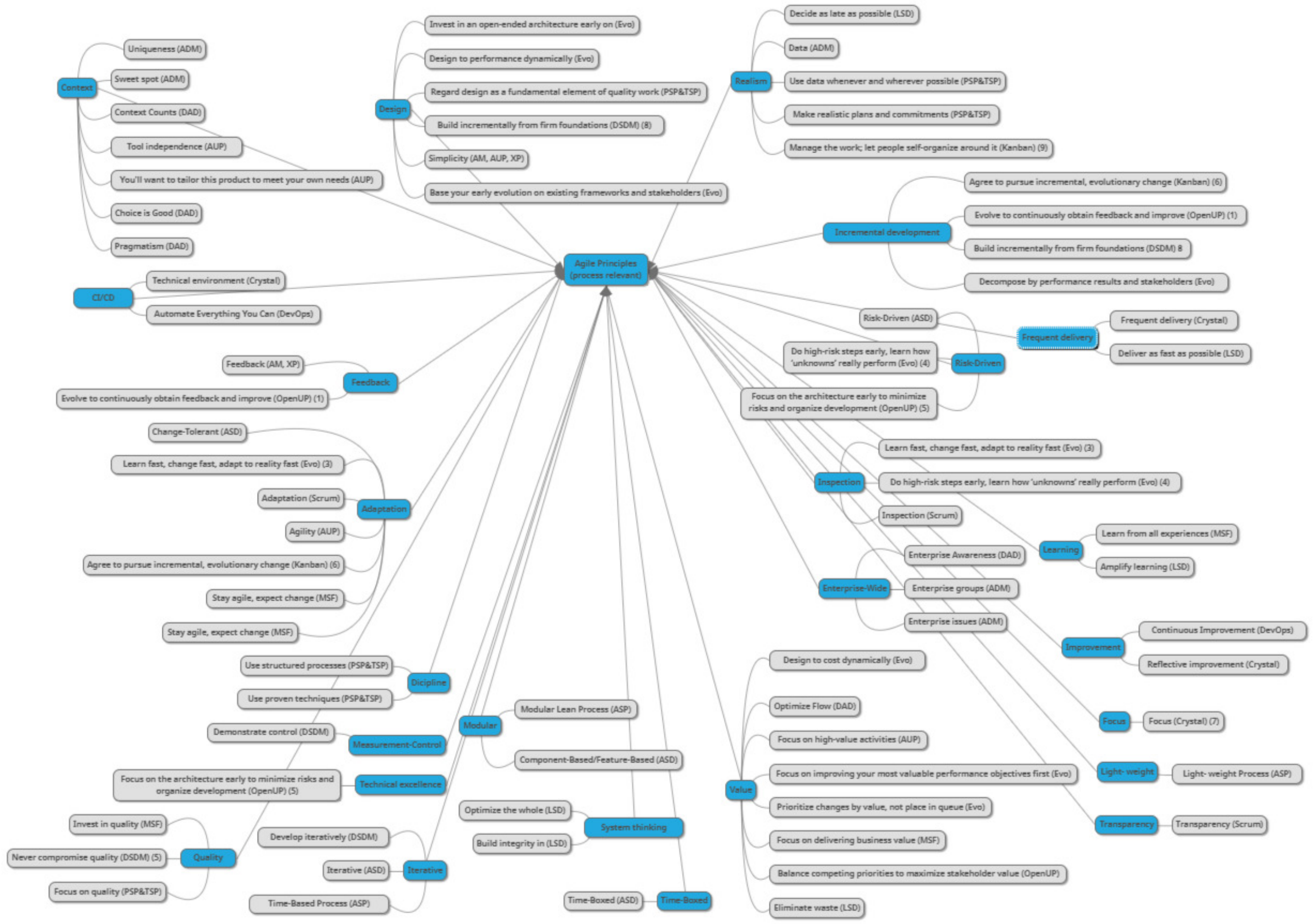
Agile Principles (process relevant)
Context
Uniqueness (ADM)
Sweet spot (ADM)
Context Counts (DAD)
Tool independence (AUP)
You'll want to tailor this product to meet your own needs (AUP)
Choice is Good (DAD)
Pragmatism (DAD)
Design
Invest in an open-ended architecture early on (Evo)
Design to performance dynamically (Evo)
Regard design as a fundamental element of quality work (PSP&TSP)
Build incrementally from firm foundations (DSDM) (8)
Simplicity (AM, AUP, XP)
Base your early evolution on existing frameworks and stakeholders (Evo)
Realism
Decide as late as possible (LSD)
Data (ADM)
Use data whenever and wherever possible (PSP&TSP)
Make realistic plans and commitments (PSP&TSP)
Manage the work; let people self-organize around it (Kanban) (9)
Incremental development
Agree to pursue incremental, evolutionary change (Kanban) (6)
Evolve to continuously obtain feedback and improve (OpenUP) (1)
Build incrementally from firm foundations (DSDM) 8
Decompose by performance results and stakeholders (Evo)
CI/CD
Technical environment (Crystal)
Automate Everything You Can (DevOps)
Feedback
Feedback (AM, XP)
Evolve to continuously obtain feedback and improve (OpenUP) (1)
Risk-Driven
Risk-Driven (ASD)
Do high-risk steps early, learn how 'unknowns' really perform (Evo) (4)
Focus on the architecture early to minimize risks and organize development (OpenUP) (5)
Frequent delivery
Frequent delivery (Crystal)
Deliver as fast as possible (LSD)
Adaptation
Change-Tolerant (ASD)
Learn fast, change fast, adapt to reality fast (Evo) (3)
Adaptation (Scrum)
Agility (AUP)
Agree to pursue incremental, evolutionary change (Kanban) (6)
Stay agile, expect change (MSF)
Stay agile, expect change (MSF)
Inspection
Learn fast, change fast, adapt to reality fast (Evo) (3)
Do high-risk steps early, learn how 'unknowns' really perform (Evo) (4)
Inspection (Scrum)
Learning
Learn from all experiences (MSF)
Amplify learning (LSD)
Enterprise-Wide
Enterprise Awareness (DAD)
Enterprise groups (ADM)
Enterprise issues (ADM)
Improvement
Continuous Improvement (DevOps)
Reflective improvement (Crystal)
Dicipline
Use structured processes (PSP&TSP)
Use proven techniques (PSP&TSP)
Measurement-Control
Demonstrate control (DSDM)
Technical excellence
Focus on the architecture early to minimize risks and organize development (OpenUP) (5)
Modular
Modular Lean Process (ASP)
Component-Based/Feature-Based (ASD)
Quality
Invest in quality (MSF)
Never compromise quality (DSDM) (5)
Focus on quality (PSP&TSP)
Iterative
Develop iteratively (DSDM)
Iterative (ASD)
Time-Based Process (ASP)
System thinking
Optimize the whole (LSD)
Build integrity in (LSD)
Time-Boxed
Time-Boxed (ASD)
Value
Design to cost dynamically (Evo)
Optimize Flow (DAD)
Focus on high-value activities (AUP)
Focus on improving your most valuable performance objectives first (Evo)
Prioritize changes by value, not place in queue (Evo)
Focus on delivering business value (MSF)
Balance competing priorities to maximize stakeholder value (OpenUP)
Eliminate waste (LSD)
Focus
Focus (Crystal) (7)
Light- weight
Light- weight Process (ASP)
Transparency
Transparency (Scrum)

