Agile and software engineering, an invisible bond

Alvaro Ruiz de Mendarozqueta[1], Fabio O. Bustos[2] and Pedro E. Colla[3]

[1] aruizdemendarozqueta@gmail.com,
[2] fabio.oscar.bustos@gmail.com,
[3] colla.pedro@uader.edu.ar
Universidad Tecnológica Nacional – Regional Córdoba[1]
Córdoba – Córdoba - Argentina

Abstract. The bond between agile practices and Software Engineering practices is clear and apparent for seasoned practitioners with experience in the operation of high maturity development environments, yet it is often ignored on the domain bibliography where mostly hybrid approaches are adopted. This article reviews a sensible sample of the bibliography to confirm that trend and develop a map between long-established Software Engineering practices on the one hand, and concepts stated as agile foundation principles on the other. Previous research efforts are integrated into reinforcing those aspects of an agile-based project which need to be addressed with priority in order to protect the additional value yield by the usage of these methodologies.

Keywords: Agile, System Modelling, Software Engineering, Real Option Value

Abstract. La relación entre metodologías ágiles y las mejores practicas recomendadas por la Ingeniería de Software es clara y evidente para los profesionales experimentados en la operación de ambientes para desarrollo de alta madurez. Es sin embargo a menudo ignorada en la bibliografía del dominio donde se tratan como si fueran enfoques separados. Este artículo revisa una muestra significativa de la bibliografía y confirma esta tendencia desarrollando un mapa de cuales son las practicas establecidas de Ingeniería de Software y los conceptos subyacentes que operan en las metodologías ágiles. Se integran esfuerzos previos de investigación para reforzar que aspectos de la gestión de proyectos basados en metodologías ágiles necesitan ser abordados con prioridad de manera que el valor adicional que las mismas generan resulte protegido.

Keywords: Metodologías ágiles, modelado de sistemas, Ingeniería de Software, valuación por opciones reales.

Background

In order to achieve their business goals, the organizations need to implement technologically advanced software-based platforms; often needing to, partially or totally, develop them to ensure they meet the business requirements as set by the competitive landscape.

Software development is, to some extent, a low maturity engineering practice; at least compared with other branches of the engineering domain. Metrics shown by the industry, in terms of schedule compliance, cost containment, and ability to meet requirements are in general terms far from what is considered acceptable in other industries (Jorgensen K. M., 2003).

Over time, good practices emerged, aiming to improve different aspects of the software development cycle, which eventually evolved as a cohesive body of knowledge known today as Software Engineering (Fairley & Bourque, 2014).

In order to avoid subjectivity into the measurement of the organization's compliance with recommended practices, different reference models such as CMMI ™ (Team, 2010), COBIT (ISACA, 2018) or even tailored versions of more generic quality frameworks such as ISO-9000 (ISO, 2020) evolved. Such reference models and standards were eventually used to objectively compare an organization's capabilities, and to mitigate the software development risks through the deployment and systematic usage of process practices and goals. The strategy to implement Software Engineering disciplines using convergence to reference models were embraced by large industry players, eager to show up their capabilities to mitigate risks, as a competitive edge compared with other vendors unable to show the same strength.

A rigorous deployment and institutionalization of a formal process reference model, and the discipline and costs associated with maintaining it over time, were adopted by a relatively small number of players willing to do the long-term commitments and investments required (M. Staples, 2007).

Other organizations, either because of lack of scale, or because software development was not within their main domain of competences, found it difficult to justify the investments required to embrace a formal process quality framework as their primary strategy to achieve their business goals. However, at the same time, these organizations still need to develop software as a crucial component of their competitiveness, or even their survival; but they identify the formal and rigorous adoption of Software Engineering premises as way too costly to afford; at the same time, they might be impacted by cost, time and quality issues derived from using a less rigorous methodological approach.

Agile methodologies all of the sudden stormed into the Software Engineering landscape as an attractive solution for small and medium businesses, which become able to achieve reasonable performance into grasping the value out of their software development efforts with a relatively small investment and organizational effort to institutionalize (Cockburn A., 2007). There is no surprise in the huge adoption rate in the industry.

Under a close study, the value proposition of the agile methodologies shows that their main advantage is coming from introducing some formal and strict development framework into the project execution. This factor can be further understood when it is possible to map that, by using any popular agile methodology, such as SCRUM, most of the requirements for an organization to demonstrate compliance with CMMI ™ level 3 can be demonstrated (McMahon, 2010). A significant number of organizations can map the usage of agile methodologies as part of their roadmap to achieve higher levels such as CMMI ™ Level 5 (McMahon, 2010) (Maller, C.Ochoa, & Silva, 2004). This is confirmed by the professional experience of the authors applying agile methodologies on environments operating at SEI-CMMI Level 5 maturity level, and seeing no contradiction whatsoever among them.

Besides the benefits from a more rigorous project execution being introduced into the development process, the flexibility to quickly align and adapt the software development activities to the business priorities; that seamless decision capability also yield value to the project and can be successfully modeled using a financial instrument called "real options" which assess the value gained by the organization by continuously decide ways to optimize their outcomes. When this evaluation is made, a significant increment in the project value emerges from this factor ((Beck & Boehm, Agility through Discipline: a debate, 2003)) (Colla P., 2012) (Colla P. , 2016).

The additional value proposition is not coming without some problems on their own, as a key understanding and strict adoption of the methodologies involved are still required. Different authors (Ismail, 2016) (Bhasin, 2012) (Miller, 2013) (Caballero, Calvo-Manzano, & Feliu, 2011) discuss problems faced by agile methodologies in terms of delays, additional costs, and product quality issues, as well as the existence of significant product backlogs. These are, basically, the issues Software Engineering has historically evolved to address.

In the professional experience of the authors, the association between agile methodologies and Software Engineering practices is often rejected by agile practitioners as not compatible, even further, in plain contradiction. Especially when the overall perception leads to the notion that most of the flexibility provided by agile methodologies can be lost if paired with Software Engineering concepts.

The authors will address in this article the intuition that a strong, albeit sometimes hidden, bond does exist between Software Engineering practices and agile methodologies, using SCRUM as the reference methodology for such analysis.

Agile and Software Engineering relationship at a fundamental level

The traditional approach has been that software is a tool for organizations to improve their internal productivity through automation efforts. The current competitive landscape drives the need for a platform to improve or even be part of the value chain to produce their income, and therefore being subject to continuous competitive pressure to innovate in very short times. This is a very volatile context where the development methodology has to support very fast development cycle times.

Ever since Ken Beck developed the ground rules of the agile methodologies, till their current massive adoption level, the bibliography proliferated with platforms, usage guidelines, strategies to implement, and practical examples in different industries (Rico, 2008) (Cohen, et al., 2004) (Pikkarainen & Passoja, 2005)(Pikkarainen & Mantyniemi, 2006) (Rico, s.f.) (Favaro, 2003) (Favaro, 2004).

The agile approach, which is contained as part of the Agile Manifesto (Beck, et al., 2001) (Duncan, 2019) prioritizes individual actions and their interactions over process and tools, leverage the software as documentation, cooperation, and close teamwork with the customer (product owner) above negotiation and, perhaps the most significant component, incorporate change into the methodology rather than opposing it following a pre-defined plan.

"*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*

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

That is, while there is value in the items on the right, we value the items on the left more. "
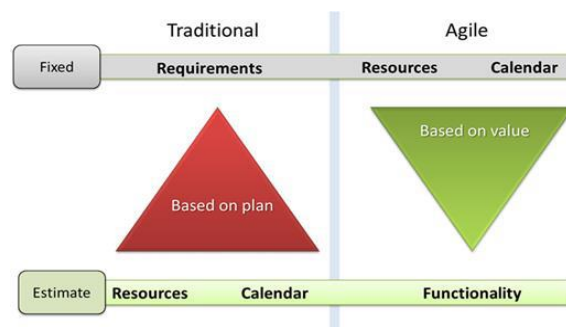

Figure 1 Agile conceptual modeling (Morse, 2012)

Given the known problems of traditional software development such as massive delays, products that did not fulfill their purpose adequately after years of development, and cost overruns, a group of pioneers thought of a radical paradigm shift. The traditional paradigm tries to establish the requirements comprehensively at the beginning of the project, whose duration is fixed, and then to estimate, based on the development plan, the effort, the necessary resources, and the schedule to be fulfilled.

There are multiple examples of failure, delays, and problems in such a paradigm. In the new paradigm (Cockburn A. , 2007), as shown in Figure 1 Agile conceptual modeling , a fixed time window is established, a small team of developers is organized and functionality is continuously evaluated, with the permanent help of the "owner" of the requirements providing the necessary sponsorship.

The manifesto is complemented by 12 principles that highlight some fundamental ground rules such as customer integration in the development process, ownership by the entire team of everything that is produced, and a sustainable pace of work.

In brief, the dominant principles are:

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

Efforts have been made to establish a structured time retrospective on the evolution of agile disciplines and software engineering practices (Agile Alliance, 2020), but we have preferred a more holistic approach based on a group of relevant bibliographic references in the judgment of the authors.

It comes as not a surprise the manifesto is solidly supported by the practices and principles of software engineering. Albert Endres and Dieter Rombach (Endres & Rombach, 2003) say that 'Requirement deficiencies are the prime source of project failures' so interactions and customer collaboration are critical for project success. This statement is covering principles 1 and 4.

