ΜΕΤΡΙΚΕΣ ΣΤΙΣ ΕΥΕΛΙΚΤΕΣ ΜΕΘΟΔΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ, ΜΙΑ ΕΚΤΕΤΑΜΕΝΗ ΕΡΕΥΝΑ ΠΕΔΙΟΥ

ΚΩΝΣΤΑΝΤΙΝΟΣ Σ. ΡΟΣΜΠΟΓΛΟΥ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ, ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΕΙΣΑΓΩΓΗ

Ο επιστημονικός τομέας της ανάπτυξης λογισμικού έχει υποστεί πολλές αλλαγές τα τελευταία πενήντα χρόνια οι οποίες είναι σχεδόν αδύνατο να κατηγοριοποιηθούν αλλαγές αυτές και η επίδρασή τους στην εξέλιξη του προαναφερθέντος τομέα. Παραδείγματα αποτελούν οι εισαγωγές νέων μεθόδων ανάπτυξης λογισμικού, τα νέα τεχνολογικά επιτεύγματα και η παλαίωση παλιών μεθοδολογιών, που συχνά λαμβάνει χώρα καθώς οι πολυάριθμες αλλαγές τις καθιστούν μερικά ή ολικά ανεπαρκείς.

Οι «παραδοσιακές» αυτές μέθοδοι, όπως ονομάζονται στους επιστημονικούς κύκλους, που περιλαμβάνουν μακροσκελή και αναλυτική τεκμηρίωση και χρήση προκαθορισμένων διαδικασιών ανάπτυξης λογισμικού έχουν χαρακτηριστεί ανελαστικές λόγω του ότι είναι συχνά ανίκανες να ανταπεξέλθουν στις αλλαγές που είναι σίγουρο ότι θα προκύψουν. Σαν μέτρο ενάντια στον κίνδυνο αυτό, οι παραδοσιακές μέθοδοι επενδύουν ένα πολύ μεγάλο κομμάτι του χρόνου ανάπτυξης λογισμικού στην ανάλυση και τον σχεδιασμό, προσπαθώντας να εικάσουν και να προλάβουν τις αλλαγές αυτές. Η πρακτική αυτή έχει χαρακτηριστεί μη παραγωγική, ένα γνώρισμα που έχει πλέον γίνει αναγκαίο κακό στις μεθόδους αυτές, κάτι που τόσο οι πελάτες όσο και οι προγραμματιστές έχουν μάθει να ανέχονται.

Όμως, μια ενδιαφέρουσα αλλαγή επήλθε στα δεδομένα της ανάπτυξης λογισμικού πριν από δέκα χρόνια περίπου, όταν εισήχθηκε μια νέα μεθοδολογία που ονομάστηκε Ευέλικτες Μέθοδοι (Agile Methods, Agile Methodology). Με την πρώτη τους παρουσίαση στο Agile Manifesto το 2001, η νέα αυτή μεθοδολογία ήρθε να απαντήσει στα υπάρχοντα προβλήματα και να προσφέρει προσαρμοστικότητα απέναντι στις αλλαγές, καθώς και βελτίωση της παραγωγικότητας, της ποιότητας και άλλων τομέων της διαδικασίας ανάπτυξης λογισμικού.

Λαμβάνοντας υπόψη ότι μια από τις κυριότερες αιτίες που εμφανίζονται τόσο συχνά αποτυχημένες εργασίες (projects) ανάπτυξης λογισμικού, είναι ότι υπάρχουν πολλά επικοινωνιακά προβλήματα αυξημένης σημαντικότητας, ώστε να δημιουργείται αστάθεια και αποτυχία μέσα σε μια ομάδα ανάπτυξης. Η ευέλικτη μεθοδολογία εστιάζει στην συλλογικότητα και όχι στον κάθε προγραμματιστή ατομικά, καθιστώντας έτσι την ομάδα σαν μια ενιαία οντότητα που μπορεί να υφίσταται και να λειτουργεί σε ένα περιβάλλον που περιλαμβάνει επιπρόσθετες τέτοιες όμοιες οντότητες που συνεργάζονται μεταξύ τους, προφέροντας στον κοινό στόχο.

Οι παραδοσιακές μέθοδοι αντιμετωπίζουν τις αλλαγές και την διόρθωση/επανεργασία (rework) σαν τα ποιο ακριβά τμήματα της ανάπτυξης λογισμικού. Προσπαθούν λοιπόν, όπως προαναφέρθηκε, να τις περιορίσουν, ή ακόμα και να τις αποτρέψουν μέσω ενδελεχούς αρχικού σχεδιασμού. Οι Ευέλικτη φιλοσοφία όμως, βλέπει την αποτυχία (failure) σαν το πιο δαπανηρό κομμάτι της διαδικασίας ανάπτυξης λογισμικού. Υποστηρίζει ότι η αλλαγές συμβαίνουν πάντα