Fig. 1. Process-relevant Principles

With **iterative** development along with **frequent delivery**, a big bunch of development is divided into smaller functional increments to understand functionality better, to manage **risk** effectively and to get **feedback** from customers and end users early. Iterative development encourages experimentation and **learning**. Through feedback and learning cycles, teams can identify areas for **improvements**. To the short cycles of iterations, a **fixed schedule** accompanies [in some methods] to reach a high level of predictability. With iterative development, the accumulation is not by default in additive kind. The system developed can yield **incremental** progress thus an organic growth of the system is achieved as required to **adapt** to changes.

In order to manage the complex world of **reality** along with its **context** variations, the human-beings who have equally complex abilities is brought up against it. Human-made proxy products, such as processes, documents, fixed plans, are neither capable of representing the actual ability of the human nor the reality itself. Instead of these intermediate solutions, by reducing their significance, people are in the foreground to counterbalance the reality. And, therefore it is aimed to be close to the customer who is relatively close to reality. As being close to the front sides, customer and end user are the real owner and user of requests, which reminds being **focusing on the customer, value, quality and goal**.

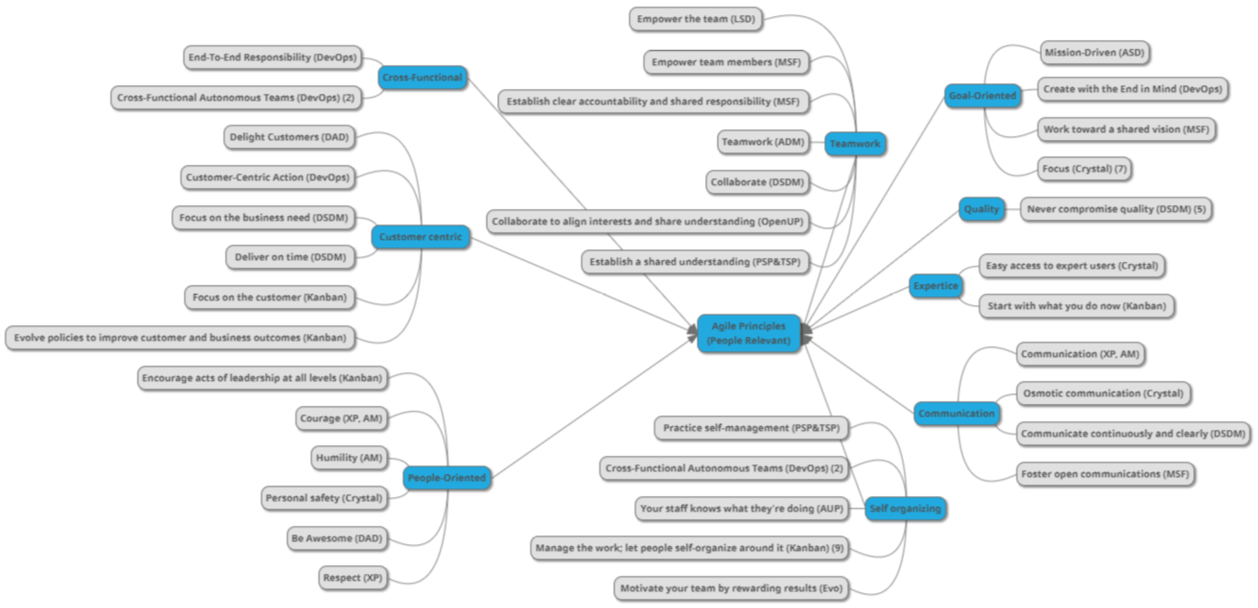
Agile Principles (People Relevant)
Cross-Functional
End-To-End Responsibility (DevOps)
Cross-Functional Autonomous Teams (DevOps) (2)
Customer centric
Delight Customers (DAD)
Customer-Centric Action (DevOps)
Focus on the business need (DSDM)
Deliver on time (DSDM)
Focus on the customer (Kanban)
Evolve policies to improve customer and business outcomes (Kanban)
People-Oriented
Encourage acts of leadership at all levels (Kanban)
Courage (XP, AM)
Humility (AM)
Personal safety (Crystal)
Be Awesome (DAD)
Respect (XP)
Teamwork
Empower the team (LSD)
Empower team members (MSF)
Establish clear accountability and shared responsibility (MSF)
Teamwork (ADM)
Collaborate (DSDM)
Collaborate to align interests and share understanding (OpenUP)
Establish a shared understanding (PSP&TSP)
Self organizing
Practice self-management (PSP&TSP)
Cross-Functional Autonomous Teams (DevOps) (2)
Your staff knows what they're doing (AUP)
Manage the work; let people self-organize around it (Kanban) (9)
Motivate your team by rewarding results (Evo)
Goal-Oriented
Mission-Driven (ASD)
Create with the End in Mind (DevOps)
Work toward a shared vision (MSF)
Focus (Crystal) (7)
Quality
Never compromise quality (DSDM) (5)
Expertice
Easy access to expert users (Crystal)
Start with what you do now (Kanban)
Communication
Communication (XP, AM)
Osmotic communication (Crystal)
Communicate continuously and clearly (DSDM)
Foster open communications (MSF)

Fig. 2. People-relevant Principles