Gerald Weinberg (Weinberg, 1992), reviewing different definitions of quality conclude that 'Quality is value for some person', covering principle 1. It is also related to principle 4 because delivering working software soon is the way of adding value to the customers which, far from being a surprise, is strongly supported by value management financial principles involving time and risk as to the main contributors or detractors for it (Brealey & Myers, 2016)

In a classic paper Davis (Davis, Bersoff, & Comer, 1988) remarks that 'For every application beyond the trivial, user needs are constantly evolving. Thus, the system being constructed is always aiming at a moving target'. This statement not only supports the manifesto values but also addresses principle 2. Another source for supporting principle 2 comes from the very CMM foundation as Watts Humphrey (Humphrey, 1989) says that trying to have stable requirements is a misconception: 'We must start with firm requirements' he remarks as an usual mistake. .

Deliver software to customers as fast as possible is referenced by Alan Davis (Davis A. , 1994); Mary and Tom Poppendieck (Poppendieck & Poppendieck, 2003) say that 'Rapid delivery is an operational practice that provides a strong competitive advantage' addressing principle 3.

Not fulfilling what is stated in principle 4 is mentioned by Steve McConnell (McConnell S. , 1996) as one of the project's classic mistakes.

Robert L. Glass (Glass, 2002) collects facts and fallacies of software engineering, one of the facts is a classic one: 'Requirements errors are the most expensive to fix when found during production but the cheapest to fix early in development' that is related to principles, 1, 3 and, 4. This topic is the main theoretical foundation on why the contention of defects needs to be performed on a given cycle avoiding them to cascade into the following.

Principle 5 is referred to and addressed by many authors, Boehm (Boehm, Improving Software Productivity, 1987) stated 'Management of people. The next most significant influence by far is that of the selection, motivation, and management of the people involved in the software process'. Steve McConnell (McConnell S. , 1996) referred to the lack of motivation as one of the project's classic mistakes. He says 'Undermined motivation. Study after study has shown that motivation ably has a larger effect on productivity and quality than any other factor' and refers to (Boehm, Improving Software Productivity, 1987). Tom DeMarco and Tim Lister (DeMarco & Lister, 1987) strongly state the importance of productive teams. Alistair Cockburn and Jim Highsmith (Cockburn & Highsmith, 2001) stress individual competence as a critical factor in project success and identifies the emphasis on people skills as a key factor underlying all Agile methodologies.

Regarding principle 6, Tom DeMarco and Tim Lister (DeMarco & Lister, 1987) addressed different problems in order to develop productive teams including communication. Luke Hohmann devoted a full chapter (Communication) (Hohmann, 1997) proposing a communication framework to get the best communication possible. Daniel Coleman (Coleman, 2015) stated that 'Interpersonal and group communication must travel multiple dimensions and optimal performance enabling the connection between two brains in the field of leadership goes through ways to improve emotional intelligence itself' and focuses on the way we communicate as a key issue to improve performance.

The meaning of what is a working software is fully covered in the traditional books of Software and Quality Engineering [(Sommerville, 2015), (Weinberg, 1992), (Fairley & Bourque, 2014), (McConnell S. , 1996), (Martin R. , 2012) among others]. Tom Gilb, (Gilb, 1988) developed an entire framework called 'Evolutionary Delivery' that includes several elements of the Agile Manifesto and the Scrum Framework. Some of the elements and definitions of the method are: 'Early, frequent iteration', 'Complete analysis, design, build and test at each step', 'Result orientation, not software development process orientation', 'On not knowing, and keeping it small and simple', covering principles 3, 7, 8 and 10.

In our understanding, the lack of quality and poor design leads to rework and thus a high Cost of Poor Quality (CoPQ), which disables the possibility to deliver value in a

fast manner and introduces wasted effort, is, therefore, one of the most counterproductive factors for team motivation (Ruiz de Mendarozqueta, Bustos, & Colla, 2019). Traditional books of Software and Quality Engineering (Sommerville, 2015), (Weinberg, 1992), (Fairley & Bourque, 2014), (McConnell S. , 1996), (Martin R. , 2012) among others, covered the topic and it is straightforward to see how the poor quality erodes the fast delivery of value.

'Requirements gold-plating and 'Developers gold-plating', are mentioned by Steve MacConnell (McConnell S. , 1996) as project classic mistakes; Mary and Tom Poppendieck (Poppendieck & Poppendieck, 2003) stated 'Eliminate Waste' as one of the fundamental principles explained as avoiding rework and not developing unnecessary functionality. All these references pointed out simplicity, the main component of principle 10.

Principle 11 is anchored to the definition of a system as a 'set of elements, dynamically related, that interact by exchanging information and energy to obtain a result providing information and energy' (Meadows, 2008); it is easy to apply the definition to the software. Systems theory states that the behavior of the system is determined by its structure (Meadows, 2008). The structure of the system is determined by the architecture and design (Sommerville, 2015), (Endres & Rombach, 2003), (Fairley & Bourque, 2014), (McConnell S. , Code Complete, 1993). The architecture is assumed to emerge. as the result of refining an initial proposal, or intentional result, with the feedback of the developers in each iteration, verifying the quality of the design and code.

The Scrum "embrace, inspect and adapt" (Institute) philosophy implements principle 12. This principle addresses the very well-known software engineering principle for continuous improvement (Humphrey, 1989), (Sommerville, 2015).

Relationship between Agility, Scrum and Software Engineering

In the previous section, we made a strong case that all basic agile premises are well established Software Engineering practices, which would lead to a reasonable conclusion that agile methodologies are a well-integrated corpus of practices that represents just another way to address requirements under the umbrella of the Software Engineering domain.

To further support our views, the authors selected a small sample of bibliography on agility, without any attempt to avoid any skewness but aiming to have a fair coverage of the bibliography and by no means exhaustive but often cited on academic efforts and as part of the daily professional exercise, and reviewing that small corpus sample with a focus on frameworks such as Scrum and XP. An immediate observation shows there is a noticeable scarcity of direct references for implementing software engineering practices. In the Table 1, we summarize a sample of a group of references and their relationship to software engineering practices and vice-versa.

Reference	References between agile and software engineering
(Shore & Warden, S., 2008)	Brief reference to software design
(Cohn, Succeding with Agile, 2010)	Brief reference to software design and code refactor
(Beck & Boehm, Agility through Discipline: a debate, 2003)	Referencing size of projects using XP
(Lan & Balasubramaniam, 2007)	No references
(SCRUMstudy, 2013)	No references
(Deemer, Benefield, Larman, & Vodde, 2012)	No references
(Schwaber & Sutherland, The Scrum Guide, 2017)	No references
(Boehm & Turner, Management Challenges to Implementing Agile Processes in Traditional Development Organizations, 2005)	Minor references
(Martin R. , 2012)	Code design and code quality in detail. No reference to agile methods nor Scrum.
(Sommerville, 2015)	Scrum and XP introduction but there is no relation with the other topics of software engineering
(O'Regan, 2017)	No references
(Schwaber, A CIO's Playbook for Adopting the Scrum Method of Achieving Software Agility, 2005)	It does not prescribe software engineering practices. Recommend to keep it simple and to let the team decides
(Duncan, 2019)	Minor references to design
(Poppendieck & Poppendieck, 2003)	Some general references to design approaches
(Cohn, Essential Scrum, 2012)	Minor references
(McConnell S. , More Effective Agile: A Roadmap for Software Leaders, 2019)	Minor references to code quality
(Martin R. , 2019)	A chapter with coding practices
(Stellman, 2014)	No references
(Fairley & Bourque, 2014)	Reference to Agile as a Method in Software Engineering Models and Methods chapter
(Johnson & Sims, 2012)	No references

Table 1 Software Engineering Bibliographical cross-reference

Software engineering bibliography, on the other hand, often considers agile methodologies as part of their body of knowledge. A lack of symmetry is observed as most of the available bibliography for agile methodologies avoid to reference their recommendation and practices as the actual implementation of different disciplines proposed by Software Engineering sources.

It is worth mentioning that, at the dawn of the agile methodologies (Cohen, Lindvall, & Costa, 2004), they emerged to overcome the drawbacks presented by the waterfall style lifecycle. From that perspective, agile practitioners saw little value in adopting well-defined processes which they perceived as rigid and value detractors while, at the same time, high maturity organizations working in compliance with SEI-CMMI™ based reference models identified that agile methods addressed most of the intermediate maturity requirements (Paulk, 2002). This trend seems to have been widespread as agile methodologies became mainstream since their inception.

A systematic bibliography review, presented in Table 2, shows that over a sample deemed relevant of 20 papers on agile topics; only 6 papers (30%) contain explicit references to Software Engineering principles and/or practices, 4 papers (20%) contain indirect references, and 10 papers (50%) contain no reference at all. This is taken as an

indicator that agile sources do a weak bridge between the concepts they describe which present correspondences with Software Engineering methods and principles.

