Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 982 – 987

Procedia Computer Science

3rd International Workshop on Computational Antifragility and Antifragile Engineering (ANTIFRAGILE 2016)

A Proposal for an Antifragile Software Manifesto

Daniel Russo[a,*], Paolo Ciancarini[a]

[a]*Department of Computer Science & Engineering - University of Bologna, via Mura Anteo Zamboni 7, Bologna, Italy*

Abstract

The disruptive nature of the antifragile approach for open and complex systems is of greatest importance and needs to be systematized, especially for software systems. In fact, antifragile software design is becoming a research issue in the software engineering community. We got inspired by the Agile Manifesto which set an important reference point to the software community, addressing primarily innovation in the software development process.

We propose a similar approach to Antifragility, namely we would like to define the principles ruling the building up of software systems which exploit faults and errors to become better and stronger. This Manifesto does not want to be a fixed and complete set of principles. It is an open contribution to the discussion which needs to be improved and re-elaborated. All rights related to the Manifesto are free, open and belong to the community. This work represents our suggestions urging the community to start elaborating antifragile principles to lead their implementation in real organizations.

Peer-review under responsibility of the Conference Program Chairs

Keywords: Complex Systems; Software Engineering; Antifragility.

1. Introduction

Antifragility is a design principle - and a quality of some systems - elaborated firstly by N.N. Taleb in 2012[8]. The most interesting aspects of Antifragility is that an antifragile system is able to evolve its identity in order to improve itself systematically in its operating context. An antifragile system does not just "resist" to external stimuli which activate design errors or pose new problems: it improves solving them. "Antifragility is stronger than resilience or robustness. The resilient entity resists shocks and stays the same; the antifragile entity gets better."[8].

A large mission critical organization like NASA is actively trying to adopt Antifragility within its organization and systems[5]. According to NASA's leadership, it is clearly emerging the need to define and formalize the paradigm of a new antifragile software architecture to "address future challenges [...] since it is not simply to do what we know how to do now better: we need to do things we currently do not know how to do"[5]. To tackle this issue, the present work has as its main goal to define Antifragility to the community of software engineers as a novel approach

* Corresponding author. Tel.: +39 051 2094861 ; fax: +39 051 20 94510.
E-mail address: daniel.russo@unibo.it

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2016.04.196

for designing open and complex systems. Some works have already been carried out in literature[1 2 3 4]. However, a common framework is missing.

This paper proposes a Manifesto for antifragile software both to systematize the knowledge acquired and to start developing a common framework for Antifragility.

The rest of this paper is outlined as follows. Section 2 explains why a Manifesto would help the community to outline this paradigm changing way of doing software. In section 3 we propose the principles for the Manifesto. Finally, in section 4 we briefly describe our main goals for the future.

2. Why a Manifesto?

A Manifesto is an affirmation of some principles concerning a domain. For instance, the Agile Manifesto wants to persuade software engineering professionals about how to build software in more effective ways to be implemented in their organizations. Many agile software development methodologies refer to and got inspiration from the Manifesto and many other are coming up[6].

The principles we propose for antifragility are just our "suggestions" to start the discussion and raise attention around Antifragility as a software design concept. The spirit should be quite similar to the Agile Manifesto, as need to serve the community. Since the antifragile community already built up a relevant amount of knowledge, it may be time to start systematizing common principles.

We suggest that Antifragile can be seen as an extreme way of doing Agile. The numbers of principles chosen (12), as also the starting point (e.g., the customer, the team) are explicitly taken from the Agile Manifesto. The reason of this is that according to our view, the Agile way of thinking is essential to build an antifragile organization. In this instance, from a software engineering perspective, Agile is a prerequisite on which antifragile software development is build on. However, it would be quite reductive to see Antifragility as a sub domain of Agile. This, for two main reasons, an intrinsic one and an extrinsic one:

- *intrinsic motivation*: the level of awareness and paradigm changing mentality. Loving errors means much more than having an open and direct discussions. It means to build structurally not only a culture of trust but to accept proactively failures, in order to improve the whole system or organization. The learning by doing process enables people to gain from effective experience and not just from hypotheses.
- *extrinsic motivation*: the application domains of Antifragility are larger than Agile. For instance, the most relevant aspect is that the main goal of Antifragility is to learn from errors to become stronger. Instead, Agile's main goal is to build in an efficient and effective way a software system (or an organization). For Antifragility this is already given for granted, it is a pre requisite.

For this reason we are proposing a framework to let people discuss about the principles of designing antifragile software.

3. Some suggested principles

In this section we will briefly outline the main principles of the Antifragile Software Manifesto. Aim is to define 12 principles that, likewise the Agile Manifesto, will lead the development of a new kind of software. All the principles are proposed according to our experience in Agile development processes and our elaboration of Taleb's theories. We are perfectly aware that a larger discussion by the community is needed to elaborate a Manifesto. Thus, these principles has to be considered as a contribute to the discussion.