In the association with the reality, people often use investigation, **inspection, learning and feedback loops** to get to know more about the reality. People abandon the passive position of classical methods and take on a more active role. Effective learning includes learning from mistakes. At this process, one of the things people need is **courage** needed for change including changing one's own self, with a feeling of being safe and having relatively high tolerance against mistakes that require **personal safety**. This calls for that both the team and the members of the team **respect** each other. Respect strengthens communication channels, supports coloration and accepting feedback. Respect assures for individuals a suitable safe place for trial and learning. Courage is also important to hearten people to make critical decisions to be able to change direction for adaptation.

"To move quickly", the information should flow quickly inside and between the teams. This is mainly why Agile teams are preferably co-located and cross-functional. Thus, with close and intense **communication**, the interaction of information increases and the information itself becomes agile: it is updated, corrected, accelerated, shared to gain experience and to develop new ideas throughout and beyond **enterprises**. **Transparency** plays as a facilitator for communication. Communication enables learning, including from the developed solution itself. It is necessary to communicate with the developed solution itself to see its behavior, listen to what it says (the process is successful, throws an error, etc.). Moreover, along with shared goals, communication also supports collaboration and **teamwork**.

Cross-functionality reduces the cost of communication by gathering the necessary competencies into the team and enables rapid action. With the contribution of cross-functionality, **self-organization** enables teams to operate around varying cases of the complex world of reality.

Agile processes are additionally equipped with **technical excellence, continuous integration and deployment, system thinking, design** capabilities, **disciplined approaches and measurement-control mechanisms**, by some of the methods.

Numerically speaking, process- relevant items (of L1) cover 64% (73/114) of the whole. Among the people- relevant items (of L2), depending on the definition of agility, adaptation to realism to create value comes into prominence. However, the enterprise-wide perspective is relatively underestimated to create this value (of the organization). This may be because of the people who created these methods having more developer backgrounds. Design may come to the fore with an effect of a similar situation. Quality emphasis has a moderate place unlike in the manifesto that gives no place for it. Although discipline in Agile approaches is hardly addressed, we see that some methods include this dimension. Time-box, frequent delivery, and iterative development practices applied by many methods are rather less apparent at the principles level. However, when considered incremental and iterative development, frequent delivery and continuous integration and deployment (CI/CD) together, they take considerable place. It is observed that the Lean approach, of which the main focus is not agility, but the literature counts it as an Agile method, creates a unique field and does not receive much support from other methods for System Thinking.

