Is Agile Antifragile?

Article · July 2019

CITATIONS
0

READS
32

1 author:

Latchezar Tomov
New Bulgarian University
86 PUBLICATIONS **15** CITATIONS

SEE PROFILE

Is Agile Antifragile?

Latchezar Tomov
dept. of Informtics
New Bulgarian University
Sofia, Bulgaria
lptomov@nbu.bg

***Abstract*— the main issues of software project management are complexity and fragility –large projects are resistant to policy optimization and good practices. Agile is a step towards solving this problem and has been proclaimed as the best solution, as "antifragile" project management. We investigate its properties, propose necessary conditions for antifragile management (related to convexity of probability density distributions and utility functions), and compare them.**

Keywords—Project management, Agile, Antifragile, Convexity, Utility functions, Probability, Statistical Process Control

I. INTRODUCTION

In this paper, we focus on software project management and more specifically the Agile framework. We analyze how it relates to the concept and definition for antifragile systems and anifragility in general. Our purpose is to improve Agile in order to move more into the proactive, antifragile domain and escape from the label of being entirely "damage control" oriented.

Software projects are high-risk investment [11]. In [18] we examined the current state of project management:

- Nearly 17% of projects have average cost overrun 200% and a schedule overrun of 70%
- Only 41% meet all three major goals – budget, schedule and quality
- When PMO project management methodology is used, 57% percent meet budget, 56-77% meet original goals and 51% stay on schedule Overall only 38% of projects are successful, with 76% for small projects and only 10% of large projects
- 75% of managers expect their software development projects to fail
- Financial losses from lacking project performance are 122millions per every 1bln invested.

How was that changd since then? The latest data (Fig.1) shows stable processes excluding the scope creep. In a previous article, we discussed the importance of process stability in framework of Deming's system of profound knowledge [6-7]. The ecosystem of methods, projects and teams in the software industry, monitored by PMI has reached a stable state with the exception of scope creep [11]. It would not improve more without reforms, based on knowledge, on theory. Agile/Scrum is part of that ecosystem and itself has reached stable state. It needs a new look, a new paradigm and quite possibly – a reformed and more rigorous theoretical foundation.

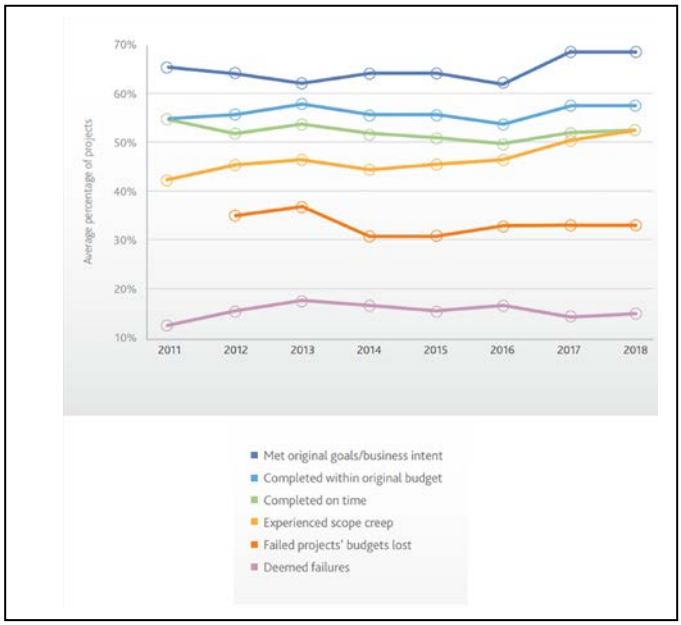

Fig. 1. Software project success by years [11]. Copyright 2018 by Project Management Institute

Agile has been created as a risk management strategy for complex systems and software projects, teams and companies form such. It still has principles that can run either towards or contrary to its main goal – making software projects immune to the unexpected shifts in customer requirements, in market

demand and other external sources of disturbances. Agile has succeeded in delivering working software under strong pressure from different sources of volatility, but those projects are just gracefully degraded versions of the initial ones [9]. The next step is to be *antifragile* – to supply products better than the customer requirements or expectations, and the higher the force of the disruptions, the better the product. The growth of the product quality or the product market utility must be supralinear [15]. Is that possible to happen with the 12 principles of Agile [2] and implementations like Scrum [13]? If yes, why it did not happen so far? If not, what has to be changed? We try to answer some of this questions in this article.