και, σαν ένα αναπόσπαστο κομμάτι της διαδικασίας, θα πρέπει να διαχειριστούν και όχι να αποφευχθούν. Κρατώντας τον σχεδιασμό και την τεκμηρίωση σε πολύ χαμηλά επίπεδα, εστιάζει στο να παραδίδει λειτουργικό λογισμικό στον πελάτη όσο το δυνατόν γρηγορότερα, αναπτύσσοντας επιπλέον υποσυστήματα και επανεξετάζοντας τον κώδικα για να τα υποστηρίζει, στην πορεία. Αυτό γίνεται με τον καταμερισμό της εργασίας σε σύντομους επαναληπτικούς (iterative) κύκλους, διαπροσωπική επικοινωνία μεταξύ προγραμματιστών, συντονιστών, διοίκησης και πελατών. Με τον τρόπο αυτό ο πελάτης λαμβάνει ένα πρόγραμμα που ανταποκρίνεται στις απαιτήσεις του κι έτσι επιτυγχάνεται ο κύριος σκοπός μιας επιχείρησης, που είναι η ικανοποίηση των πελατών της.

Η προγραμματιστική πολυπλοκότητα, που συχνά αναφέρεται και ως πολυπλοκότητα λογισμικού (Software Complexity), είναι ένας όρος που περικλείει τις πολυάριθμες ιδιότητες ενός λογισμικού. Καθώς το πλήθος των οντοτήτων αυξάνεται ο αριθμός των σχέσεων και αλληλεπιδράσεων μεταξύ τους αυξάνεται εκθετικά και συχνά φτάνει σε σημεία που είναι αδύνατο να γνωρίζουμε και να μπορούμε να κατανοήσουμε όλο το φάσμα τους. Επιπροσθέτως όμως, αυξάνεται και η πιθανότητα να παρουσιαστούν ελαττώματα κατά τις αλλαγές. Τέτοιοι παράγοντες, στις προχωρημένες περιπτώσεις ανάπτυξης λογισμικού καθιστούν τον μετασχηματισμό πολύ δύσκολο ή ίσως και αδύνατο.

Για να αντιμετωπιστεί λοιπόν το παραπάνω ζήτημα εφαρμόζονται οι μετρικές λογισμικού (software metrics). Μετρική είναι μια διαδικασία μέτρησης ενός τμήματος του λογισμικού ή των επιμέρους χαρακτηριστικών του. Έχουν προταθεί πολλές μετρικές για πολλές περιπτώσεις λογισμικού και όχι μόνο και συχνά οργανώνονται σε κατηγορίες ανάλογα με το είδος των αποτελεσμάτων που προσφέρουν. Μια τέτοια κατηγοριοποίηση είναι το εξής παράδειγμα:

- *Πληροφοριακές Μετρικές*, οι οποίες παρέχουν πληροφορίες για τα διάφορα τμήματα της διαδικασίας ανάπτυξης λογισμικού, όπως ο αριθμός των προβλημάτων που έχουν λυθεί σε κάθε κύκλο.
- *Διαγνωστικές Μετρικές*, που υποδεικνύουν σημεία στα οποία χρειάζεται βελτίωση, όπως ο χρόνος στον οποίο λύνονται τα προβλήματα που προκύπτουν, κατά μέσο όρο
- *Παρακινητικές Μετρικές*, ο σκοπός των οπίων είναι να τονώσουν το ηθικό των προγραμματιστών αναφέροντας τα ποσοστά επιτυχίας, όπως το ποσοστό ολοκλήρωσης των εργασιών της ομάδας σε κάθε επαναληπτικό κύκλο.

Ο σκοπός της εργασίας αυτής είναι η διεξαγωγή μιας εκτεταμένης έρευνας πεδίου πάνω στο θέμα των μετρικών για τις ευέλικτες μεθόδους. Ακολουθώντας τα πρότυπα και τις ερευνητικές μεθόδους που έχουν τεθεί από σημαντικούς ερευνητές, στα πλαίσια της εργασίας αυτής εξετάστηκαν μελέτες περίπτωσης, έρευνες πεδίου και τυπικά/ελεγχόμενα πειράματα που δημοσιεύτηκαν έως το 2010. Τα δεδομένα που αντληθήκαν από την έρευνα επεξεργάστηκαν ώστε να προκύψουν χρήσιμες πληροφορίες και συμπεράσματα, τα οποία παρουσιάζονται σε ένα paper που είναι πιστό στις διεθνείς επιστημονικές προδιαγραφές και είναι συνταγμένο στην Αγγλική γλώσσα.

Empirical Studies on Metrics and Agile Development: A Systematic Literature Review

Panagiotis Sfetsos
Department of Informatics,
Alexander Technological Educational Institution,
Thessalonki, Greece
sfetsos@it.teithe.gr

Konstantinos S. Rosmpoglou
Department of Informatics,
Alexander Technological Educational Institution,
Thessalonki, Greece
kostasok@hotmail.com

Abstract

Agile software development is, since its birth almost ten years ago, evolving rapidly, undergoing changes, modifications and improvements. In order to keep track of the current state of agile software development, measurements and studies are required. Measurements provide qualitative and quantitative insights to the solidity of the process, enabling researchers, developers, managers and agile coaches to plan their actions more efficiently. A systematic literature review of empirical studies on quality and metrics of agile software development up to and including 2010 was concluded. The initial search identified 871 articles that had relevance to the subject, of which only 72 were taken under consideration. This study attempts to provide an adequate walkthrough of the research conducted, classify many of the metrics used in the industry and provide useful information for both current and future researchers and analysts.