In the people-relevant dimension, we see that human and team relevant principles come to the fore. It is natural in this sense that the channels of communication equipping human abilities are seen at a high level of principles. The context dimension, which needs human abilities to manage rather than the ability of the process dimensions, takes an important place. Parallel to reality-driven, customer orientation is also located at higher levels at the people-relevant side. However, the expertise of individuals who are expected to pose a parallel level with the context dimension and being crucial for self-organizing teams takes a lower place. Similarly, cross functionality, which is proposed by many methods, is relatively at a low level. We see that this principle is supported by DevOps, which is bounded by this principle very profoundly.

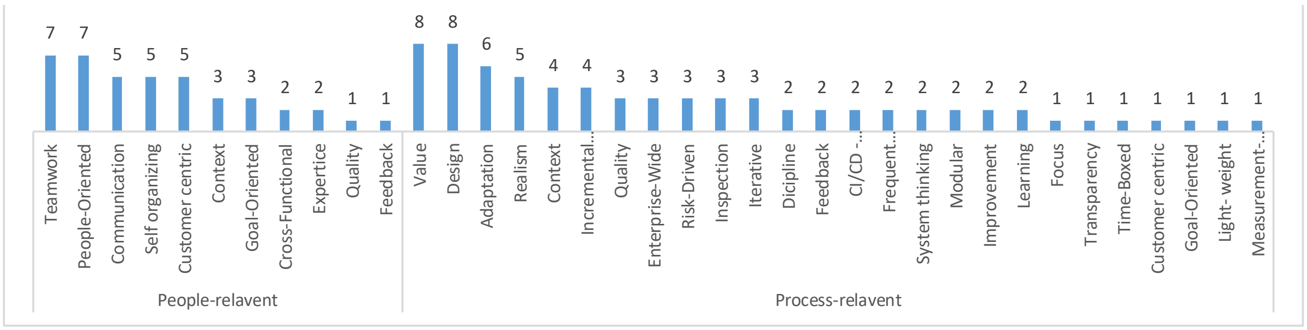

Fig. 3. Count of Principles

Unsurprisingly, we can say that those methods with a process expression in their names such as Agile Software Process (ASP) or in their definitions such as of Scrum outweigh the process side. As the first instance of Agile methods, Evo approaches agility mainly from the process side. Lean Software Development poses a very process-oriented image with its focus on the waste in the processes.

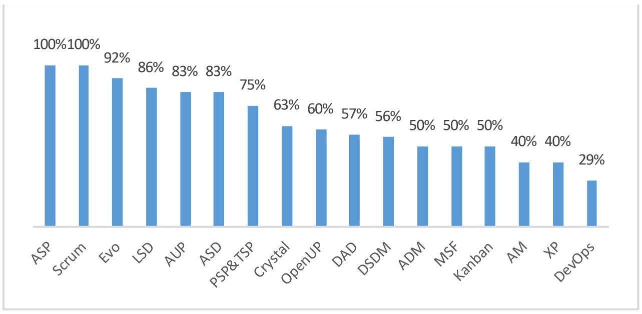

Fig. 4. % of Process-Relevance of Principles

Although DevOps has many aspects that touch processes, it is remarkable that DevOps is at the forefront of people's dimension. Agile Modeling bases on XP in defining its values. In the context of this study, since XP and AM are included with values instead of principles and as values are more people-oriented, it can be considered normal that these two are seen at the forefront of people-relevant dimension.

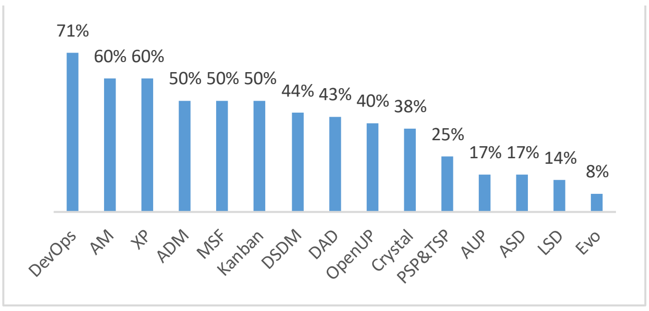

Fig. 5. % of People-Relevance of Principles

C. *Comparison of Principles with the Manifesto (RQ3)*

When looking at the degree of overlap of the principles (L2) with those of the manifesto, it is seen that more than half of the determined principle categories are touched by the manifesto. Cohen argues that all Agile methods follow the four values and twelve principles of the Agile Manifesto [16], yet they provide more principles than the manifesto in terms of the coverage.