Reference	Agile and Software Engineering
(Bustard, Wilikie, & Greer, 2013) (Hoda, Salleh, & Grundy, 2018) (Cohen, Lindvall, & Costa, 2004) (Kuhrmann, et al., 2019) (Ebert & Paasivaara, 2017) (Harvie & Agah, 2016)	Papers on Agile methodologies that contain explicit references to Software Engineering. In general, the agile process which considers SW Engineering practices are different SCRUM flavors, particularly when done at-scale. The emergence of hybrid development flavors (water-scrum-fall) is also observed.
(Vijayasarathy & Butler, 2016) (Mohan, Ramesh, & Sugumaran, 2010) (Falessi, et al., 2010) (Karlstrom, 2005)	Papers on Agile methodologies that contain indirect references to Software Engineering. In general the references appear in connection with SW architecture or overarching product management practices.
(Mantovani Fontana, Reinehr, & Malucelli, 2015) (Vallon, Strobl, Bernhart, Prikladnicki, & Grechenig, 2016) (Dingsøyr, Fægri, Dybå, Haugset, & Lindsjørn, 2016) (Chora, et al., 2020) (Bick, Spohrer, Hoda, Scheerer, & Heinzl, 2018) (Jorgensen M. , 2019) (Kersten, 2018) (Cockburn & Highsmith, 2001) (Akbar, 2019) (Telemaco, Oliveira, Alencar, & Cowan, 2020)	Papers on Agile methodologies that do not contain references to Software Engineering. It is observed that some of these papers discuss well-known development issues (e.g. coordination among teams, need of a maturity model for agile, requirements management, need of metrics to evaluate performance, etc.), without reverting to the well-established practice base provided by the SW Engineering to address them.

Table 2: Agile Methodologies Papers Bibliographical cross-reference

The very same factors that erode into the value on typical non-agile software development projects are observed on projects using agile methodologies; it is not difficult to observe that these factors are often not addressed as systemic problems, approach which hinders the capability to address them. Factors such as defect fallback from one cycle (sprint) to the next, rework effort, the increased effort devoted to addressing the technical debt on the product backlog and the need to rigorously validate & verify the developed components. are observed with enough frequency to be self-evident. In this sense, statistics from Chaos Standish Group (Liebert, 2019), shows that "agile project success rates are two times higher than success rates of waterfall projects. However, it also states that over 50% of evaluated projects have failed to meet all requirements of project constraints — time, budget and scope". Those figures reveal a poor performance record, even for the most successful software development methodology applied in the industry today.

Systemic modeling of the agile methodologies value

In his landmark book (Weinberg, 1992), Gerald Weinberg states that a systemic view and system modeling for software management and steering patterns is needed for coping with the traditional software development problems.

A previously developed line of work exploring the value of SCRUM (Colla P. , 2012) (Colla P. , 2016) followed by the exploration of typical software development issues and how they are expressed on typical agile projects (Ruiz de Mendarozqueta,

Bustos, & Colla, 2019) show that without great care to manage the main parameters of the software development cycle, an agile approach provides some extra protection of the project ultimate value, but at some point might end up eroding on that value. Software processes do not usually introduce restrictions to apply any given methodology of choice, only to deploy the controls to ensure no inviolate is overridden.

Simulation means seems to be the handiest tool to evaluate the relationship between depending variables of the system with their independent counterparts, as well as to explore the potential relationship and the degree of independence among variables. Any evaluation made based on simulation requires a fair estimation of the values assigned to different parameters and their assumed distributions; not much more than an advance to stronger quantitative methods based on field information.

The adoption of mature and well-proven as effective Software Engineering practices preserves the value of the project, by minimizing deviation from the business scenarios in terms of cost and calendar. This aims to achieve the overall balance of income and expenditure as well as optimizing other organizational and intangible factors typically factored into the opportunity cost used to discount cash flows, in this way the value can be measured by using the *Net Present Value* (NPV) of the project flows. The analysis tries to grasp the value for the organization from an investment standpoint, as it considers the cash flow and the risk to materialize it from a given a-priori point of view.

Simultaneously, the possibility to prioritize requirements over time, in a way that enhances almost continuously the value proposition of the organization, configures options, which can be valued using the *Real Option Valuation* methods (Brealey & Myers, 2016) (Mun, 2002).

The overall relationship among systemic variables can be expressed as a cause-effect model (Ruiz de Mendarozqueta, Bustos, & Colla, 2019) where the two main contributors to the overall value, the Net Present Value (NPV) and the Option Price Value (OPV) are established as dependent variables of several independent variables defined by the industry and organizational context as well as the decisions taken and results obtained during the project execution, being the sum of both values named the extended net present value of the project (eNPV) The resulting cause-effect model used represent independent variables defined by the organization outside the scope to manage from within the project, whilst other organizational factors are represented by some assumed distribution, and, finally, with intermediate variables with some systemic relation with the rest to express, understand, simulate, and extract conclusions from the systemic overall behavior into the dependent variables of interest.

From that approach, the main interest is to evaluate mainly factors that erode the total value of the project, which, in turn, is represented by the net present value defined by cash flows involved on it, plus the option values introduced by the agile methodology itself. The details of the analysis can be obtained in the referenced bibliography and will not be reproduced here due to of lack of space. But, as a summary, when projects with typical organizational values and intermediate variables distributions deemed as reasonable or supported by the bibliography are evaluated, some conclusions can be obtained as a further insight on the factors involved in the value erosion.
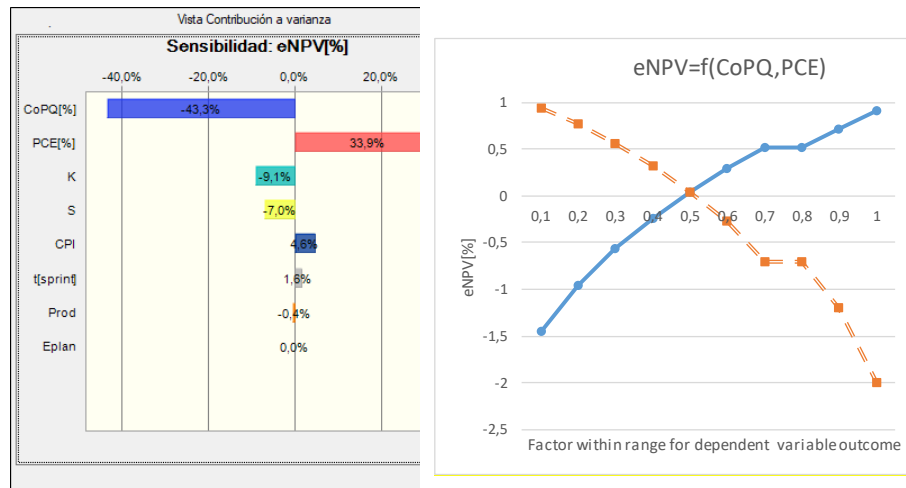
Figure 2 Sensitivity of total value with manageable factors and influence of main contributors ((Ruiz de Mendarozqueta, Bustos, & Colla, 2019)

From the identified contributors to the project extended net present value on agile projects, the most relevant is the CoPQ followed by some expression of the Phase Containment of Errors (PCE) which measures how much of the quality issues of one sprint is carried to the incoming as "*technical debt*". This effect can be rationalized considering the defects a value waste and the carry-over to be affected by a *cost increase factor* (K), as part of the value-added nature of activities on subsequent sprints and thus representing to the project net productivity hit if that happens. Agile methodologies do introduce additional sources of value, which creates buffers to manage deviations probably better than other methodologies; this can be seen as a qualitative confirmation on the reason why organizations prefer agile over other methods.

However, at the same time, a conclusion is that if no attention is paid to structural process variables, such as the ones traditionally watched by Software Engineering disciplines, eventually, the value is eroded to a point that, even with the added value of agile methodologies, the results turn against the organization. The conclusions of prior work suggest that CoPQ can be in the neighbor of 18% as the upper acceptable limit, and 80% as the lower limit for PCE for this effect to be noticeable. It comes as no surprise that these values are in the neighbor of those achieved by organizations in their early effort of applying structured methodologies traditionally recommended by traditional Software Engineering sources and matched values reported by the bibliography (Sandu & Salceanu, 2018) as obtained on successful typical agile projects; therefore, even minimal deviations might push the project beyond profitability, evidencing a link, somewhat hidden in the bibliography, between agile methodologies and Software Engineering practices. The results of the simulation, although preliminary, seem to be in line with some of the flow items of software value streams, namely defects and debt, identified by Kersten (Kersten, 2018).

Business perception of the agile methodologies value

A research effort has been carried out ((Ruiz de Mendarozqueta, Bustos, & Colla, Agile in practice, a systemic approach, 2019) aiming to understand how organizations perceive the relation between agile methodologies and traditional software engineering practices. As part of it, a field experiment was designed starting with a data-gathering among software organizations in Argentina. The analysis of the collected data throws some additional light on the subject. In these software organizations, activities are held mainly towards the development of standard products and customized implementations, update and maintenance of existing products, as well as embedded applications for electronic devices.

Research questions

Our research questions were:

- ¿Are the adoption of Agile methodologies and the embracement of software engineering practices perceived as related by the organizations?
- ¿How the adoption of agile methodologies and deployment of software engineering practices are related to the organizational size and age?
- ¿What is the influence on the adoption of agile methodologies and/or software engineering practices related to the markets the organizations participate in, the deployment of formal quality models evaluation and the operation under incentive programs? ¿In particular how both correlate to de Argentina´s software promotion law (Ley 25922)?

- **Data Source**

The scope of the collected survey attempts to include a group representing a variety of software organizations in Argentina. It is composed of few questions related to several organizational characteristics, context factors, and the usage of both agile methods and software engineering practices. A combination of Yes/No, Multichoice, and 5-Likert categorical values are captured through the questions. The survey went public thru different social media and professional network channels. Collecting enough answers to meet the confidence required by the design of the experiment is an ongoing activity. However, it is possible to preliminary explore, with a reduced number of answers, some initial results accepting a modest precision of the conclusions. Being a subject with little or no previous research efforts, some initial results bring some value in the authors´ perspective, and therefore they are shared in this paper. Further work will continue to collect enough data points to significantly improve the precision of the conclusions.