1. INTRODUCTION

Since the beginning, the turbulent and unsure field of software development has been undergoing continuous evaluations, trial and testing in order to deliver software solutions that are cheaper, faster and of higher quality, thus providing increased customer satisfaction. Over the years, many ideas and suggestions have sought to bring the aforementioned development process and one of those waves of ideas was the Agile methodology [72, 74]. This movement, although treated with skepticism at first, proved to evolve and grow and today we can see that it has affected thousands of developers and had a huge impact on how software is developed worldwide [77, 73], by introducing evolutionary changes affecting positively software quality assurance, control and management, forming a disciplined process with built-in quality [75]. It is only natural for the adoptability and

overall utilization of agile methods to be predated by numerous studies, research and of course, metrics. Agile methods are quite different from traditional established methods and incorporate development procedures and practices that call for different metrics and measurements such as *frequency of product delivery* and *iterative development cycles* [76]. Developers use measurements to help them understand and manage their progress, while customers use measurements to help determine the quality and functionality of products. Quality Assurance teams, composed of testers and maintenance specialists, also use measurements to ensure quality, reliability and reusability of the final delivered software. Plenty of methods such as maturity models, measurement frameworks, goal-directed paradigms, process languages etc. have been proposed to support this idea [31].

Empirical studies are very common nowadays when applied to the investigation of evolution and changes in the field of software engineering. Individual studies however, cannot offer significant results due to their limited in scope or population, nature. A consequence of the growing number of empirical studies in software engineering is the need to adopt systematic approaches to assessing and aggregating all the research outcomes in order to provide a balanced and objective summary of research evidence for a particular topic.[78] This systematic literature review seeks to evaluate, synthesize, and present the empirical findings on agile software development regarding to metrics and quality and provide an overview of topics researched, their findings and implications for research and practice.

This paper is organized in seven sections. Section 2 features some results of previous relevant studies and describes the overall methodology utilized to create this review. Section 3 presents an overview of the studies and the individual results in an easily understandable format. Section 4 proceeds to discuss the impressions derived from each study in conjunction with the aforementioned findings. Section 5 offers the overall conclusion of our review. Section 6 offers the necessary acknowledgements and references, respectively, while Section 7 presents the research data synthesis table.

2. SYSTEMATIC LITERATURE REVIEW METHODOLOGY

2.1 Related work and context

Previous to this dissertation there have been several findings of related work, which were examined thoroughly and provided significant results, base information and inspiration for this study. Therefore, it is important that some of these past findings are provided here.

In 2003, a survey conducted by [47] reported that 93% of participants believed that correct measurement and application of agile methods offered their enterprise increased productivity, over 83% stated that quality and business satisfaction had increased significantly and 46% remarked that costs were

unchanged with implementation of agile. The next year, a survey transacted Thoughtworks researchers, brought to public view that over 40% of the individual persons' opinions valued quality and productivity related metrics, whereas near 65% of the organizations' points of view found metrics related to project and cost management very valuable and stated that there is initiative to improve such metrics within the respective companies. Furthermore by a survey enacted by [68] in 2006 with more than four thousand participants we have learned that although 54% of participants had little knowledge or familiarity with agile, around 60% of the total responders stated that productivity and quality had increased, along with 58% of their respective stakeholders satisfaction. Last but not least, the important annual surveys of VersionOne [43, 44, 45] in the years 2006 - 2008 with significantly more participants each following year (percentage of participants was incremented by +321% in the second year and a further +72% in the third), shows us that the most important barriers on the road to agile adoption are the unwillingness to change, usually because the companies' principles are contrary to agile practices and disciplines (near 44% of participants), and the lack of up-front planning (20% of responders, increased to 37% within two years) and loss of management control (37%) . Nevertheless, 84% stated that their companies had adopted agile methods in some or all parts of their software development process and in particular, 37% of those enterprises have embraced agile in over 75% of all their projects which can be viewed as a significant number. Nearly 40% are measuring Velocity and Testing metrics. Furthermore, Iteration Planning and Unit testing have increased by +17% in the course of one year and Burndown was popular in 60% of participants. Finally, over 60% of the organizations utilize Microsoft Project or Microsoft Excel as their preferred agile tool.

In order to assess and aggregate the research outcomes of the growing number of studies on measurement and metrics in agile methods and to provide a balanced and objective summary of research evidence, we applied the systematic literature review approach inspired by [78]. This review examines a variety of case studies, experiments and literature published up to 2010 and attempts to present and evaluate the empirical findings regarding metrics and quality in agile practices. The study was completing in steps, from inception to realization. These steps are: initial planning and protocol development, establishment of research questions, institution of the inclusion and exclusion criteria, organization of our sources, selection of primary studies, quality assessment, data extraction and synthesis.

2.2 Initial Protocol

When planning to perform our systematic literature review we wanted to adhere to the guidelines and procedures that are used and suggested by [79, 78, 77,

75] and include all of the aforementioned stages and steps. We believe that these practices offer a rigorous research framework and useful techniques.