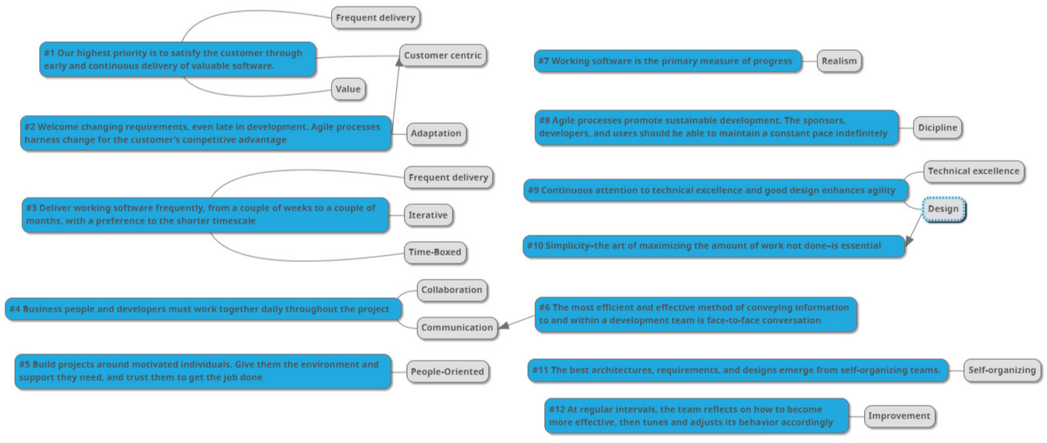
#1 Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
Frequent delivery
Customer centric
Value
#2 Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
Adaptation
#3 Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
Frequent delivery
Iterative
Time-Boxed
#4 Business people and developers must work together daily throughout the project
Collaboration
Communication
#5 Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
People-Oriented
#7 Working software is the primary measure of progress
Realism
#8 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
Dicipline
#9 Continuous attention to technical excellence and good design enhances agility
Technical excellence
Design
#10 Simplicity–the art of maximizing the amount of work not done–is essential
#6 The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
#11 The best architectures, requirements, and designs emerge from self-organizing teams.
Self-organizing
#12 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly
Improvement

Fig. 6. Map for the Manifesto Prıncıples

However, even if the feedback is not explicitly stated, it is assumed to receive feedback on the delivery of the product with the early delivery, providing inspection accordingly. Similar logic can be put forward for incremental development in relation to iterated progress. Cross-functionality can be seen as a prerequisite for self-organizing teams. Similarly, although transparency is not explicitly stated, it can be considered as a capability gained automatically by establishing intensive (especially on a daily basis) communication. Principles relating to goal-oriented, focus and modular may not be seen as primary to be included in the manifesto, within a dedicated mentioning.

D. *Evaluation of the Principle by Experts (RQ4)*

The first author conducted semi-structured interviews with two experts to evaluate the principle categories (L2), especially in terms of their contribution to agility. Expert A has 15 years of experience in total, of which 5.5 years as a product owner in a bank in Turkey applying Scrum. Expert B has 13 years of experience in total mainly from two different banks in Turkey, of which 4 years in a Scrum development team. The following statements directly convey the views of experts on the principles.

Expert A states that each of these principles determined supports the agility. Using these principles together in the whole picture will be beneficial for maintaining balanced, healthy and sustainable agility. According to her, although adaptation is important, the market has a lot of emphasis on it, which can lead to an unbalance in some other points. For example, in some cases of adaptation without a balance, quality, enterprise-wide, risk-driven, systematic, realistic (adaptation to realistic changes) approaches and sufficient inspection phases may be damaged. This approach may lead to the emergence of unsustainable structures that will not benefit the customers in the long run. Teams that move away from the holistic picture with the effect of adaptation pressure can result in isolations across the teams themselves, such as happening in impact analysis mostly conducted in non-sufficient and isolated ways. In addition to agility, the necessity of elements such as expertise and discipline to support it manifests itself. Expertise for instance is important enough, as becoming a prerequisite for self-organizing teams to be able to self-organize. Unstable teams and teams with a low level of expertise unlikely to become self-organized. In addition to adaptation pressure, time-boxing may lead to compromise on quality and value with a similar effect.

She states although value and customer orientation are important, a blindfolded dedication to the customers may cause human values of teams to be ignored and remained in the background. With a customer-driven approach, development teams come to a more passive position, and customer demands that do not go through enough filters of the customers put more pressure on the teams. Considering these situations, principles such as system thinking, organization-wide, quality and realism stand out for sustainable agility. In addition, incremental and iterative development, teamwork, cross-functionality come into prominence in a way supporting agility fundamentally.

Expert B asserts that transparency contributes to reality by supporting open and clear environments, in a way of reducing reworks. She adds that frequent delivery increases quality. Frequent delivery, on the hand, cannot be possible in some cases depending on the nature of the project. Progressing iteratively reduces the risk for the users and developers as the users see the increment at the early stages and give feedbacks. For developers working the design up-front as much as possible reduces the risk as well.

According to her, teams with a deadline coming with the iteration time-box can have positive and negative effects depending on the situation. In both cases, determining the end of the iteration by the teams supports self-organization. It supports meetings to be more productive. However, for self-organized individuals, time-boxing will be meaningless. Daily meetings and time- boxing will be effective in a positive way with pressure for non-self-motivated individuals. However, this pressure can also have a negative effect on some people.