Analysis Framework

The organization size, measured as the direct software development resources, is mapped as a token of the organization's strength in terms of scale, at the moment to decide whether or not to perform investments on improving their performance. The organization age is used as a direct indicator for the room to collect feedback from customers, experience, and actual results, into the need to introduce structural compliance with software process methodologies.

Light needs to be thrown over some factors which subject to decisions being made by the management, whose relationship to the agile and/or software engineering practices would benefit from being evaluated. Among these factors, the actual core business and the nature of the markets served might define the need for the organization to raise the level of software development performance. Other parameters are the management decision to embrace formal quality-related evaluations and the affiliation to external programs that might be related to the fulfillment or adoption of industry frameworks. Pedro Colla (Colla & Montagna, 2008) referred a research made that predicts a significant relationship between the *organization size* (N) and the likelihood of embracing formal quality models. Intuitive as it might seem, this notion had received little attention in published papers in terms of validation.

Finally, the analysis includes as a factor assumed to operate as facilitator of the formal adoption of quality systems, the adherence to the Argentina's Software Law (agencia.mincyt.) (Ley 25922) which is incorporated also as a parameter whose relationship needs to be explored.

Design of experiment

Although a full census would be desirable to understand the full research scope, this is deemed impractical as a source of information about the factors addressed by this paper. Many organizations would refuse to go public with their internal data in fear of exposing competitive information of internal nature. Because of that, a sample survey has been attempted with a pre-defined level of representation of the target organizations which derives on a measurable confidence interval on the results. The sample could be considered, in broad terms and not completely void of skew factors, a random one as the call for answers was made public and no individual answers were solicited. After saying that, the affiliation and personal network of the authors play a role that might skew to some extent the results. However, the resulting dataset collected is deemed acceptable as it reaches the sampling error as preliminary acceptable at this stage of the experiment.

For the analysis's sake, generalizations would be made with the collected information assuming a random sample data has been collected and understanding the threat to validity this factor might introduce.

According to the data made available by CESSI (OPSSI, 2016) close to 650 organizations are involved in the software development business in Argentina, delivering to different segments and capabilities. This probably would be a very conservative number as many organizations might not be truly devoted to software development but other

activities of the value chain of the software industry, however, assuming a larger-than-needed number, would make the results stronger in terms of the confidence level.

In order to identify what would be the minimum sample size to achieve a given significance of the results is needed, this factor defines both the precision and the confidence interval of the results. It is a judgment call of the authors to balance the precision achieved with the realistic resources available to perform the data collection.

Cochran (Cochran, 1977) recommends a sample size (n_0) for a very large population:

$$n_0 = \frac{Z^2 \times p \times q}{e^2}$$

Ecuación 1

Where the normalized random variable (Z) represents the value at the confidence level assuming a normal distribution, using a value of 1.96 to achieve 95% confidence level. The assumed proportion of the population with a given attribute (p) and the lack of it (q) is assumed in the worst case by assigning the same value to both (0.5). At this point with over 30 valid and unique responses available the analysis is carried out with an accepted error level of 20%.

Yamane (Yamane, 1967) provides a criterion to define the sample size for small populations, when the sample size might be comparable to the total population or in any case, it cannot be considered as much larger, the result of the analysis yield similar conclusions in terms of the precision obtained with the available data points.

The overall assumed accepted error level might look a little high, but consideration needs to be given to the fact this research is aimed to obtain preliminary insights on a previously unexplored subject, and the authors consider this sort of precision a reasonable balance between the available resources and the robustness of the conclusions made possible.

Survey design

Two factors represent the dependent variables under study, the *degree of agile deployment* (AGILE, Y_1) and the *degree of software engineering practices deployment* (SWE, Y_2). Both are captured as categorical variables represented using a 5-Likert scale where the minimum level is little or no implementation and the maximum full adoption whereas the mid-scale represents the awareness and some fair level of usage. Both scales are designed to represent a similar depth of adoption per level.

Organizational characteristics are assigned as independent variables. Organizational size (X_1), Organizational age (X_2) are both assigned with 5-Likert categorical values. For the size, the CESSI (OPSSI, 2016) usual categorical scale is used, while for the organizational age an experimental sequence is adopted.

The main goal of the organization is based on development type performed, markets served, quality accreditations achieved and technology focus are also captured with multi-choice options that can be manipulated as different kinds of discrete answers with convenient grouping.

Survey Design and distribution

As design criteria, the total survey was created as a "*one-pager*" in order to increase the likelihood of being answered (Mardsen & Wright, 2010). A small operating definition is attached to each question and general instructions for fulfilling and returning are provided as well. A confidential statement ensures the participant that no individual answer will be used or published, and all the results would be statistical aggregates characterizing the sample in order to understand the whole population. Fulfillment helps are provided in terms of drop lists and checkboxes to uniform the answers provided within the defined categories. Google Forms (Ruiz de Mendarozqueta, Goggle Forms) has been used to implement the survey form and several validation and verification tests were performed by the authors to ensure the functionality of different options.

The survey was published on the *LinkedIn* account (Ruiz de Mendarozqueta, Linkedin) and other social media platforms for all the authors. A fair amount of bouncing from direct network professionals was observed allowing the survey to reach a larger audience resulting in the request to reach several hundred individual practitioners at the end of the diffusion process.

Survey Analysis

A total of 30 valid and unique responses were provided as collected by the Google Forms tool. The distribution of organizational size and age is given by Figure 3.

Figure 3 Organization size and age

The technology area where the organizations perform and the markets they serve is represented by Figure 4.
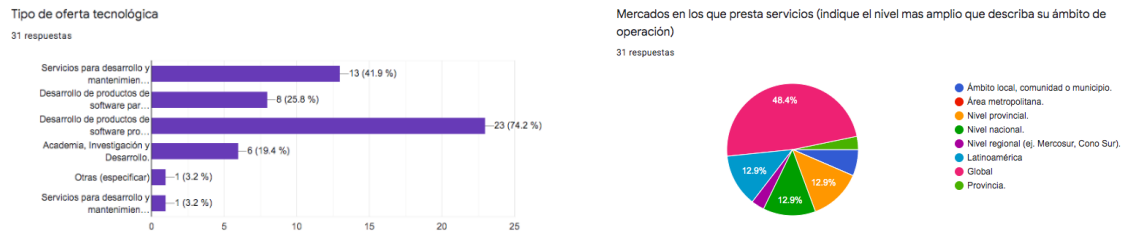
Figure 4 Organization technology area and markets served

The organization type and the formal quality system under which the organizations were evaluated are described by Figure 5.
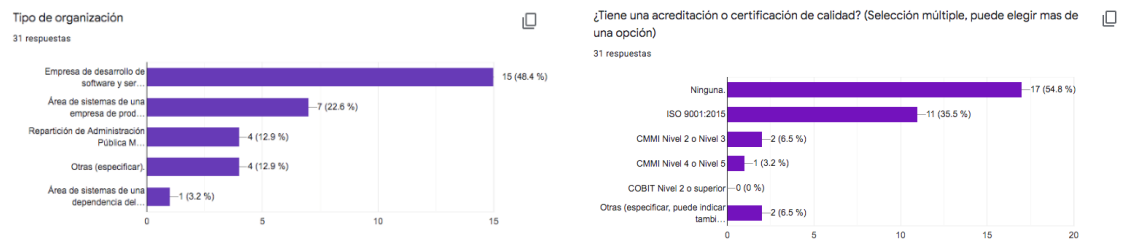

Figure 5 Organization type and formal quality system

As per the subject of interest for the survey, the agile adoption and the deployment of software engineering practices were found to be distributed as shown by Figure 6.
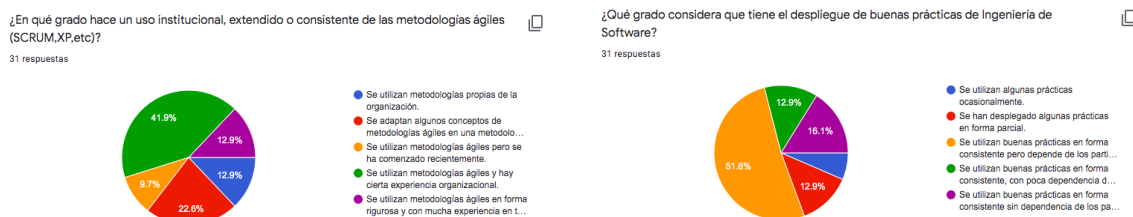

Figure 6 Organization adoption of agile methodologies and deployment of software engineering practices

Evaluation of dependent variables

The main tools for statistically analyzing a dataset differ depending on whether the distribution of the data follows a normal distribution or not. For non-normal distribu-

tions, "non-parametric" tools are used, which, in general, are less powerful and versatile. It is therefore an accepted practice, to use tools aimed at normal distributions even in cases where the distribution differs from it to a lesser extent.

The organization size is found not to follow a normal distribution since the Anderson-Darling normality test has a p-value=0.005. The organization age (AGE) normality test has also a p-value=0.005 and does not follow a normal distribution either.