2.3 Establishment of Research Questions

Answering the following research questions is the main concern of our effort.

- What is the state and the critical success factors in agile software development implementation?
- What is the state of the metrics used in the industry in terms of validity, utilization and producing results?

2.4 Institution of the Inclusion and Exclusion Criteria

In order to discover which of the primary studies were eligible in our review we took under consideration the following criteria:

- ✓ Studies must present empirical data on agile software development and must adhere to the requirements of our quality assessment procedure.
- ✓ Studies should be written in the English language.
- ✓ Research studies should be published up to 2009.
- ✓ Studies could be of professional developers and researchers as well as of students.

However we ought to establish some exclusion criteria so that our search would be more refined. Such criteria were:

- ✗ Studies did not present empirical data or their focus was not on agile software development.
- ✗ Studies did not have a research design and goal.
- ✗ Studies only presented the opinions of the researchers or provided only simulation data.

2.5 Sources Organization

The scope of the search was broad. We examined electronic databases and journals, as well as the proceedings of international conferences in agile methods and the web pages of some important agile practitioners. In order to provide a balanced and objective summary of research evidence for measurement and metrics

in agile methods we followed the systematic literature review. We focused on the journals and electronic databases of the ACM Digital library, IEEE Xplore and ScienceDirect - Elsevier, SpringerLink, the proceedings and newsletters of the international conferences in agile methods, such as XP and Agile Universe, and the web pages and blogs of some important practitioners in agile methods.

2.6 Selection of Primary and Secondary Studies

This process was performed in three distinct stages. In the first stage, we used simple but relevant keywords in order to search the articles that were included in the electronic databases and online proceedings. These keywords sought to match words in the titles, abstracts and key words sections of the online articles.

In specific, we used the following search terms:

A. Agile **AND** Software
B. Agile **AND** Development
C. Agile Practices **AND** Metrics
D. Agile **AND** Metrics **AND** Experiment
E. Agile **AND** Metrics **AND** Case Study
F. Agile **AND** Metrics **AND** Survey
G. Agile **AND** Empirical **AND** Study

The entirety of these search terms were also combined utilizing the “OR” Boolean operator in hopes that our results would include any articles that contained only one of the terms we sought. As [77] put very simply, we searched

A OR **B** OR **C** OR **D** OR **E** OR **F** OR **G**

All 871 articles that were retrieved were stored and sorted in a database with their matching search criteria in display. After some refining in order for the articles to assume a similar title, filename and date formats, they were inserted into Excel in order to make the following stage easier and modular.

In the second stage, both the authors worked together and reviewed all the studies derived from the previous stage in an effort to exclude the articles that were outside the research scope or had no relevance with agile methodology and practices whatsoever. Such excluded examples were articles that contained only abstracts, introductions and prefaces, interviews, discussions and comments, tutorials, presentations, advertising posters. A total of 725 articles and files were excluded.

In the third and last stage the authors collaborated and focused on the remaining 146 articles trying to flesh them out and find the ones that would be used to contribute in our review. Many articles were discovered to be of little to no relevance to agile software development despite the fact that their title said so. These files that were passed in the second stage were excluded here. Those that remained underwent our quality assessment procedure.

2.7 Quality Assessment

In order to better define our research we have set a number of questions which are to be used a guidelines for this review in order to refine our search on the existing literature and produce valid results. This way our goal can be clarified and defined properly. To screen through our 146 studies that passed through the net of stage two, we adapted a well used, proven and very effective procedure, in which the articles in study were assessed by the authors to see if they met requirements of rigor, credibility and relevance, which can be broken down to the following criteria:

1. Does the study in question answer to our review's questions?
2. Does the study present an empirical research and not summary or individual points of view?
3. Are the objectives of the research clearly stated and explained?
4. Is the context of the study explained in adequate detail?
5. Is the design of the study adequate to address the aims of the research?
6. Was the methodology followed to collect data described adequately?
7. Were the results affected by the relationship between researchers and participants?
8. Were the findings clearly expressed and provided?
9. Does the overall study contribute to future research?

Taken into consideration and cross-referenced, the above criteria ensured that the studies that passed quality assessment would make valuable contributions to our research. Of the total of 871 articles gathered in stage one, only 146 were forwarded to stage two for further examination and quality assessment, which were reduced to 72 valid studies which were taken under consideration for this review.

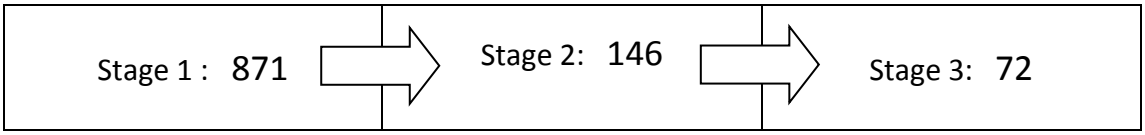

Figure 1. The number of studies deemed worthy of inclusion via the three stages.