II. PRINCIPLES OF AGILE

It is important to analyze the core principle of Agile in order to see how they correspond to the definition and the concept of antifragility. We start with the Agile Manifesto.

A. The Agile Manifesto

The agile manifesto has 12 principles, some of which are too specific and other either too vague or unquestionable like "continuous attention to technical excellence and good design enhances agility" which is both. We will focus in our analysis on the principles that are closely related to risk management of complex systems [2]:

- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- Working software is the primary measure of progress. Agile processes promote sustainable development.
- The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The value system of Agile consists of four points, three of them are important in complex systems context:

- Individuals and Interactions over processes and tools
- Customer Collaboration over contract negotiation
- Responding to Change over following a plan

These values and principles stand out as a framework for creating and implementing a universe of different methodologies such as Scrum, Kanban, Lean, Extreme Programming and many more [9]. All of them have contributed to the current, stable state of the software project ecosystem, which indicates that development of methodology must return to principles and reexamine them.

There are few additional key points, not explicit in the manifesto but used as derivative rules in the literature [9]

- Teams must be "cross-functional"
- The focus of Agile on short iteration cycles is oriented towards hyper productivity via maximum intensity – thus the cycles are called in Scrum "sprints"

B. Agile values and principles from the risk management point of view

The main risk management tool is the iterative nature of development. Software projects are of high complexity and dividing them into smaller sub projects decreases the risk of failure, budget overrun and time delays. Small projects have success of 76%, while large projects have only 10% success rate. According to [14], Agile projects have 4% failures for small projects and 23% for large, which I almost 4-fold increase as well as over 3-fold decrease in success rate from 58% to 18%.

TABLE I. AGILE PROJECTS SUCCESS RATES

Project success by size	**Project stat**		
	Successful	***Challenged***	***Failed***
small	58%	38%	4%
medium	27%	62%	18%
larg	18%	59%	23%

Even Agile projects are fragile in respect to project size – the success rate is decreasing and probably convex function. That makes the principle of accepting changing requirements **a** two-edged sword**.** As Fig.1 shows, scope creep increases with the years that quite possibly is related in part with the reactive attitude towards customer requirements because of their incremental nature. Here an important factor for risk management is to use the "simplicity" principle – only to remove and never to add features and complexity later in the project. It is our first suggestion for improvement and is related with antifragility of either of the project or the entire software projects ecosystem.

Another two-edged sword is the principle of building projects over motivated individuals. This is not complex systems approach - this is reductionist approach. Systems are much more dependent on their structure, than on their individuals. System design and interactions according Deming and his data from consulting hundreds of companies are the cause of 94% of problems and possibilities of growth and 6% are personal. Teams are not individuals – they are separate entities that need focus and motivation. Team performance and individual abilities are not directly linked, according to the studies, examined in [18] such as the Google Aristotle Project for

Identify applicable funding agency here. If none, delete this text box.

example. The dangers of focusing on individuals and managing only individuals are in two directions

- Making projects dependent on individuals and fragile relative to unpredicted absences due to illness or firm leaving. This is lack of focus on repetitive performance in software project development if the main cause is individuals who vary as people and qualities and not knowledge, which can only increase.
- Rewarding individuals for team performance that destroys cooperation. The link between individual and team performance is very complicated and subject of active research in the realm of electronic sports for example – and gives only probabilistic results [10].

Motivation and self-organization are key principles in Agile. Intrinsic motivation is what drives self-organization, and can be destroyed by personal ranking systems, replaced by bonuses and fear. Motivation is dependent on the climate of the organization, which is analogous to the critical temperature under which magnetization occur in the Ising model [4]. All teams in one company are co-dependent due to this "coupling factor" and innovations are adopted in the whole organization only when it is strong enough [5]. Even intrinsic motivation is not internal, and self-organization is not entirely "self". The climate *must* be appropriate for that – organization rules, structure and culture. The influence between an individual and the climate is asymmetric – the climate influences the individual far more than the individual influences the climate, and the difference increases rapidly with the size of the firm. Just putting the best people with high motivation will not guarantee self-organization. The focus over individuals and interactions is good principle, but the order needs to reverse:

"Organizational climate, interactions and individuals over processes and tools"

"Build projects around motivated individuals and teams. Give them the environment and support they need to stay aligned with the common goal, and trust them to get the job done."