Assumed both dependent variables represent equivalent levels of implementation for both agile practices and software engineering practices, the Mann-Whitney test compares the sample medians to be equal vs. not equal, resulting in a p=0,7958 therefore the null hypothesis cannot be rejected and both populations can be considered as having the same median value. The paired t-test needs to be used with caution because of the lack of normal distribution on both variables but it yields a T-Test of mean difference $= 0$ (vs $\neq 0$): P-Value $= 0,889$ therefore the null hypothesis of no mean difference cannot be rejected.

Using Ordinal Logistic Regression (Kruskal, 1954), an evaluation on the relation between the dependent variables with both organizational size (N) and organizational age (AGE) is made, a result of $p>0.05$ means there is insufficient evidence to claim the model does not fit the data adequately, and therefore the variables are related as seen in Table 3.

	Y (AGILE)	Y(SWE)
N	**0.435**	0.183
AGE	0.12	0.948
GLOBAL	**0.062**	0.244
SPI	0.604	0.007
EXT	0.104	0.322

Table 3 Ordinal Logistic Regression analysis (Goodman-Kruskal)

The impact of parameters such as the market being served (GLOBAL), the adoption of quality systems certification/assessments (SPI), and the operation under external program (EXT) is evaluated in terms of the dependency of the agile or software engineering practices adoption with them using a Chi-Square method (Table 4).

Source	p-value		Source	p-value
N	0.667		N	**0.060**
AGE	**0.032**		AGE	0.487
GLOBAL	0.473		GLOBAL	0.877
SPI	0.199		SPI	**0.079**
EXT	0.270		EXT	0.474

Table 4 Relation between parameters and dependent variables using the Chi-Square method

Using a Generalized Linear Model regression between the independent variables and parameters and the adoption of agile methodologies can be also seen in Table 4, where a p-value of less than 0.1 means a dependency was found, whilst a larger p-value indicates the independence (null hypothesis) cannot be rejected.

Repeating the analysis, but now with the implementation of software engineering practices, can be seen at Table 4 as well.

Finally, the discretized responses representing agile methodologies and software engineering practices adoptions are related to the adoption of Argentina´s software promotion law as a discrete (binary) variable. The Chi-Square analysis between SWE and LEY yield p=0.033 so a dependency has been found while the relation between AGILE and LEY yield p=0.783 and a dependency has not been found.

Discussion

The adoption of agile and software engineering methodologies are similar in organizations, the higher the one, the other correlates as higher too. This is a hint that organizations apply stricter agile methods as they are aware of the need to deploy software engineering practices as well. Organization size dominates the adoption of agile methodologies in a stronger way than the adoption of software engineering practices, whilst the opposite is suggested for software engineering practices.

The operation servicing global markets is related to the adoption of agile methodologies whilst the adoption of strict, committed or certified, quality frameworks is related to the adoption of software engineering practices, surprisingly the usage of external incentive programs seems to relate stronger with agile than the adoption of software engineering practices. The operation under the benefits of Argentina's Software Law drives the adoption of software engineering practices but it is not related to the usage of agile methodologies within the statistical margin assumed.

ISO 9001 as a quality framework for software development

Mentions were made through this article about the need for some organizations to have references to implement their internal practices, either because of the need to objectively guide software process improvement efforts in search of increase their competitiveness or because of competitive pressure to show some token of capabilities. The ISO 9001 standard is very well-known in the industry at large. According to the American Society for Quality (ASQ), organizations use the standard to demonstrate the ability to consistently provide products and services that meet customer and regulatory requirements. It is the most popular standard in the ISO 9000 series and the only standard in the series to which organizations can certify.'. The ISO 9001 is not industry-specific and can be applied to organizations of any size. Software companies, like any other industry, can use the standard, and there is a guideline (ISO 90003, 2018) that provides guidance for software organizations in the application of ISO 9001:2008.

In Argentina, the ISO 9001 standard is widely accepted by software development companies. In the 2019 annual survey among members of the CESSI chamber, over 60% of the respondents indicated as having an ISO 9001 certification. The main reason for this percentage is likely the incentives yield by the so-called Ley de Economía del Conocimiento (Ley 27506), which provides tax reductions, among other benefits, for those companies achieving a quality certification among other requisites.

Relationship between Agility, Scrum and ISO 9001 requisites

A case about the extension of Software Engineering principles to agile practices can be made by using a rigorous approach to measure it. Stålhane and Hanssen (Stålhane, 2008) outlined a map between the ISO 9001 requirements and agile practices, showing that a very good fit can be identified, and also performing a preliminary overview of items that need a further and deeper analysis for a full ISO 9001 compatibility. The experience paper (Ruiz de Mendarozqueta A. &., 2016) comments on two implementations of agile concepts and Scrum at two companies as their strategy for obtaining their ISO 9001 certificates; their approach was using a high-level map between ISO requirements to fulfill agility and SCRUM requirements.

Map between Agility, Scrum, and ISO 9001 requisites

The research question is to explore up to which extent agile principles and Scrum have strong and comprehensive coverage of ISO 9001 requisites. In order to achieve that, a high-level map between Agile principles and Scrum compared with the ISO 9001:2015 document sections will be performed. An actual ISO 9001 certification will require additional activities and proof of institutionalization mechanisms, therefore such a map is not a guarantee for getting an ISO 9001 certification nor is it an exhaustive mapping between possible relationships, it is meant to be a help to create a roadmap in that direction.

Agile Manifesto as a policy

The Agile Manifesto previously discussed is aiming at deconstructing the excessive rigid formalisms and focus on customer needs instead. Formal evaluation frameworks requires formal policy to be outlined which is sometimes regarded as a rigid view opposed to the degrees of freedom required to successfully deply agile practices. However, We believe it's quite the opposite. The Agile Manifesto can be considered as the policy for establishing agile. In particular, this paper focuses on Scrum as the agile methodology of choice given the widespread acceptance by Argentina's organizations as confirmed by previous research activities performed.

The Scrum approach institutionalizes activities using 5 key ceremonies which are:
- Backlog grooming (product backlog refinement)
- Sprint planning.
- Daily scrum.
- Sprint review.
- Sprint retrospective.

ISO requirements to Agile principles map

Follows an analysis between sections of the ISO 9001 framework where a stronger relationship with Agile and SCRUM can be found as compared with the Agile Manifesto. Only sections that refer to requirements for operationalization are considered, in that regard sections 0 to 4 are deemed as infrastructure needs of the standard and thus constitute a non-functional statute. The high-level mapping can be seen at Table 5.

Development Area	ISO9001:2015 Section	Agile principle
Leadership	5.1 Leadership and commitment	Management participation Leadership key to agile success
	5.1.2 Customer Focus	Agile principle 1 to 4
	5.2 Policy	Clear statement of policy (12 principles)
	5.3 Organizational roles, responsibilities, and authorities	Overall agile management approach
Planning	6.1 Actions to manage risks and opportunities	5 key ceremonies
	6.2 Quality goals	
	6.3 Change management planning	
Support	7.1 Resources	Agile principles 5 and 6
Operational planning and control	8.1 Operational planning and control	Scrum master cycle & ceremonies
Requirements for products and services	8.2.1 Customer communication	Agile principle 1 et al.
	8.2.2 Determining the requirements related to products and services	Product owner participation Scrum ceremonies
	8.2.3 Review of requirements	
	8.2.4 Changes for products and services	Backlog prioritization Agile principle 2
Design and development	8.3 Design and development of products and services	Agile principle 9 to 11
Performance	9.1 Measurement, analysis and evaluation	5 Scrum ceremonies, some additional activities might be required.
	9.2 Internal audit.	
	9.3 Upper management review.	
Continuous Improvement	10.2 Non-conformance correction.	5 Scrum ceremonies in particular daily scrum and retrospective ceremony
	10.3 Continuous improvement	
Products and Service Management	8.5 Production and service provision	Agile principle 5 Scrum lifecycle Sprint Ceremonies
	8.6 Release of products and services	
	8.7 Control of nonconforming	

Table 5 Table 5 1 High-level map between ISO9001:2015 and Scrum

ISO 9001:2015 Sections 0 to 4 are part of the framework infrastructure as non-functional requirements and thus there is no need for a mapping to be performed as are unrelated to the analysis being performed.

Leadership

In Lean philosophy (Poppendieck & Poppendieck, 2003), managers must apply the Lean principles and also must be the teachers who educate the staff in the principles of the Lean philosophy. In turn, they must go to the place where things are done to understand them, in the same place they happen. The Agile manifesto (Beck, et al., 2001) are the principles that the management must establish and promote in the company. Leadership is a key aspect in agile success (Cohn, Essential Scrum, 2012) (Cohn, Succeding with Agile, 2010) (Cockburn & Highsmith, 2001) (Cockburn A. , 2007).

Regarding policy, the agile approach prioritizes individual actions and their interactions over process and tools, leverage the software as documentation, cooperation, and close teamwork with the customer (represented by a product owner) above negotiation and, perhaps the most significant component, incorporate change into the methodology rather than opposing it following a pre-defined plan.

Finally, to implement customer focus the Agile Principles number 1 to number 4 (Beck, et al., 2001), stated a strong customer focus to avoid the common pitfalls in software projects, reduce risk and delivers value and high-quality software.

The organizational roles, responsibilities, and authorities are clearly defined in Scrum as roles and responsibilities for its execution and performance which satisfies this requirement (SCRUMstudy, 2013) (Cockburn A. , 2007) (Cohn, Succeding with Agile, 2010) among other references.