3.1. The customer

The first principle regards the customer, who should be the end side of the process and he who profits more from its implementation. In order to doing this, the product has to be non linear, active and self adaptive. This means that software has to be:

- *non linear*: inputs are crucial source of learning. So the system has to be designed for considering input not just as orders to execute, but as structural elements of the system Antifragility.
- *proactive*: it has to be proactive in the the recognition of the error as element for the system improvement.
- *self adaptive*: after the recognition of the error, it has to improve, adapting the system to recognize future similar error to be antifragile.

The main priority of Antifragility is so the satisfaction of the customer's needs, delivering an antifragile system.

P. 1 – Our highest priority is to satisfy the customer by building a non-linear, proactive, and self adaptive system

3.2. The context

The software development process has to be deeply aware of the context. So, context awareness means to recognize that high impact and unexpected events (i.e., Black Swans[7]) as the key elements of a system (as also an organization). This means that a system, to be antifragile, has to be designed according to Principle 1, welcoming and adapting to changing scenarios. It is the experience that characterizes an antifragile system, and experience is gained by changing scenarios. A welcoming aptitude to changes is, at the end, the best way to leverage from Antifragility.

P. 2 – We welcome changing scenarios where unexpected events (Black Swans) are the real paradigm shifting entities

3.3. The tolerance

From a software engineering perspective, embedded fault tolerance is one of the main characteristic of Antifragility. It is an aspect which is widely studied in literature for decades[9]. Fault tolerance, intended as detection and correction of errors has to be both adaptive and embedded. These two elements are an intrinsic feature of any antifragile system.

P. 3 – We deliver assuring embedded and adaptive fault tolerance

3.4. The stakeholders

Like in any organization, the support by the stakeholders is pivotal to turn projects into action. So, it is also in the antifragile one. However, to build an antifragile system leading inputs do not only come from stakeholders, but also from the environment. In this instance, it is a sort of open system, which gains also from external stimuli. It is not only the organization itself which leads the system, it is also the broader environment which shapes continuously it. These two dimensions can be summarized as follow:

- *internal – the organization*: it is the active part. The design, development and testing still rely on the organization, which has to carry out the system.
- *external – the environment*: it is the passive part. The broader environment is the natural source for errors, so the primary learning arena.

These two dimensions are perfectly complementary. The internal one sets up the system, while the external one shapes it, making it antifragile.

P. 4 – All stakeholders, and the broader environment, lead the antifragile organization

3.5. The team

The development team is the core asset of the organization which intends to implement an antifragile system. The team will take care of the architectural design up to testing, trying to keep aligned with the organizations needs. For many aspects, the team is the crucial element of the internal dimension. Therefore, people need do be motivated to deliver an antifragile system and also skilled in doing it. Open mindedness is important to deliver a product which is non linear per definition. Being open to something new is the only way to build a system which gets stronger learning from faults.

Furthermore, both the environment and team support are crucial to get the job done. Since they are perusing a new path, blames by colleagues may harm their motivation and put in danger the whole project. Trust is the glue which links the team with the organization to reach the expected goal.

P. 5 – Build antifragile projects around motivated, skilled and open minded people. Give them the environment and support they need, and trust them to get the job done

3.6. The communication

Trust by the organization is not enough for building an antifragile system. In fact, if the team's familiarity is quite loose, their team spirit is weak. Enhancing team building is essential to let people focus on the work and not in negative social interactions. Even if there is no blame from the organization or other colleagues, developing something new and challenging needs also an internal support.

The best way to support familiarity within the team is communication. For this reason honest, open and transparent communication is the way to build such a pioneering team. It has to be honest because no misunderstanding of the proper skills need to be doubted. Since any team member has to relay to the other, openness is the way to interact with each other. No kind of social constraint should hinder this. Therefore, it is also transparent, where any team member is aware of the others communication. There is no missing information and all information are aligned to all members.

P. 6 – The most efficient and effective method of building an antifragile organization is building on honest, open and transparent communication

3.7. The exposure

An antifragile system is *per se* exposed, to improve its functionality. On the other hand, automatic fixing is the outcome of the process. These two elements, input and output are the primary measures of the antifragile system.

It is a measure of functionality, in terms of efficiency of the system. So it is a product measure. Furthermore it is also a process measure, since the progress of the development could be tested.

P. 7 – Continuous exposure to faults and automatic fixing is the primary measure

3.8. The maintenance

An antifragile organization is led by the experience, which is shaped by the context. Thus, it promotes an environment which is aware of the context. At the end, organization are formed by people and each individual take part to

build the antifragile organization. So, only if people are fully aware of the environment, they are able to build such an organizations, which implements such systems.