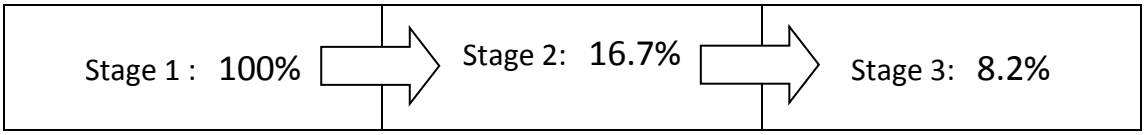

Figure 2. The percentage of studies that remained as they were diluted in each step.

2.8 Data Extraction

After we had narrowed down the scoped of our research, we reviewed each study individually and extracted information, which was entered in Excel for our convenience. Thus, a large table was formed in Excel where we stored every bit of information we found useful. For each study, the information gathered was:

1) **Authors**. Referring to the people who collaborated to bring their study to bear.

2) **Year**. Referring to the year the study was published and/or retrieved from the internet.

3) **Topic**. Referring to the exact Title of each study.

4) **Institution/Company**. Referring to the name and/or brand of the institution or company where the study has taken place, or failing that, the place of work of the authors, if the study was not…"distributed".

5) **Country**. Referring to the country of the company or institution.

6) **Type of study**. Referring to what type of study was examined. We have split the studies into three categories:
    - Experiments
    - Case Studies
    - Surveys

7) **Approach**. Referring to which approach of agile practices was utilized. Examples are eXtreme Programming (XP) and Scrum. If more than one methodologies were utilized we report it as Combined. If it is unclear as to the methods we left it as it was.

8) **Population**. Referring to the number of participants in experiments and surveys, as well as the number of case studies researched and mentioned in the studies.

9) **Remarks**. Referring to individual points of interest found and gathered from the studies, many of which were considered to produce our results.

The results of the above methodology can be viewed in appendix A.

3. RESULTS

Metrics are quantitative measures of performance or production used to indicate progress or achievement against strategic goals. In other words, a metric is a measurable element of a service, process or function. The real value of metrics is seen over time, as they evolve and change. Reliance on a single metric is not advised, especially if it has the potential to affect the behavior of developers and teams in an undesirable and counter-productive manner[50]. Therefore one must have collected knowledge of the different types of metrics and their qualities.

Following the examination of 72 studies several useful segments of information have been extracted which will be presented in this section.

3.1 Metrics Quality

As far as the quality of metrics is concerned, the familiar Iron triangle is used by many practitioners as an example of better understanding it. An individual metric's quality is affected by three factors. Time, Cost and Scope. In order to achieve the highest quality possible of a metric use the organization must first assess its Quality attribute by considering these factors. An example would be:

a. **Cost:** The amount of resources it will take to apply the metric in a project (are these resources available?).
b. **Time:** The time that measurement will take up (will it hamper the project's schedule?)
c. **Scope:** Does this metric contribute to the measurement result that the organization wishes to receive?

Figure 3. The Iron Triangle, showing the factors that affect quality in agile metrics.

However the above approach is theoretical and simple, lacking the required flexibility to be used by organizations in the industry mainly because from enterprise to enterprise the differences are many. Thus, as suggested mainly by [35, 37, 50], there have been established some criteria in order to assess the quality of a metric. By referencing to these criteria, each individual enterprise can collect data about their respective metrics and judge whether they want to use them or not.

An agile metric is considered of good quality if it:

I. Affirms and reinforces Agile principles.

Supports the customer-intimate traits and value focused traits that reinforce and strengthen Agile principles. This requires that people who understand Agile participate in metrics design. The truism "you get what you measure"[37] reminds us that behaviors which are counterproductive and contrary to the principles may appear if the wrong things like overtime and paperwork are enforced.

II. Measures outcome, not output.

In Agile practices, one of the principles promoted is to “reduce the overall mount of work not done” and the most impressive outcome might be achieved by reducing planned output while maximizing delivered value. Outcomes are measured in terms of delivered Customer *value*.

III. Follows trends, not numbers.

Measures "one level up" to ensure that aggregated information is measured and not sub-optimized parts of a whole. In addition, it Aggregates above the individual team level for upper management use.

IV. Belongs to a small set of metrics.

A "just enough" metrics approach is recommended: too much information can obscure important trends.

V. Is easy to collect.

For team-level diagnostics the most suitable is "one button" automation - where data is drawn from operational tools (i.e. the Product Backlog, acceptance test tools, code analyzers). For management use, avoid rework and manipulation of lower level data, aggregation is preferable.

VI. Reveals, rather than conceals, its context and significant variables.

Should be visibly accompanied by notes on significant influencing factors, to encourage improvement and discourage false assumptions.

VII. Provides fuel for meaningful conversation.

Face-to-face conversation is a very useful tool for agile process improvement. A measurement isolated from its context loses its meaning.

VIII. Provides feedback on a frequent and regular basis.

To amplify learning and accelerate process improvement, metrics should preferably be available at each iteration retrospective, and at key periodic management meetings.

IX. May measure Value (Product) or Process.

Depending on where problems lie, diagnostics may measure anything suspected of inhibiting effectiveness. Consider the appropriate audience for each metric, and document its context and assumptions to encourage proper use of its content.