Planning

Planning is performed at a high level in the planning ceremony and daily during the Scrum meeting, this addresses key principles 1 thru 7 with enough evidence being collected to assess compliance with the planning activities, risk management, and change planning.

Support

Regarding resources, agile principles 5 and 6 have the same objective as the ISO 9001 7.1 section for providing infrastructure and an adequate environment for the people performing the activities.

Operational planning and control

This requirement from ISO 9001 is covered in the Agile paradigm (Cockburn, 2007), as shown in Figure 1 Agile conceptual modeling (Morse, 2012) because a fixed time window is established, a small team of developers is organized and functionality is

continuously evaluated, with the permanent help of the "owner" of the requirements providing the necessary sponsorship. The team planned which requirements are going to be delivered at the end of the time window as discussed in previous sections. Backlog Planning and Sprint Planning activities inside Scrum (SCRUMstudy, 2013) determine the planning activities.

Requirements for products and services

Customer communication requirements are addressed by almost all agile principles starting with number 1 as team communication, empowerment, and feedback is a key feature of agility and therefore a strong basis for customer communication. Those principles are instantiated in Scrum (SCRUMstudy, 2013) (Schwaber & Sutherland, Scrum.org, 2017) activities and roles such as Product Owner, Sprint Planning, Backlog Planning, and Sprint Review.

The requirements definition and review are performed during the different ceremonies in Scrum (Schwaber & Sutherland, Scrum.org, 2017) activities for requirements definitions, management and implementation are clearly stated with artifacts: Product Backlog, Sprint Backlog, roles: Product Owner and ceremonies: Sprint Planning, Backlog Planning and Sprint Review.

The critical management of requirement's changes are stated as part of the agile principle number 2 (Beck, et al., 2001), is addressing the changes of requirements utilizing Scrum (Schwaber & Sutherland, Scrum.org, 2017) activities for requirements definitions, management, and implementation such as artifacts: Product Backlog, Sprint Backlog, roles: Product Owner and ceremonies: Sprint Planning, Backlog Planning and Sprint Review.

Design and development of products and services

The critical technical activities involved with the design and development are addressed at a high level by agile principles number 9, 10, and 11.. Software architecture and design determine software behavior. The behavior must satisfy the customer's requirements including quality attributes. Agility aims to eliminate waste (Poppendieck & Poppendieck, 2003) considering waste things like partially done work, extra features, task switching, defects because they add no value. A good design minimizes defects so enhance value and lead to working software as stated by Agile Principle number 1

Product and service management

In Scrum (Schwaber & Sutherland, Scrum.org, 2017) there is an activity called Sprint Review where the Scrum Team and stakeholders collaborate about what was done in the Sprint. The done criteria are checked and the finished work is analyzed for potential release. If an item shows failures or is not conforming to what is expected by the stakeholders, the Product Backlog and Sprint Backlog are modified in order to cope with those nonconformities.

The Agile Principles number 5 is strong guidance for assuring that the product or service implements requirements, fulfills the done criteria and avoids nonconformity.

Performance

As part of the 5 key ceremonies (Schwaber & Sutherland, Scrum.org, 2017), in particular, the retrospective based activities opportunities to collect and analyze performance data is given to the team, learning from metrics collected, issues managed and key performance metrics are collected, in particular information related to the team capabilities expressed as the velocity to fulfill requirements.

However in order to fulfill the requirements expressed in section 9.2 (internal audit) which usually requires an independent view that isn't explicitly contained in the Scrum methodology and needs to be instantiated separately. In this sense, previous work (Gislén, 2016) shows that it can be achieved by having internal and external auditors participate in ceremonies, and by tailoring ISO terminology within the meetings, albeit not in the audit reports (e.g. Hit/Miss for ISO 9001 terminology Major/Minor Non-Conformity, Observation or Noteworthy efforts)

A similar situation happens with section 9.3 (upper management review) as the scrum defines the team as self-contained where the maximum authority is the product owner, different organizations might be structured around higher management layers than the product owner, it is implicit these levels are informed but in fact, this is not explicitly required by agile requirements and therefore needs to be instantiated to satisfy requirements from ISO9001. A reasonable means to achieve that is by having the product owner act as a proxy for the upper management, we believe that this role is well suited for the task because it has a clear notion regarding the degree up to which the customer requirements are satisfied as the product evolves sprint after sprint.

Continuous Improvement

As part of the 5 key ceremonies (Schwaber & Sutherland, Scrum.org, 2017), in particular, the retrospective-based activities opportunities to collect and analyze performance data is given to the team towards the fulfillment of the agile principle 12. This information can be used together with proper process improvement methodologies to perform adjustments in the team performance and delivery across different sprints.

Best practices and lessons learned

The results shown by the previous analysis at the conceptual, bibliographic and systemic dimensions, although preliminary, seem to be pretty consistent with the practical experience of the authors in real-world projects of different sizes and complexities where, more often than not, the projects where old fashioned, Software Engineering fundamentals are not enforced, the technical debt increases with the successive sprints eroding customer trust in the new features incrementally delivered, generating schedule overruns at a product level, and forcing to add extra effort, and hence cost, in the form

of additional sprints whose backlog is mainly composed of defect-correction stories. Very little is included in the agile methodologies corpus reinforcing the need to take special care of these technical aspects. This kind of situation is against some of the Agile principles, first and foremost the one that states that "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software". The value of the software is put then in question and could be destroyed if the project deviates from its goals beyond acceptable thresholds. More often than not, the actual investment the software project enables is highly leveraged with a much bigger investment return, and therefore, the entire investment is jeopardized. In addition to that, the effort consumed by sprints devoted to defect correction stories is essentially waste, contradicting, therefore, the Agile principle that states that "Simplicity – the art of maximizing the amount of work not done, is essential". The author´s experience shows that in order to fulfill at product level the Agile principle that "working software is the primary measure of progress", certain practices and metrics borrowed from the plan-driven software engineering processes, may be relevant to be exercised.

In terms of instruments, ways, and means to protect value, what the experience shows and the results of the simulation preliminary confirm is that, by large, the Cost of Poor Quality is the main driver in terms of value erosion all along the development cycle of actual software products, especially considering that a typical development cycle normally takes a significant number of sprints. This result is aligned with the classical principle that states that the cost of fixing a bug increases exponentially through the development process (Boehm & Basili, Software Defect Reduction Top 10 List, 2001). Attention needs to be paid to the importance of the capability to detect and correct errors in the sprint where they were introduced, which is measured by the PCE metric, as defects escaped from one sprint to the following ones, erode value with greater speed because of the value-added nature of the activities of subsequent sprints.

An immediate conclusion is the need to create a stronger awareness about the foundation nature of the Software Engineering practices, and the need to blend them in the day-to-day agile activities. Map how the different major goals correlate to agile activities needs to be done and understood by the team, metrics collection on subjects other than velocity and crump down related evolutions needs to be introduced as well. The authors believe that the definition of practices and collection of these metrics shall be as agile as the rest of the process, for example identifying the stories where defects from previous sprints need to be corrected and deriving PCE from them, and considering the story points of the backlog devoted to defect correction stories as a measure of CoPQ. In the same manner, as a burndown chart is kept and used as a measure of progress, curves of planned vs actuals of PCE and CoPQ could be kept and used as key elements for product release decisions and for appropriate planning of successive sprints.

The evaluation under a formal quality framework is shown as an additional dimension of the bond between the rigurosity required by some Software Engineering best practices with the main concepts beneath agile approaches to the point that formal evaluation methods can be integrated as a formal token of the rigorous implementation of certain requisites collectively deemed as a quality system; on the other hand, it helps the organization to embrace mature practices for the management of their business. Organizations, especially SME, are often willing to embrace agile concepts as a way to

improve their technical delivery capabilities and customer satisfaction, but the connection between agile practices aren't always linked with the actual execution of software engineering practices, and even when connected aren't perceived as good enough to aim for formal evaluations or certifications, We hope this work will help to laid down a step into deconstructing that notion.

Several bibliography references and the previous work from the authors build a conceptual roadmap on the relevance of agile methodologies as the conduit for the adoption of software engineering practices, the importance of protecting the value yield by the agility by using best practices, still, the survey of Argentinean organizations suggests that the preferred quality framework form SME sized organizations is still ISO9001. Little effort has been made to map how rigorous execution of agile methodologies can at the same time address the requirement of such framework and thus help not only to execute but also to assess in a structured way the operations.

The map developed shows how the different major requirements of ISO9001 correlates to agile activities that need to be done and understood by the team, metrics collection on subjects other than velocity and crump down related evolutions needs to be introduced as well. Two exceptions are identified as part of this map, one related to the need to introduce an independent view on the team operation (internal audit) and mechanisms for the upper management to be explicitly being kept informed. The authors believe that the definition of practices and collection of these additional activities shall be as agile as the rest of the process

Future work

Further work is needed to develop ideas toward a framework following the line of work of the I+D effort this paper is part of, including the identification of prototype projects where factual data can be extracted for further validation of the premises, as well as to collect metrics enabling the comparison of defect and phase containment behavior consistent with the ones captured from the bibliography. The results, in terms of product defects and development costs, could then be compared with those of similar projects that have not introduced these practices. Also, a further characterization of the emergent trend to apply hybrid approaches to software development in terms of mixtures between agile and Software Engineering process models is needed. Particularly for projects at some larger scale, where the importance of uncovering, understand and effectively applying the links between these two approaches will be increasingly important for practical purposes and, as such, a topic for further relevant research work. A great deal of confidence is placed on the completion of the research effort whose preliminary results are shared in this paper in order to obtain further degrees of confidence in the conclusions.