She says it is usually expressed that organizations trust the Scrum teams, yet it is a utopia to trust the team in an absolute manner. Factors and rules outside the teams do not allow the teams to be truly self-organizing. Self-organization can also be a problem, especially in the setup stages of Scrum. Scrum does not respect the context dependencies much. Depending on the context, it may be difficult to set up Scrum with its factory settings, especially during the transformation stages or in disciplined environments like in a bank.

She adds that Agile [Scrum] comes with a customer-oriented process setup. Customer feedback directs the development. What the customers want is accepted as master and generally does not go through a filter. Project-based team structures eliminate the need to work on a modular basis. Cross-functionality is thus provided for the project via such temporary teams. It is actually a structure that supports context diversity and process flexibility.

VI. Discussion

Among the **methods**, some of them focus on project management (like Scrum and DSDM), while some others focus mainly on software development activities (like XP, Crystal), mostly on the team level, ignoring organization-wide perspective. The main reason for being mostly on the team level may be that the creators of the methods mostly come from the software development background. While those such as DevOps and Kanban provide a continuous stream for delivery (continuous planning, integration, delivery, feedback etc.), some others like DSDM, Scrum uses segmented units of the timeline to manage the pipeline. Thinking the time within the segmented iterations like a sprint in Scrum can be an advancement for a big bunch of development lines of plan-driven approaches of yesterday, yet it cannot be regarded as a contemporary method of today. Contrary to the agile logic, handling these static time frames of iterations with strict planning and expecting a concrete product at the end is very instance of a plan-driven approach. Instead, to keep with fluctuations of the complexity of the reality that is at very atomic level of granularity, a continuous approach to development providing a very mutual and natural atomic level of reflections may be needed. This is probably why DevOps, Scrumban and Kanban stay alive among those a few, by providing a continuous stream for the pipeline.

Staying alive among those a few, XP and DevOps take place as focusing on people-related issues in terms of principles. However, being human oriented and being-system oriented seem to be a binary choice within the methods. While many models establish their main structure on the roles of people, there are some methods such as Kanban, LSD that focus the system rather than people.

As the most used method, an interesting issue with Scrum that takes a process-oriented approach to development, assertively delegates this duty of process-orientation to its a few basic roles of people. And as it is expected, this intense process orientation is prone to be derailed by people who are naturally far from providing a standard approach to what these intense processes require.

The aim of LSD is to approach the zero (waste) point. Agility leans more on the expansion of perspectives; learning (fail fast), reworks (creating features only to understand customers better at the earliest) and so on. This "haste” to respond quickly in Agile may “make waste", implying that Lean and Agile approaches can be contrast serving in two different directions. However, there is a Lean perspective in the manifesto by advocating just enough documentation, reducing “ineffective communication” occurring in the hierarchy, tools and processes. This shows us that the Lean and Agile approaches are used together in the manifesto, maybe with confusion, even if they contain some contradictions.

When it comes to the **manifesto**, interestingly, we see no quality-related emphasis on it. Another interesting point in the manifesto is that the agility of Agile Software Development is considered a separate and isolated body, not directly connected with the organization wide perspectives. The main reason for this may be that the manifesto writers come from the software development background, too. In the context of software development, it will not be enough to include the customer in the processes. Therefore, considering the Agile Software Development separate from the whole organization come with some issues. Another important issue in the manifesto appears in contextualization. It is usual for the reality to vary depending on the context, which calls for each unique practitioner to define a space for their context and to shape their own agility within this space. However, the manifesto does not explicitly refer to the context dependencies, which is an important dimension of realism, nor is there any concern about the expertise of people, which is a crucial factor to deal with the context, in the vertical dimension. Although in Agile approaches, T-shaped specialization is recommended instead of general specialization with assuming that it contributes to collaboration within cross-functional teams, yet it only provides a horizontal dimension to the context-related issues, which should remind us not to ignore the issues related to the depth of the context, especially of the complex world.

Some of the determined **principles** (L2) are close to each other (such as teamwork and self-organizing or incremental and iterative development, frequent delivery and CI/CD), some are closely supporting each other (such as feedback, communication and transparency). Others are not open to debate, as they make an absolute positive contribution (such as improvement and learning). We will discuss here debatable ones, in general, without mentioning the differences between those close to each other.

As one of the principles, moving within iteration is an old school tradition, seen mostly in the first generation of the methods. Maintaining this tradition with building walls (with fixed times) and trying to live agility within limitations of these walls of iterations are a kind of reduction to and conflict for people who have more atomic level, more sensitive, stronger agility capabilities in themselves. As an excuse, fixing iterations with a "deadline" to speed up the development to assure the fulfillment of the customer's top present needs, or using fixed iterations for motivating development teams (as stated by Expert B) are just expected benefits. Using a combination of adaptation and iteration with time-boxes may create artificial pressures on teams causing compromise on some other values (Expert B). It implies that this artificial 'solution' produced for indirect problems (not being motivated, not being value oriented) creates a cause for another problem; trying to imprison the reality of the future by artificial parameters of time. However, the reality of future is so dominant and free that it does not fit in an artificial frame of time (like sprints with a fixed end), then it gets out of iteration limits, enforces obedience of all other parameters. For example, towards the end of the fixed iteration which does not progress according to the plan enforces a situation where the scope or quality will be compromised. It is reminded that time is one of the strongest among parameters, then people should learn to get along with it instead of imprisoning it.