X. Encourages "good-enough" quality.

The definition of what's "good enough" in a given context must come from that context's Business Customer, not the developers or management.

3.2 Metrics Classifications

There appear to be quite enough cases in the industry where the same metrics have been utilized in both traditional and agile methods, but often with a different name or under a different strategic category. Therefore, we collected all the metrics we surveyed and classified them into different categories adhering to the Metrics Educational Toolkit (METKIT 1993), which proposed a rigid framework of classification values, so that metrics will be easily categorized and retrieved. To that we have added a few more categories and fleshed out the existing ones in an effort to point out their respective attributes and incorporate new classifications.

PROCESS METRICS
▪ ***Maturity Metrics*** ➢ Organization metrics ➢ Resource, personnel and training metrics ➢ Technology management metrics ➢ Documented standards metrics ➢ Process metrics ➢ Data management and analysis metrics ▪ ***Management Metrics*** ➢ Project Management Metrics ➢ Quality Management Metrics ➢ Configuration Management Metrics ▪ ***Life Cycle Metrics*** ➢ Problem definition metrics ➢ Requirement analysis and specification metrics ➢ Design metrics ➢ Implementation metrics ➢ Maintenance metrics
PRODUCT METRICS
▪ ***Size Metrics***

- Number of elements
- Development metrics
- Size of components

- ***Architecture Metrics***
  - Components metrics
  - Architecture characteristics
  - Architecture standard metrics

- ***Structure Metrics***
  - Component characteristics
  - Structure characteristics
  - Psychological rules metrics

- ***Quality Metrics***
  - Functionality metrics
  - Reliability metrics
  - Usability metrics
  - Efficiency metrics
  - Maintainability metrics
  - Portability metrics

- ***Complexity Metrics***
  - Computational complexity metrics
  - Psychological complexity metrics

RESOURCES METRICS

- ***Personnel Metrics***
  - Programming experience metrics
  - Communication level metrics
  - Productivity metrics
  - Team structure metrics

- ***Software Metrics***
  - Performance metrics
  - Paradigm metrics
  - Replacement metrics

- ***Hardware Metrics***
  - Performance metrics
  - Reliability metrics
  - Availability metrics

Table 1. Agile metrics categorization according to METKIT

Another classification suggested by [50] of a collection of specific metrics is presented below

▪ ***Build*** ➢ Frequency of Builds ➢ Average Duration of Builds ➢ Number of Broken Builds per Iteration ➢ Number of Builds per Iteration ➢ Average Duration of Broken Build
▪ ***Tests*** ➢ Unit Tests per Story ➢ Functional tests per story ➢ Defects carried over per iteration ➢ Defects per story
▪ ***Development*** ➢ Cyclometric Complexity Measures ➢ Distribution of Method and Class Levels ➢ Rate of Change of Source ➢ Proportion of Source Code that is Test Code
▪ ***Scope*** ➢ Scope Change (stories removed or added from scope due to redundancy of rewrite per iteration) ➢ Scope Changes not caused by additional stories per iteration ➢ User Stories Carried Forward (Hangover) per Iteration ➢ Number of Stories held in Analysis, Development, testing per Iteration.

Table 2. Agile metrics classification

One additional classification of metrics suggested by [4] can be viewed in the following table, along with the appropriate tools to measure them.

Metrics	***Tools***
○ Business Metrics ❖ Running Tested Features ❖ Earned Business Value ❖ Net Present Value ❖ Internal Rate of Return ❖ Return of Investment	• Agile task management tool/plug-in • Issues management system • Microsoft Excel
○ Code Metrics ❖ Cyclomatic Complexity ❖ Best Practices Violation ❖ Coding Standards Violation ❖ Possible Bugs ❖ Code Duplication ❖ Code Coverage ❖ Dead Code ❖ Tests Quality	• Checkstyle • PMD/CPD • Jester • Findbugs • Simian • Maven website plug-in • Intellij IDEA Inspections
○ Design Metrics ▪ Code Dependencies ❖ Incoming (Affering Coupling) ❖ Outgoing (Efferent Coupling) ▪ Abstractness ❖ Number of Abstract Classes and Interfaces ❖ Number of Concrete Classes	• JDepend • Eclipse CAP plug-in
○ Process Metrics ❖ Agile Practice Maturity ❖ Impediments Cleared per Iteration ❖ Impediments Carried Over the Next Iteration ❖ User Stories Carried Over the Next Iteration ❖ Defects Carried Over the Next Iteration ❖ Team Member Loading ❖ Velocity ❖ Backlog Size	• Issues Management System • Special Agile tools/plugins • Physical Task Management Tools • Microsoft Excel

○ Automation Metrics ❖ Code Coverage ❖ Number of Builds per Day ❖ Time Taken per Build ❖ Number of Failes/Successful Builds ❖ Trends in Core Metrics	• Continuous Integration Tools ▪ CruiseControl ▪ TeamCity ▪ Bamboo ▪ Hudson ▪ Continuum • Cobertura • Clover • Maven Dashboard plug-in
○ Testing Metrics ❖ Acceptance Tests per Story ❖ Defects Count per Story ❖ Tests Time to Run ❖ Manual Tests per Story ❖ Automation Percent ❖ Time to Fix Tests	• FitNesse • Concordion • Selenium • Issues Management System • Testing Automation Tools

Table 3. Metrics and Tools by category

3.3 Other Metrics

In addition here we will present the remaining metrics that we have encountered [9, 10, 13, 14, 15, 16, 17, 21, 22, 24, 26, 28, 30, 52, 53, 56, 57] that are used by today's practitioners and are not included in any of the presented classifications.