"The best architectures, requirements and design emerge from self-organizing teams, which are product of well-organized companies"

The responsibility for a project cannot be entirely shifted down in hierarchy and the teams involved in different projects are not completely independent from the organization and from each other. This is shifting of fragility from top management to teams and team leaders and should be vice versa – the top management has to be able to fail, not the software projects.

Our definition of organizational climate is purely statistical:

Def. *Organizational climate* is the set of rules that produce the general system performance - the general probability density distributions of measures of quality and success.

This Deming labels "the common causes of variation" [6-7]. The quality of software and different measures have mixture distributions – a sum of general distribution and different specific distributions for different special causes of variation [19]. These can be as teams following different methodology. It could be a machine with some repeatable error pattern – a Pareto glitch (small percentage of equipment or code produces large percentage of problems). It could be individuals with performance out of the control limits in the positive or negative direction. In order to understand climate, one must use the statistical process control tools and follow the approach of Deming. The reason why in management literature there is debate whether organizational climate exists is that it is tied to perception, and not performance. We clear this now. There are common and special causes for variation of all measures of performance. The intrinsic part of the common causes of variation is the organizational climate. This is the variation that stays even if we replace 10% of employees every year and is people independent. It is the reason for the asymmetrical relation between individuals and organization in performance perspective.

Agile definition of progress is "not failing enough". Working software is a vague concept – it has no comparison with the initial specification and goal for the project. The reactive nature of Agile allows to save more projects from full failure and deliver working software, which is different than the initial goal and in the case of good management – with far less features. This is not progress, because is relativistic. Progress must be towards some fixed goal and must be continuous process not only between iterations, but between projects.

For example, our first Agile project delivered Software 1.0 and fulfilled 70% of our initial features. If Software 2.0 with the second Agile project fulfils 75% for the same quality of the software, this is progress. If it fulfils 70% with better quality, it is again progress. It can be a progress for the client who learn not to ask for the impossible or the unnecessary. It could be an improvement in teams and individuals, or even in the organization. Progress is a motion towards fixed goals with tangible improvements over previous moments in time. We propose our definition in software projects context:

Def. *Progress* is a continuous improvement in delivering working software closer to the initial specification and with better quality.

Working software should not be a primary measure of progress – it is only a necessary condition. To try not to fail is not enough as Fig.1 and Table.1 shows with the stability of the success rate and the fragility over the project size. Reactive and reflective behavior (reactive towards the customer, reflective towards the project problems and successes) are not enough in order to become antifragile. You need knowledge and theory.

III. WHAT IS ANTIFRAGILE

In this chapter, we introduce and analyze antifragility with the intent to improve the concept and apply it to agile methodology.

A. Definitions and properties

Antifragile is a term, coined by Nicolas Nasim Taleb [15] and rigorously defined in [16]. It is related to fragility, but is not the exact opposite. There are three concepts, defined by him: fragility, robustness and antifragility. The definition is probabilistic with the probability density distributions of the

payoff from the random events. The payoff or the utility function is nonlinear – shown on Fig.2.

- *Fragility* –it is a probability density distribution with fat left tail and thin right tail. Fat left tails leads to significant probabilities of very large negative payoff, and thin right tail leads to very small probabilities of very large positive payoff. Fragile things lose from variability and with enough trials (or time), the expected payoff is negative. This distribution of payoff is due to a concave payoff function from the most symmetric and thin tailed distribution of random events

  (Normal distribution as the most common). An example is success rate of project as a function of the size. Software projects probability of success decrease rapidly with their size, even for Agile projects. A small increase in project scope gives large increase of chances of failure, budget overrun, or time delay.