Bibliography (Stålhane, 2008)

(s.f.).

agencia.mincyt. (s.f.). *agencia.mincyt.* Recuperado el 17 de 10 de 2020, de Ley de Software: http://www.agencia.mincyt.gob.ar/upload/ley_25922-2.pdf

Agile Alliance. (12 de 08 de 2020). *Agile Practices Timeline*. Obtenido de Agile Practices: https://www.agilealliance.org/agile101/practices-timeline/

Akbar, R. (2019). Tailoring Agile-Based Software Development Processes. *IEEE Access, 2019.*

al, B. C. (2001). *Agile alliance.* Obtenido de https://www.agilealliance.org/agile101/the-agile-manifesto/

Alegrìa, J. H., & Bastarrica, M. (2007). *Implementing CMMI using combination of Agile methods.* V9(N1).

Appleton, B., Berczuk, S., & Cowham, R. (2005). *The Agile Difference for SCM*. Obtenido de CrsossRoads: https://www.cmcrossroads.com/article/agile-difference-scm

Banerjee, A., Narasimhan, B., & Kanakalata, C. (2011). *Experience of Executing Fixed Price Off-shored Agile Projects.* Proceedings of the 4th India Software Engineering Conference. ACM.

Beck, K., & Boehm, B. (2003). Agility through Discipline: a debate. *June 2003*.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Principles behind the Agile Manifesto*. Retrieved from http://agilemanifesto.org/principles.html

Bhasin, S. (2012). Quality Assurance in Agile –A study towards achieving excellence. págs. pp 64-67.

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. (2018). Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. *IEEE Transactions on Software Engineering, Year: 2018, Volume: 44, Issue: 10.*

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *The Journal of Political Economy, V81*(N3), págs. pp 637-654.

Boehm, B. (1987). *Improving Software Productivity.* IEEE Software.

Boehm, B., & Basili, V. R. (2001). Software Defect Reduction Top 10 List. *IEEE Computer, January 2001*.

Boehm, B., & Turner, R. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations.

Brealey, R., & Myers, S. (2016). *Principles of Corporate Finance 12th Edition.* McGraw-Hill, 6th Edition.

Bustard, D., Wilikie, G., & Greer, D. (2013). The Maturation of Agile Software Development Principles and Practice: Observations on Successive Industrial Studies in 2010 and 2012. *20th Annual IEEE International Conference and Workshops on the Engineering of Computer Based Systems (ECBS).*

Caballero, E., Calvo-Manzano, J., & Feliu, T. S. (2011). Introducing Scrum in a Very Small Enterprise: A Productivity and Quality Analysis. págs. pp. 215-224.

Chora, M., Springer, T., Kozik, R., López, L., Martínez-Fernandez, S., Ram, P., . . . Franch, X. (2020). Measuring and Improving Agile Processes in a Small-size SoftwareDevelopment Company. *IEEE Access (Volume: 8), 2020.*

Clark, B. (2000). Quantifying the effects of Process Improvement on Effort. *IEEE Software. Nov 2000*.

Cochran, W. G. (1977). *Sampling Techniques, 3rd Edition.* Wiley.

Cockburn, A. (2007). Agile Software Development. , Addison-Wesley.

Cockburn, A., & Highsmith, J. (2001). Agile Software Development: The People Factor. *IEEE Computer Year: 2001, Volume: 34*.

Cohen, D., Lindvall, M., & Costa, P. (2004). An Introduction to Agile Methods. *ADVANCES IN COMPUTERS, VOL. 62*.

Cohn, M. (2010). *Succeding with Agile.* Addison Wesley.

Cohn, M. (2012). *Essential Scrum.* Adisson Wesley.

Coleman, D. (2015). *El cerebro y la inteligencia emocional: Nuevos descubrimientos.* Penguin Random House Grupo Editorial España.

Colla, P. (2012). *Marco para evaluar el valor en metodología SCRUM.* La Plata-Argentina.: 13th Argentine Symposium on Software Engineering.

Colla, P. (2016). *Uso de opciones reales para evaluar la contribución de metodologías KANBAN en desarrollo de software.* Tres de febrero: SADIO ISSN: 2451-7593.

Colla, P., & Montagna, M. (2008). Software Process Improvement Behavior of Small and Medium Organizations in Argentina. *CACIC'08 .* Chilecito: CACIC'08 .

Davis, A. (1994). *FIFTEEN PRINCIPLES OF SOFTWARE ENGINEERING.* IEEE.

Davis, A., Bersoff, E., & Comer, E. (1988). *A Strategy for Comparing Alternative Software Development Life Cycle Models.* IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 14, NO. IO,.

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2012). *Scrum Primer.* Recuperado el 31 de 05 de 2020, de https://scrumprimer.org/

DeMarco, T., & Lister, T. (1987). *Peopleware.* Dorset House.

Dingsøyr, T., Fægri, T. E., Dybå, T., Haugset, B., & Lindsjørn, Y. (2016). Team Performance in Software Development Research Results versus Agile Principles. *IEEE Software (Volume: 33 , Issue: 4 , July-Aug. 2016)*.

Duncan, S. (2019). *Understanding Agile Values & Principles.* C4Media, InfoQ.com.

Ebert, C., & Paasivaara, M. (2017). Scaling Agile. *IEEE Software (Volume: 34 , Issue: 6 , November/December 2017)*.

Endres, A., & Rombach, D. (2003). *A Handbook of Software and Systems Engineering.* Pearson Addison Wesley.

Fairley, R., & Bourque, P. (2014). *SWEBOK v 3.0 Guide to the Software Engineering Body of Knowledge.* IEEE Computer Society.

Falessi, D., Cantone, G., Sarcia, S. A., Calavaro, G., Subiaco, P., & D'Amore, C. (2010). Peaceful Coexistence: Agile Developer Perspectives on Soft-ware Architecture. *IEEE Software Year: 2010, Volume: 27, Issue: 2*.

Fritzche, M., & P.Keil. (2007). Agile Methods and CMMI: Compatibility or Conflict ?

Gilb, T. (1988). *Principles of Software Engineering Management.* Addison-Wesley.

Gislén, M. (2016). *Achieving Agile Quality - An Action Research Study.* Karlskrona.

Glass, R. (2002). *Facts and Fallacies of Software Engineering.* Addison Wesley.

Glazer, H., Dalton, J., Anderson, D., Konrad, M., & Shrum, S. (2008). *CMMI or agile: why not embrace both!* SEI TECHNICAL NOTE.

Goldenson, D., A.Liu, & Jianping, Q. (2006). *CMMI-Based Process Improvement: How and When Does Success Happen?* CMMI Technology Conference: Software Engineering Institute.

Good, J. M. (2003). *A Pragmatic Approach to the Implementation of Agile Software Development Methodologies in Plan-Driven Organisations (MSc Thesis).* Lincoln University.

Hallowell, D.L. (2003). *Six Sigma Software Metrics Maturity*. Recuperado el 2019, de iSixSigma: https://www.isixsigma.com/industries/software-it/exploring-defect-containment-metrics-agile/

Harvie, D., & Agah, A. (2016). Targeted Scrum: Applying Mission Command to Agile Software Development. *IEEE Transactions on Software Engineering (Volume: 42 , Issue: 5 , May 1 2016).*

Hoda, R., Salleh, N., & Grundy, J. (2018). THE RISE AND EVOLUTION OF AGILE SOFTWARE DEVELOPMENT. *IEEE Software (Volume: 35 , Issue: 5 , September/October 2018).*

Hohmann, L. (1997). *Journey of the Software Professional* . Prentice Hall.

Hummel, O., & Burger, S. (2013). A pragmatic means of measuring the complexity of source code ensembles.

Humphrey, W. S. (1989). *Managing the Software Process.* Addison-Wesley.

Hung, M., & So, L. (2010). The Role of Uncertainty in Real Option Analysis.

Institute, S. (s.f.). *Scrum Institute*. Recuperado el 08 de 06 de 2020, de https://www.scrum-institute.org/inspect-and-adapt-scrum-framework.php

ISACA. (2018). COBIT 5 Framework. En ISACA.

Ismail, N. (2016). *UK wasting 37 billion a year on failed agile IT projects*. Obtenido de https://www.information-age.com/uk-wasting-37-billion-year-failed-agile-it-projects-123466089/

ISO. (08 de 06 de 2020). *ISO 9000:2015*. Obtenido de https://www.iso.org/obp/ui/es/#iso:std:iso:9000:ed-4:v1:es

Johnson, H., & Sims, C. (2012). *Scrum: a Breathtakingly Brief and Agile Introduction.* Dymaxicon.

Jorgensen, K. M. (2003). A review of software surveys on software effort estimation. *Proceedings ISESE 2003.* , (págs. pp-223-230). Rome, Italy.

Jorgensen, M. (2019). Relationships Between Project Size, Agile Practices, and Successful Software Development Results and Analysis. *IEEE Software Year: 2019 Volume: 36, Issue: 2.*

Karlstrom, R. (2005). Combining agile methods with stage-gate project management. *IEEE Software, Year: 2005, Volume: 22, Issue: 3.*

Kersten, M. (2018). What Flows through a Software Value Stream? *IEEE Software Year: 2018, Volume: 35, Issue: 4.*

Knox, S. (1993). Modeling the Cost of Software Quality. págs. pp 9-16.