The term artificial means iteration is not in a pure form of time itself rather a kind of proxy of it, a sort of representative of the time at a different platform. In this sense, it takes the process away from reality. Moreover, iteration-based planning means adding determinism into the complexity of the future, especially if it comes with a fixed end time. This approach indicates an attempt to manage non-deterministic software development with deterministic methods. Using iterations as a batch feedback method with some static rituals is to communicate with an artificial cycles as well. For instance, an issue at the beginning of the sprint may, not necessarily but most probably, delay to the review or retrospective meetings that are located at the end of the sprints. Fixed rituals break the natural flow of the reality (such as in getting feedback when it is ready). So, it is recommended to synchronize the loop of feedbacks with its own cycle of the realism instead of an artificial one. Thus, with iterations saved from fixed events, the sooner the solution is delivered, the sooner feedback can be received.

Realism is to be driven by the reality itself instead of the proxy of it. For example, processes to organize real operations aiming to be a projection of the reality, with trying to represent it or even direct it by going ahead of it are also a sort of artificial proxies. However, a process is not the reality itself. It is a kind of artificial entities produced by humans. After all, models are human-made products, and every human-made product (software, hardware, ideas etc.) is defective. Like in the time parameter, the reality as the master dominates the static [process] frameworks, models and methodologies that try to be real.

Self- organization increases the ability to respond to change while decreasing the speed of response for decision-making in quickly, easily and adequately manner (as stated by Expert A). Advantageously, it strengthens the concept of "move" in the definition of agility by means of delegating the work to those who know it closely and expanding decision capabilities, yet it should not be regarded as a way that contributes to agility in absolute terms. Cross-functionality reduces the cost of communication by gathering the necessary competencies into the team and enables rapid actions. However, self-sufficient (!) teams weaken their abilities in the holistic picture with their possible estrangement. Even though the ability, speed and convenience of moving increase inside the teams, these capabilities may be in danger in the context of multiples teams (as stated by Expert A).

Agility is easier when managed in the abstract dimension, which calls for more design up-front. Managing the solution with a "concrete running software" may be costly, hand-binding, and waste. If the customer's need is "discoverable" a bit from the front, the up-front investigation should be located. Agility is also more sustainable when combined with system thinking, quality and organization-wide perspectives (Expert A) and discipline (Expert A, B).

Software developers develop software mostly for people, with people. However, the human is not a pure representative of the deepest level of the reality that is in a perpetual state of change. As a proxy, they cannot perceive and convey the reality as it is, sometimes deliberately and they add their natural interpretations, perspectives, and limitations of their context to the reality, making them a very strong decrement point in transmitting it. Hence, driving the change solely by people may be misleading (as partially stated by Expert A).

In parallel, Parnas and Clements [40] states (as paraphrased by [2]) that a system's users seldom know exactly what they want and cannot articulate all they know. Even if they could state all requirements, there are many details that we can only discover once we are well into implementation. Brown's study [41] reports three different perspectives about the same project varying dramatically with the role of people. The customer will of course be a mediator of the change. The important thing here is to be the seeker of the reality together with the customer and not regarding customers sacrosanct and accepting them as the absolute point of the reality. There is less visible yet another crucial layer between the customer and the reality to discover with them together.

VII. Conclusion and Future Work

The study does not attempt to redefine agility in the software solution development in a full-fledged way. It rather makes an evaluation based on the principles, considering a particular approach to the definition of the agility, with some threats to validity when considered low level validation by experts. Even so, the study may provide specific contributions, especially with its progressive position that has two faces: locating the principles on the center, looking at the relationship between the principles and the methods and examining how these principles support agility. In this sense, as future work, it can be investigated to what extent a specific method supports agility through these principles. However, as the next study, we prefer to improve these principles by combining results from other related studies and examine how and to what extent each element in the final set supports agility.

References

[1] A. Mordi, and M. Schoop, "Making It Tangible–Creating A Definition Of Agile Mindset", ECIS, 2020.

[2] C. Larman and V. R. Basili, "Iterative and incremental developments: a brief history", Computer, vol. 36, pp.47–56, 2003.

[3] E. Trist, "The evolution of socio-technical systems", Occasional paper, vol. 2, 1981.

[4] H. Takeuchi, and I. Nonaka, "The new new product development game", Hardvard Business Review, vol. 64, no.1 1986.

[5] G. Morgan, Images of organization, Sage Publications: Beverly Hills, 1986.