Metric	***Description***
Sprint effort factor [26]	Sprint effort factor = (Items in current sprint/total feature list) +[∑ (change requests from previous sprints)].
Sprint complexity factor [26]	Sprint effort factor = ƒ (modules it interacts with # of interface points with other modules.
Change request effort [26]	Change request effort = ƒ (adding new features + changing previously defined features - deliberate elimination of features).
Customer expectation baseline [26]	Customer expectation baseline = (minimal set of expectation features from the sprint).
Impact on budget [26]	Impact on budget = ƒ (change request effort, customer expectation baseline.
Reusability Factor X [26]	Identifying reusable components in system = # of components added to library.
Reusability Factor Y [26]	Reuse of reusable components in system = # of

	components reused from library.
Facetime [26]	Facetime = ƒ (time each developer is with business person and with other developers on whom their work is dependant).
Budget at Complete [15, 16]	What is the targeted budget for the release? This can be expressed in either dollars or hours
Iteration Length [15, 16]	How long are each of your iterations or Sprints, assuming that planned iterations are of the same length?
Planned Iterations [15, 16]	How many iterations are planned to be included for this release?
Planned Release Story Points [15, 16]	How many Story points have you estimated to be included in the release?
Product Size [9]	Presents the amount of complete work
Pulse [9]	Measure how continuous the Integration is
Burn-Down [9, 21]	Shows the project's remaining work versus the remaining human resources
Faults [9]	Counts faults per iteration
Number of Process Improvements Requiring Organizational Support [10]	Counts improvement requests that passed through management to organizational support
Number of Process Improvements Not Requiring Organizational Support [10]	Counts improvement requests that did not demand support outside of the teams.
Total Lines of Test Code [13, 14, 17]	Counts the total number of test points in the system. One test point is defined as one step in an automatic acceptance testing scenario or as one non-blank, non-comment line of unit test code.
Source Lines of Code [13]	Counts the number of lines in the text of the program's source code. SLOC is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate productivity or effort once the software is produced.
Weighted Methods per Class [14, 17]	Measures the complexity of classes in an object-oriented system
Class Size [14]	Counts the total number of non-blank, non-comment lines of a class in the system
Number of Commits [14]	Counts the total number of individual commits to the source control repository.
Number of Lines Changed [14]	Counts the total number of lines (not only source code) added, removed, and updated in the source control repository.
Number of Delivered Stories [14]	Counts the total number of stories implemented in an iteration and approved by the customer.
TeamMorale [14]	Empirical way of assessing the team's morale state
Schedule Variance [22]	Indicates the deviation from the planned

	schedule for development
Budget Variance [22]	Indicates the deviation from the initial budget appointed to the software development
Cost of Defect Correction [22]	Measures the total extra cost derived from effort to fix and address to defects.
Effort Estimation Accuracy [10]	Measures the deviation of the total effort after production from the initial effort estimation.
Function Points [28, 52, 56, 57]	Express the amount of business functionality a software product provides to a user, introduced and used by the IFPUG Functional Size Measurement Method.
Time to market [30]	Measures the total time spent from a product's development until it reaches its destination.
New Product Sales [30]	Considered by enterprises as one of the most (if not *the* most) popular metrics.
Requirements Volatility [53]	Measures the number of changes (added or deleted) to the requirements of a project.
Resource Use [53]	Indicated the percentage of available resources utilized during the software development cycle.
Number of Defects [44, 56]	Measures the defects in number in the overall software development cycle, or per iteration.

Table 4. Additional Metrics

One more point of interest is the differences between traditional Earned Value Management and Agile EVM [15, 16, 41] as far as their terms are concerned, as presented by Sulaiman [16]. The terms in question are Performance Measurement Baseline (PMB), Schedule Baseline (SB, often integrated in PMB), Budget at Completion (BAC), Planned Percent Complete (PPC), Actual Percent Complete (APC).

	Traditional EVM	**AgileEVM**
PMB	The sum of all work package schedule estimates (duration and effort)	Total number of story points planned for a release
SB	The sum of all work packages for each time period calculated for the total duration	The total number of planned sprints multiplied by sprint length
BAC	The planned budget for the release	The planned budget for the release

	or project	
PPC	Percentage of completion expected to be achieved at a specific point in the project. Can be a subjective estimate, or a calculation of the dollar value of the cumulative tasks planned to be complete by this point in time divided by the performance baseline	The number of the current sprint divided by the total number of planned sprints
APC	The dollar value of work packages actually completed divided by total dollar value of the budget at complete	The total number of story points completed (potentially shippable increments) divided by the total number of story points planned

Table 5. Distinction between Traditional EVM and Agile EVM

3.4 Metrics and Diagnostics

In response to increased demand on correct and accurate metrics for agile projects, Hartmann and Dymond [35, 37] have introduced a different classification of metrics, the Diagnostics. Based upon the principle that too many metrics applied leads to unnecessary waste of time, cost and hampering of the teams' efforts, this strategic definition helps to understand which metrics should be used and where should they be applied to measure.