Therefore, the maintenance of a system should run indefinitely. This for three reasons:

- the system is self adaptive, so it grows like a human being. It receives stimuli which allows to make it stronger.
- it needs care. Like any human being, it may not be auto sufficient, it needs adjustments for its evolution. This level of attention may be only given by a context aware organization, which perceives the contingency of Antifragility.
- it uses ontologies. Ontologies are non linear and scalable technologies. Thus, they are the ideal ground for the adaptivity process. Such technologies represents a way to classify and recognize ongoing phenomena which are part of the system fitting.

To really achieve this, it is not only an effort of the organization but of the stakeholder. This is a broader definition for all people which are involved in the organization. So employees, management and shareholders. If they are engaged, maintenance should not represent an issue.

P. 8 – An antifragile organization promotes a context aware environment. The stakeholders should be able to maintain a system indefinitely

3.9. The dimensions

There are three main dimension for building an antifragile system. In detail they are:

- *technical excellence*: skills are not just required by people, also from organization. An organization which does not care about technical excellence do not feel Antifragility as really contingent. Employees needs to relay on both cutting edge technical tools, as also to the organization's attention to solve customer's problems. Even to those problems that are not incurring yet but may incur soon. On this instance we also find clear examples in literature[4].
- *reality*: the sense of reality leads the team and the organization in the definition of priorities and budget.
- *redundancy*: the practical effect of non – linearity. Antifragile systems do not evolve linearly. They are redundant, profiting from continuous and similar stimuli.

P. 9 – Continuous attention to technical excellence, reality, redundancy

3.10. The error

Errors are the primary source of an antifragile system. They shape the system letting become it antifragile. Errors are the input of any antifragile system, becoming a learning element. A system without error can not be considered as antifragile, since there would not be any input which lead to a system improvement.

In front of an error e.g., wrong input, the system do not block itself. Runtime adaptation mechanisms are active parts of the error-loving architecture. A antifragile system should not block itself in front of unknown inputs, it has to accept them and to use machine learning algorithms to improve its efficiency.

Therefore, error loving does not just mean to be open to them. They are not just a likely option. Errors loving is, in last instance, the deep essence of Antifragility.

P. 10 – Error loving - the art of learning to be antifragile – is essential

3.11. The architecture

The architectural design of an antifragile system has to be non linear. Non linearity, in this instance means to have the awareness that Black Swans are paradigm changing entities. For this reason, no out the shelves solutions are practicable.

Teams have to self – organize themselves according to the context in which they are operating. This is a way to avoid linear fallacies, since there is no one way to design an antifragile system.

P. 11 – Antifragile architectures emerge from self – organizing, context aware teams

3.12. The reflection

This characteristic of Agile, may be also used for Antifragility. Context is continuously changing. Starting assumption while designing a system may change during both development and maintenance life cycle. So, reflection about the features helps in delivering and maintaining antifragile software.

P. 12 – At regular intervals, the developing team reflects about the context situation, on how to become more effective, then tunes and adjusts its behavior accordingly

4. Conclusions

This paper is a proposal for an antifragile Software Manifesto. Aim of this contribution is to start a discussion within the community to elaborate principles which should lead practitioners in the developing of an antifragile organization.

In our proposal we addressed the Agile Manifesto as disruptive software development methodology. However, in our view, Antifragility is not a sub domain of Agile for two reasons. The first one, regarding the intrinsic motivation, because there is a paradigm changing in terms of loving errors and non linear design. The second one, extrinsic, concerns the application domains, which are multiple and very diverse.

We hope to get some consensus in this proposal to start outlining a software Antifragility development methodology.

Acknowledgements

This paper has been partially supported by MIUR PRIN IDEAS and the Consorzio Interuniversitario Nazionale per l'Informatica (CINI).

References

1. V. De Florio, *Antifragility = Elasticity + Resilience + Machine Learning*, Procedia Computer Science 32(1), pp. 834 – 841, 2014.
2. V. De Florio, *On resilient behaviors in computational systems and environments*, Journal of Reliable Intelligent Environments, 1(1), pp. 33–46, 2015.
3. M. Monperrus, *Principles of Antifragile Software*, eprint airXiv:1404.3056, 2014.
4. M. Monperrus, *Software that Learns from its Own Failures*, eprint arXiv:1502.00821, 2015.
5. K. H. Jones, *Engineering Antifragile Systems: A Change In Design Philosophy*, Procedia Computer Science, 32(1), pp. 870–875, 2014.
6. D. Russo, *Benefits of Open Source Software in Defense Environments*, Software Engineering for Defence Applications (SEDA) 4th International Conference in, May 2015.
7. N. N. Taleb, *The Black Swan: The Impact of the Highly Improbable*, 1st ed. Random House, Inc., New York, 2007.
8. N. N. Taleb, *Antifragile: Things That Gain From Disorder*, 1st ed. Random House, Inc., New York, 2012.
9. D. J. Taylor, D. E. Morgan, J. P. Black, *Redundancy in data structures: Improving software fault tolerance*, IEEE Transactions on Software Engineering (6), pp. 585594, 1980.