Kruskal, L. A. (1954). Measures of Association for Cross Classifications. *Journal of the American Statistical Association*, 732-764.

Kuhrmann, M., Diebold, P., Münch, J., Tell, P., Trektere, K., McCaffery, F., . . . Prause, C. R. (2019). Hybrid Software Development Approaches in Practice: A

European Perspective. *IEEE Software (Volume: 36 , Issue: 4 , July-Aug. 2019)*.

Kunz, M., Dumke, R. R., & Zenker, N. (2008). Software Metrics for Agile Software Development. págs. pp. 673-678 .

Lan, C., & Balasubramaniam, R. (2007). Agile Software Development: Ad Hoc Practices or Sound Principles. *April 2007*.

Lawlis, P. K., M., F. R., & B., T. J. (1995). A Correlational Study of the CMM and Software Development Performance. págs. pp. 21-25.

Lee, G., & Xia, W. (2010). TOWARD AGILE: AN INTEGRATED ANALYSIS OF QUANTITATIVE AND QUALITATIVE FIELD DATA ON SOFTWARE DEVELOPMENT AGILITY. págs. pp 87-114.

Liebert, F. (2019). *BARRIERS TO SUCCESSFUL REALIZATION OF NEW PRODUCT DEVELOPMENT PROJECTS IN THE IT INDUSTRY.* Silesian University of Technology, Faculty of Organization and Management.

M. Staples, M. R. (2007). An exploratory study of why organizations do not adopt CMMI. *The Journal of Systems and Software 80* , p.p. 883–895.

Mahnic, V. (2012). A Capstone Course on Agile Software Development using SCRUM. *IEEE TRANSACTIONS ON EDUCATION, VOL. 55, NO. 1, FEBRUARY 2012*.

Maller, P., C.Ochoa, & Silva, J. (2004). Lightening the software production process in a CMM level 5 framework. *IEEE Latin American Transactions, V3(N1)*(pp 15-22).

Mantovani Fontana, R., Reinehr, S., & Malucelli, A. (2015). *Agile Compass: A Tool for Identifying Maturity in Agile Software Development Teams.* IEEE Software (Volume: 32 , Issue: 6 , Nov.-Dec. 2015).

Marcal, A., DeFreitas, B., Furtado, F., & Belchior, A. (2008). Blending SCRUM practices and CMMI Project Management Process Areas. *Innovation System Software*(pp 18-29).

Mardsen, P., & Wright, J. (2010). *Handbook of Survey Research.* Emerald.

Martin, R. (2012). *Código limpio: Manual de estilo para desarrollo ágil de software.* Anaya.

Martin, R. (2019). *Clean Agile: Back to Basics.* Prentice Hall.

Matson, J., Barrett, B., & Mellichamp, J. (1994). Software development cost estimation using function points. *20.4, pp. 275-287*.

McConnell, S. (1993). *Code Complete.* Microsoft Press.

McConnell, S. (1996). *Rapid Development.* Microsoft Press.

McConnell, S. (2019). *More Effective Agile: A Roadmap for Software Leaders.* Construx Press.

McMahon, P. (2010). *Integrating CMMI and Agile Development.* Addison-Wesley Professional;.

Meadows, D. (2008). *Thining in Systems: a primer.* Chelsea Green.

Mendarozqueta, A. R., & Andriano, N. (2014). Un enfoque para la mejora continua basado en los principios ágiles.

Miller, G. (2013). Agile problems, challenges, & failures. *PMI® Global Congress 2013*, págs. pp.1-8.

Mohan, K., Ramesh, B., & Sugumaran, V. (2010). Integrating Software Product Line Engineering and Agile Development. *IEEE Software (Volume: 27 , Issue: 3 , May-June 2010)*.

Morse, L. (2012). *3 Paradigm Shifts of Agile*. Recuperado el 04 de 05 2019, de Solutions IQ: https://www.solutionsiq.com/resource/blog-post/3-paradigm-shifts-of-agile/

Mukker, A., Mishra, A. K., & Singh, L. (2014). Enhancing Quality in Scrum Software Projects. págs. pp 682-688.

Mun, J. (2002). *Real Options Analysis, Tools and Techniques for Valuing Strategic Investment and Decisions*. Hoboken, New Jersey: John Wiley & Sons.

O'Regan, G. (2017). *Concise Guide to Software Engineering*. Springer.

OPSSI. (2016). *Reporte anual sobre el Sector de Software y Servicios Informáticos de la República Argentina*. Buenos Aires: CESSI.

Paulk, M. C. (2002). *Agile Methodologies and Process Discipline*. Institute for Software Research. Paper 3.

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit*. Addison Wesley.

Rafaela Mantovani Fontana, S. R. (s.f.).

Rico, D. F. (2008). What is the ROI of Agile vs. Traditional Methods? págs. pp. 9–18.

Ruiz de Mendarozqueta, A. &. (2016). Certificación ISO 9001:2008 en organizaciones ágiles. *CoNaIISI*. Salta: UCASAL.

Ruiz de Mendarozqueta, A. (s.f.). *Goggle Forms*. Recuperado el 10 de 17 de 2020, de https://docs.google.com/forms/d/1c0nhdf7hENrL9_r7mD0mUPQfOiIjOvD5 FdJ2mcU9WOM/edit

Ruiz de Mendarozqueta, A. (s.f.). *Linkedin*. Recuperado el 17 de 10 de 2020, de Linkedin: https://www.linkedin.com/posts/alvarordm_encuesta-sobre-agilidad-y-pr%C3%A1cticas-de-ingenier%C3%ADa-activity-6701246489215168512-0BgA/

Ruiz de Mendarozqueta, A., Bustos, F., & Colla, P. (2019). *Agile in practice, a systemic approach*. Paper accepted for 48 JAIIO-ASSE 2019, to be published in 49 JAIIO-ASSE 2020.

SAFe. (s.f.). Recuperado el 08 de 06 de 2020, de https://www.scaledagileframework.com/agile-architecture/

Sandu, I., & Salceanu, A. (2018). New approach to agile cycles containment effectiveness metrics in automotive software development. págs. pp. 3-8.

Sargent, R. (2009). Verification and validation of simulation models. *Proceedings of the 2009 Winter Simulation Conference, ed. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls,*.

Sauer, J. (2005). *Agile Practices in Offshore outsourcing- An analysis of published experiences*. ECSCW 2005.

Schwaber , K., & Sutherland, J. (2017). *Scrum.org*. Recuperado el 31 de 06 de 2020, de The home of Scrum: https://www.scrum.org/resources/scrum-guide

Schwaber, K. (2005). *A CIO's Playbook for Adopting the Scrum Method of Achieving Software Agility*. Srum Alliance.

Schwaber, K., & Sutherland, J. (2017). *The Scrum Guide*. Obtenido de Scrum.org

SCRUMstudy. (2013). *A Guide to the SCRUM BODY OF KNOWLEDGE.* Recuperado el 31 de 05 de 2020, de https://www.scrumstudy.com/: https://www.scrumstudy.com/

Shore, J., & Warden, S. (2008). *The Art of Agile Development.* O'Reilly.

Shuterland, J., Jakobsen, C., & K.Johnson. (2008). Scrum and CMMI L5 The magic potion for the code warriors. *V*(N).

Sommerville, I. (2015). *SOFTWARE ENGINEERING 10th Edition.* Pearson.

Stålhane, T. &. (2008). The application of ISO 9001 to agile software development. *The Norwegian University of Science and Technology. Springer Professional.*

Stellman, A. (2014). *Learning Agile: Understanding Scrum, XP, Lean, and Kanban.* O'Reilly.

Team, C. P. (2010). *CMMI for Development, version 1.3.* Pittsburgh, Pennsylvania, USA: Software Engineering Institute (SEI), November 2010.CMU/SEI-2010-TR-033.

Telemaco, U., Oliveira, T., Alencar, P., & Cowan, D. (2020). A Catalogue of Agile Smells for Agility Assessment. *IEEE Access, Year: 2020, Volume: 8.*

Turner, R., & Jain, A. (2002). Agile meets CMMI: Culture clash or common cause. *XP/Agile Universe LNCS 2418.*

Vallon, R., Strobl, S., Bernhart, M., Prikladnicki, R., & Grechenig, T. (2016). ADAPT A Framework for Agile Distributed Software Development. *IEEE Software (Volume: 33 , Issue: 6 , Nov.Dec. 2016).*

Vijay, D., & Ganapathy, G. (2014). Guidelines to minimize the cost of software quality in agile SCRUM process. *Vol.5, No.3*, págs. pp 61-69.

Vijayasarathy, L. R., & Butler, C. W. (2016). Choice of Software Development Methodologies – Do Project, Team and Organizational Characteristics Matter? *IEEE Software (Volume: 33 , Issue: 5 , Sept.-Oct. 2016).*

Vishal, S., & Kishen, I. (2007). *Will Agile Methodologies work in offshore outsourcing?* San Diego, USA: SWDSI07.

Weinberg, G. (1992). *Quality Software Management (Vol 1 Systems Thinking).* Dorset House.

Yamane, T. (1967). *Statistics, An Introductory.* NY: Harper and Row.