- *Robustness* is related to limited loss and thin-tailed left part of the distribution of the payoff. An example is the barbell method – a mixture of very low risk investment and very high risk investment. The minimum wealth is constrained above zero by the very low risk investment. The barbell does not let you fail. A bet with a constrained sum is also robust – you can not lose more than the money in it. The left tail below that negative payoff is not thin, it is non-existent. Robustness and fragility are related – robustness shows the level above which non-adaptive things like buildings become fragile. A building designed to withstand earthquake up to 8.0 magnitude are fragile relative to bigger earthquakes.

- *Antifragility* implies robustness in the left tail and fat right tale (thicker than the left) with unlimited positive payoff and significant probabilities for large positive payoffs. It is achieved with convex payoff function. An example is the life on Earth – the larger the size of disturbance (like the meteor 65mln years ago) and the larger the percentage of living organisms killed, the stronger and the more diverse the bio system becomes, with more adaptable species. There are two reasons for that. The strongest organisms survive (on average) and they transfer the best genes. This is the robust part. The time drives further evolution and speciation that enhances survival. Life on Earth is not destroyed by time, it is strengthened by it. Every successive generation improves genetic stability and decreases probability of harmful mutation. An antifragile system consists of fragile elements. The fragility of organisms gives the antifragility of species and life. Mammals on Earth have antifragility in respect to their size – the larger the animal, the longer its timespan. Risk of cancer per cell decreases with the increase of animals mass, instead of increasing, due to beneficial adaptations [17]. Risk of cancer per size of animal remains constant.

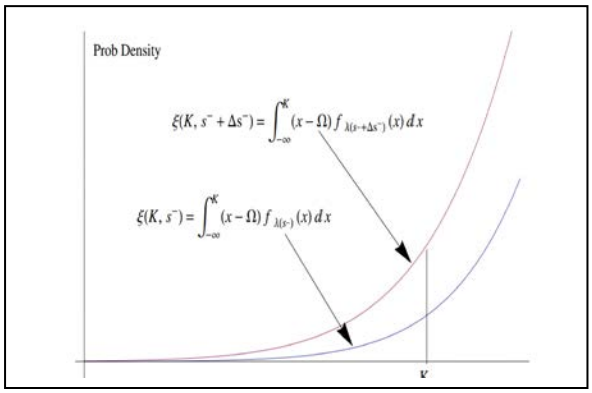

Fig. 2. Anifragility.Source: [16]

B. Antifragility of knowledge.

A very important example of antifragility is the nature of scientific knowledge in physics and mathematics, consisting in theory and experiment (yes, even in mathematics, discoveries are often made by experiment). As Eugene Wigner has pointed in [20], there are many examples of which a very small amount of experimental data gives rise to a radically new theories that explain old phenomena and predict new ones. An example is Newtonian gravity and the discovery of Neptune. Another is the general relativity theory and black holes. A third are the predictions of new elementary particles by application of group theory from mathematics in physics. A fourth is the application of conic sections to general relativity. In all of these examples, some part of the investigation is done in the absence of external data; with the latter two, in complete absence. Group theory was developed for other purposes, and conic sections were studied as purely intellectual pursuit. Knowledge with theory is capable of producing new information without any external input. This is the defining property of knowledge.

Def. *Knowledge* formed by theory and experiment is a system that has information, which is a convex function of the input information.

Since both random events and the payoff can be considered (meta) information, knowledge by that definition is antifragile. Its probability for large positive payoff (generated system information) is large, and it is limited in the left tail. Knowledge can only increase and this is the source of its robustness. By the Popperian standard scientific knowledge consists of fragile theories that are refutable and eventually destroyed by variability of experiment and evidence. The refutation of a theory is in itself new information, thus knowledge increases in both case, positive and negative. Thus the knowledge itself is antifragile. Two key properties of knowledge give the convexity of information:

- Scientific knowledge produces information out of lack of information by the crucial ability to distinguish randomness from information. This is meta-information. Knowing that you do not know is the Socratic surplus of information. It is crucial, as randomness is fundamental part of nature. It also gives the ability to Meta-predict – to predict in which case

you cannot predict. An example is in the work of Nassim Taleb for rare events, description areas in which Black Swan appear – unpredicted but not unpredictable. Another is in chaos theory and Lyapunov time, where an exact period of predictive behavior can be found.