[6] P. Hohl, J. Klünder, A. van Bennekum, R. Lockard, J. Gifford, J. Münch, and K. Schneider, "Back to the future: origins and directions of the "Agile Manifesto"–views of the originators," Journal of Software Engineering Research and Development, vol. 6, no.1, 2018.

[7] N. G. Abbas, A. M. Gravell and G. B. Wills, "Historical roots of agile methods: Where did "Agile thinking" come from?, International conference on agile processes and extreme programming in software engineering, pp.94-103, 2008.

[8] A. Cockburn and J. Highsmith, "Agile Software Development: The Business of Innovation", Computer vol. 34, no.9, pp.120–127, 2001.

[9] P. Kruchten, "Contextualizing agile software development", Journal of Software: Evolution and Process, vol. 25, no. 4, pp. 351-36, 2013.

[10] K. Conboy, and B. Fitzgerald, "Toward a conceptual framework of agile methods: a study of agility in different disciplines" ACM workshop on Interdisciplinary software engineering research, pp.37-44, 2004.

[11] J. Highsmith, Agile Project Management, Boston: Addison-Wesley. . 2004.

[12] J. Miler and P. Gaida, "On the agile mindset of an effective team–an industrial opinion survey", Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 841-849, 2019.

[13] https://stateofagile.com/

[14] R. Suddaby, "Editor's comments: Construct clarity in theories of management and organization", 2010.

[15] C. Larman, "Agile and Iterative Development: A Manager's Guide", C. Alistair, H. Jim, (eds.), Pearson Education: London, 2004.

[16] D. Cohen, M. Lindvall and P. Costa, "An Introduction to Agile Methods", Advances in Computers, pp.1–66, 2004.

[17] T. Gilb, "Evolutionary Development", SIGSOFT Softw. Eng. Notes, vol. 6. No.2, 1981.

[18] J. Stapleton, DSDM, dynamic systems development method: the method in practice, Harlow: England, 1997.

[19] K: Schwaber and J. Sutherland, "The scrum guide", Scrum Alliance, 2011.

[20] https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[21] M. Aoyama, "Agile Software Process model," 21st International Computer Software and Applications Conference, 1997.

[22] E.S. Raymond, The Cathedral and the Bazaar, O'Reilly: Cambridge, 1999.

[23] A. Cockburn, Surviving object-oriented projects: a manager's guide, Addison-Wesley: Longman Publishing, 1998.

[24] J. A. Highsmith, Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, New York: Dorset House, 2000.

[25] K. Beck, Extreme programming explained: embrace change, Addison-Wesley Professional, 2000.

[26] P. Coad, J. D. Luca and E. Lefebvre, Java modeling color with UML: Enterprise components and process with Cdrom, Prentice Hall PTR, 1999.

[27] M.A. Cusumano and D. B. Yoffie, "Software development on Internet time." IEEE Computer, vol. 32, no.10, pp.60-69, 1999.

[28] S. Ambler, Agile modeling: effective practices for extreme programming and the unified process, John Wiley & Sons, 2002.

[29] S. Ambler, Agile database techniques: Effective strategies for the agile software developer, John Wiley & Sons, 2003.

[30] M. Poppendieck, T. Poppendieck, Lean Software Development: An Agile Toolkit, Addison-Wesley Professional, 2003.

[31] http://www.ambysoft.com/unifiedprocess/agileUP.html

[32] M. Turner, Microsoft solutions framework essentials: building successful technology solutions, Microsoft Press, 2006.

[33] P. Kroll, B. MacIsaac, Agility and Discipline Made Easy: Practices from OpenUP and RUP, Pearson Education, 2006.

[34] G. Kim, J. Humble, P. Debois, and J. Willis, "The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations", IT Revolution, 2016.

[35] C. Ladas, "Scrumban-essays on kanban systems for lean software development", Lulu.Com, 2009.

[36] D. J. Anderson, Kanban: successful evolutionary change for your technology business, Blue Hole Press, 2010.

[37] S. W. Ambler, and M Lines, Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise, IBM press, 2012.

[38] Watts, Using a defined and measured Personal Software Process, https://www.amazon.com/Introduction-Software-Process-Watts-Humphrey/dp/020147719X.

[39] D. Rosenberg, M. Stephens and M. Collins-Cope, Agile development with ICONIX process, New York: Editorial Apress, 2005.

[40] D. L. Parnas and P.C. Clements, "A rational design process: How and why to fake it", IEEE transactions on software engineering, vol.2, pp. 251-257, 1986.

[41] A. D. Brown, "Narrative, politics and legitimacy in an IT implementation, Journal of Management Studies, vol. 35, pp.35-58, 1998.

[42] N. Ozkan, "Imperfections Underlying the Manifesto for Agile Software Development", 1st International Informatics and Software Engineering Conference (UBMYK), 2019.

[43] H. van Manen, H. van Vliet, "Organization-Wide Agile Expansion Requires an Organization-Wide Agile Mindset", Product-Focused Software Process Improvement. Ed. by A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, M. Raatikainen, pp. 48–62, 2014.