- Scientific knowledge produces new information internally in the absence of information. It is proactive, and not reactive. Reactivity in itself does not guarantee antifragility. It only adds new information, but does not produces it. This is special property of mathematics with its ability to produce proofs and certain results. The mathematical nature of physical laws allows this information to transfer to scientific knowledge in physical domain without the need of stimulus from empirical data. Application of Group theory in quantum mechanics is a prime example.

These properties of knowledge comply with the definition of antifragility by Nassim Taleb but run against his examples, where theory is in the fragile domain. A combination of theory and experiment is antifragile in both the simpler, "linear" domain of physics and in the field of complex systems and the domain of Black Swan. Taleb's research in itself is an example of usage of theory for generating meta-information about limits of predictability. Extreme value theory and complex systems give us the opportunity to predict where we cannot predict, what is predictable and what – not. Thus, Deming is right in his view of the role of prediction in management [7]:

> "Any decision that management makes, that anybody makes for himself or for other people, is prediction. The simplest plan is prediction, with a chance to be wrong. How may I get home tonight? I predict that my automobile will start and run, or that the bus will come, or that the train will come. I make plans. Those plans are predictions. Management is prediction; our lives are prediction. We predict what will happen. We try to choose a course of action that will react in favor of us. That's our aim. We predict the consequence of actions."

Prediction is essential part of proactive behavior and necessary condition for antifragility. We must be able to predict where no prediction would work. In that we differ from Nassim Taleb – we put Socrates before stoics.

IV. RELATION OF AGILE TO ANTIGRAGILE

Is Agile fragile, robust or antifragile? This question is improperly framed. Software projects can be fragile, robust or antifragile. The software ecosystem can belong to one of these categories. Agile principles can lead software project methodologies towards some of these categories. The question is how easy they will lead to a certain outcome.

Antifragility and robustness are defined mathematically, but these are properties, directly linked to choice. An antifragile system built of fragile elements such as a multicellular living organism has choices – which and when to kill its cells with apoptosis. The cells do not have choice. They are programmed to live or die. They are fragile. If the cells become independent and have choice, we have cancer – and then the organism will become fragile.

An ecosystem of software projects is antifragile only if they are fragile. If software projects are able to fail on time and often, the rate of failure will decrease in time. Instead, they are very rarely failing, because of the additive version of Agile management leading to continuous increase of scope creep and constant rate of failures. Software projects are antifragile, but software project management is fragile. No lessons are learned, no overall improvement overall is happening. In an antifragile system like the airline industry [15] every airplane crash decreases the probability of the next crash because proper measures are taken based on analysis after each one. One of the major reasons is that planes can and do crash.

A. *Right and wrong antifragility*

The wrong antifragility of Agile software projects is that they react to changes in customer requirements by growing too large and complex – hence the scope creep. The scope size gains from the variability in scope definitions [1], since Agile adjust the goal of the project dynamically. It is robust in the left tail (project scope is bounded from below) and the right tail is not bounded (in reasonable limits) and fat, due to positive feedback. Every new added functionality leads to more functionality that leads to a power law distribution of connections. In OOP it is a double Pareto distribution [8] in which small number of classes have most of the connections.

The right antifragility is of quality, not of size. Quality must increase with every change of customer requirement. A necessary condition for such antifragility is only to subtract and never to add functionality. It is necessary for the antifragility both of the project (up to a point) and the software ecosystem. The project can only gain from the changing requirements if it drops functionality and scope. The initial specifications in project themselves are antifragile – the size and complexity are robust in the left tail and fat tailed in the right. There is no limit of what non-technical persons can ask to be done in a limited amount of time, money and constrained set of people. Dropping functionality can help antifragility up to a certain point – below that the project will fail. For the common interests of developers and companies such projects must be left to fail as early as possible – when the customers want something that is discovered impossible or implausible. This is an incentive for Agile teams and companies to **invent** as much as possible. Thus the downside will be small and bounded and the upside is unbounded. This is at least robust. For antifragility, there are other necessary conditions, which Agile needs to fulfill.

B. *Agile principles and antifragility – a comparison*

- *Cross-functionality*. Antifragility is about having more choice. Cross-functional teams have choice and can adapt to varying circumstances. Having Cross-functional teams can be antifragile in project development towards quality. A cross-functional team does not consists of cross-functional people, however. The fragility of the elements assures the antifragility of the system. If the elements are antifragile, the system is fragile. Co-dependency and specialization are necessary for convex utility functions [3] – people need

to become experts to contribute maximally. For convex functions, the function of the average is less than the average of the function. If skills and expertise are more equally distributed, since they are constrained (people have finite capacity to learn for a given timeframe), then the convex function will have smaller value. We can give a simple example: two skills and quadratic utility (a sum of squares of skills) and the total sum per agent and the minimum sum of skills for the system is 10. If we have two agents with skills (5,5) the result would be 100, and if we have (10,0) and (0,10 the result would be 200. If the people are "cross-functional" and have the full set of skills, they have more choice. They can work for themselves either inside the team or outside it. If employees do not depend on each other, there is no incentive to cooperate.

- *Constant pace of work and regular intervals for iterations*. The size of the iterations is one of the things that makes the project fragile – larger sub-projects, higher probability of failure. The other is the regularity of the iteration sizes. Many software projects cannot be split uniformly and that leaves some of the iteration much harder and therefore riskier than other. Iterations need to be split accordingly to balance the risk – and that can be done dynamically. It is complementary to the drop of functionality – some iterations may end sooner than expected when scope is adjusted to reality instead of reality adjusting to the scope. Constancy of pace is very ambiguous idea since it implies marathon like efforts and is understood in applications like Scrum as series of sprints. Sprinting is high intensity effort that cannot be repeated indefinitely with constant rest intervals unless they are much larger than the sprinting duration. They are self-limiting processes. The sprints are used for "hyperproductivity" which is actually "hyperfragility". Optimized systems are fragile, redundant systems are antifragile. Working close on the edge of the possible increases exposure to the harm of the unpredictable. Machines do unpredictable errors predictably - people do unpredictable errors unpredictably. The number or errors is convex to the working pace due to the limited capacity of people and therefore the quality is concave to it. The time for repair is convex to the number of errors (limited capacity of people). There is need for time redundancy – time between sprints and projects, and time in sprint planning. More time will lead to less errors and more time to fix them, thus overall improved productivity. Less time leads to many more errors and less time to fix them, thus less productivity. This will also prevent excessive narrowing of the scope due to reduced volatility of project execution.

- *Lack of theory*. In order to have convexity of payoff in project management, the causes of every problem needs to be identified. One necessary condition for that is to have a systematic theory, in order to distinguish randomness from information. Deming calls this "common and special causes for variation" [6]. Common causes are from the variation of the inputs of the system or its internal structure. Examples for the inputs are market volatility and customer requirement changes. The internal structure is the organizational climate. Special causes are faulty machines or people doing something outside of the general rules for work procedures. Successful separation of those two causes allows removal of special causes for bad performance and learning from special causes of good performance. Probability and sociophysics are also necessary tool for understanding the team and project dynamics in order to make better predictions or meta-predictions, which are crucial in the initial specification phase. The better we can predict cost and duration of project under certain conditions, the smaller the risk even if these conditions are not met. Better conditional prediction implies better understanding of the project. There is a need of systems theory for software development processes that will unify these two approaches.

- *Reactivity*. Reactive behavior is necessary, but not sufficient for adaptation and antifragility. Adaptation is proactive. Adaptation includes overcompensation that can be understood as prediction for the magnitude of further disturbances. An example is strength training which produces overcompensation, as the body "predicts" that in the future the load could be heavier and prepares for it. This is redundancy of strength. Overcompensation is not optimal; it brings redundancy (higher energy expenditure). The team needs not only to respond to the changes in customer requirements, but also to predict them partially, to overcompensate for the goal and produce better than expected and earlier than expected. This overcompensation should not be in scope, but in quality or in the antifragile case – with innovation. As Henry Ford said, no customer ever asked for an automobile. They all wanted faster hrses.

V. TRANSFORMATION OF AGILE

Here we give our propositions in order to transform Agile methodology into antifragile one.

In addition to our previous propositions of updating principles and values of Agile, we propose a few more in order to help transitioning to robustness and possibly antifragility. We propose this update of a basic value of Agile:

"Preparing for and embracing change and innovation over following a plan".

We propose an update of the reactive principle of Agile:

"Prepare for and welcome changing requirements, even late in development. Agile processes harness moderate change, innovation and downsizing projects for the customer's competitive advantage."

In order to transfrom from fragile/robust into antifragile, the Agile values need to be updated with the following one:

"Theory and experiment over belief and rationalization. Data and knowledge over ideology and personal opinion"

The principle of reflection needs to be updated towards proactive learning:

"*At regular intervals, the team gathers and analyzes the total project data to become more effective, then tunes and adjusts its behavior accordingly.*"

It is important at every step to analyze the total volume of project data in order to prevent influence of randomness and sampling size in decision-making [6].

REFERENCES

[1] S.Amjad et.al., Calculating Completeness of agile scope in scaled agile development, 2017, IEEE Access. PP. 1-1. 10.1109/ACCESS.2017.2765351.

[2] K.Beck et.al. Principles behind the Agile Manifest, 2001, https://agilemanifesto.org/principles.html

[3] D.Coviello, A.Ichino, N.Persico, Measuring the gains from labor specialization: theory and evidence (June 28, 2018). Available at SSRN: https://ssrn.com/abstract=3204848 or http://dx.doi.org/10.2139/ssrn.3204848

[4] S.Galam, Y. Gefen and Y. Shapir, "Sociophysics: A mean behavior model for the process of strike", Math. J.of Sociology 9, 1-13 (1982)

[5] S.Gonçalves, M.Laguna, J.Iglesias. (2012). Why, when, and how fast innovations are adopted. The European Physical Journal B. 85. 192. 10.1140/epjb/e2012-30082-6.

[6] W.E.Deming. Out of the crisis, July 2000, MIT Press

[7] W.E.Deming. Management today does not know what its job is (part 2), January 1994, IndustryWeek, https://www.industryweek.com/quality/dr-deming-management-today-does-not-know-what-its-job-part-2

[8] I.Herraiz, D.Rodriguez, R,Harrison, On the statistical distribution of object-oriented system properties. 2012 3rd International Workshop on Emerging Trends in Software Metrics, WETSoM 2012 - Proceedings. 56-62. 10.1109/WETSoM.2012.6226994.

[9] P. Measey - Agile Foundations - Principles, practices and frameworks, O'Reilly, 2015

[10] T.Minka, R.Cleven, Y. Zaykov, TrueSkill 2: An improved Bayesian skill rating system MSR-TR-2018-8 https://www.microsoft.com/en-us/research/publication/trueskill-2-improved-bayesian-skill-rating-system.

[11] Pmi.org. 2018 Pulse of the Profession | PMI. [online] Available at: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2018.pdf

[12] D. Russo, P.Ciancarini, "A Proposal for an antifragile software manifesto", Procedia Computer Science, Vol.83, 2016, po.982-987

[13] M.Sliger, Agile project management with Scrum, Paper presented at PMI® Global Congress 2011—North America, Dallas, TX. Newtown Square, PA: Project Management Institute.

[14] The Standish Group. The Standish Group CHAOS Report. The Standish Group, 2013. Retrieved from https://www.projectsmart.co.uk/white-papers/chaos-report.pdf.

[15] N. N. Taleb. Antifragile. Random House, 2012

[16] N.N. Taleb, R. Douady. Mathematical definition, mapping, and detection of (Anti)Fragility. 2014. ffhal-01151340f https://hal.archives-ouvertes.fr/hal-01151340/document

[17] M.Tollis, AM. Boddy, CC.Maley. Peto's Paradox: how has evolution solved the problem of cancer prevention?. BMC Biol. 2017;15(1):60. Published 2017 Jul 13.

[18] L. Tomov, System complexity and system quality for project Management, Computer Science and Education in Computer Science, 2017, Vol.1, pp.245-255

[19] D.Vicentin, S.Bezerra I.Piccirillo, O. Fernanda. Monitoring process control chart with finite mixture probability distribution: An application in manufacture industry. International Journal of Quality & Reliability Management, 2018. 35. 00-00. 10.1108/IJQRM-11-2016-0196.

[20] E.Wigner. The unreasonable effectiveness of mathematics in the natural sciences. Communications in Pure and Applied Mathematics, 1960, 13:1-14.