Except from the key metric(s), the goal that the organization will set, individual teams will need to support the aforementioned goal, by defining and performing their own localized measurements. Because these metrics support the main metrics by helping to diagnose and improve them, they are identified as *Diagnostics*. Usually, it is proposed by [35] that a **key** metric should be chosen, and all the remaining supplementing metrics should be regarded as means to improve and support this key metric. The key metric should be tied closely to the economics of investment and therefore the Business Value Delivered is recommended.

They are valid in the context of particular Processes and their inherent constraints (for example: a software development team). In order to avoid unnecessary measurements and maintain the focus to the key metric(s), it is strongly advised that diagnostics should be designed carefully and when they are applied to

measure, they should be done so with a set termination parameter, such as predetermined length of time of measurement or some conditions that allow their discontinuation. Perhaps the realization of this distinction between metrics and diagnostics will be difficult to implement by teams at first glance but once everyone has adopted the principle, the teams will be able to design and utilize god metrics that improve the agile processes. Always the diagnostics should be used in conjunction with metrics to produce valuable results, when operating in this context.

An easy way to distinguish diagnostics from metrics has been proposed by Hartmann and Dymond [37] and has the form of a question.

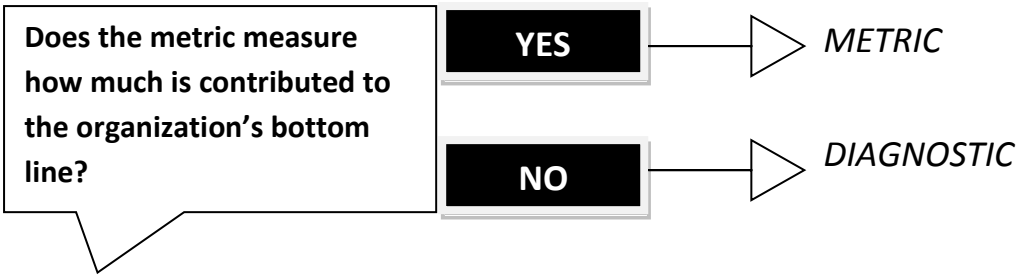

Figure 4. Distinction between Metrics and Diagnostics.

To wrap it all up, Metrics measure something that has direct value to the key metric, which as proposed in this case is Earned Business Value and diagnostics measure factors that are related to the ability of producing the above value. A list of important diagnostics is given below.

Diagnostics

- Agile Practice Maturity
- Obstacles Cleared per Iteration
- Team Member Loading
- Obstacles Carried Over Into Next Iteration
- User Stories Carried Over Into Next Iteration
- Iteration Mid Point Inspection
- Unit Tests Per User Story
- Functional Tests Per User Story
- Builds Per Iteration
- Defects Carried Over to the Next Iteration
- Velocity

Table 6. Some of the most important Diagnostics

3.5 Metrics Suites

Last but not least it is noteworthy to report the metrics suites that we encountered along with their respective metrics. The *ckjm* metrics suite utilizes object-oriented metrics and was studied by [13]. Chidamer and Kemerer's Metric Suite, Fernando Brito e Andreu's MOOD Metric Suite and Bansiya and Davis' QMOOD Metsics Suite were applied and studied in a case study by [1, 12].

Metrics Suite	***Metrics utilized***
ckjm	◊ Afferent Couplings (CA) ◊ CBO Coupling between Class Objects (CBO) ◊ Java specific CBO (CBOJDK) ◊ Depth of Inheritance Tree (DIT) ◊ Number of Children (NOC) ◊ Number of Public Methods (NPM) ◊ Lack of Cohesion in Methods (LCOM) ◊ Response for a Class (RFC) ◊ Weighted Methods per Class (WMC)
CK	◊ Weighted Methods per Class (WMC) ◊ Depth of Inheritance Tree (DIT) ◊ Coupling Between Objects (NOC) ◊ Response for a Class (RFC) ◊ Lack of Cohesion of Methods (LCOM)
MOOD	◊ Attribute Hiding Factor (AHF) ◊ Method Hiding Factor (MHF) ◊ Attribute Inheritance Factor (AIF) ◊ Method Inheritance Factor (MIF)
QMOOD	◊ Average Number of Ancestors (QMOOD_ANA) ◊ Cohesion Among Methods (QMOOD_CAM) ◊ Class Interface Size (QMOOD_CIS) ◊ Data Access Metric (QMOOD_DAM) ◊ Direct Class Coupling (QMOOD_DCC) ◊ Measure of Aggregation (QMOOD_MOA) ◊ Measure of Functional Abstraction (QMOOD_MFA) ◊ Number of Methods (QMOOD_NOM)

Table 7. Complete Metrics Suites

As we can see, since the model that Pressman suggested in 1994 has not only evolved, but even more models and frameworks have been suggested.

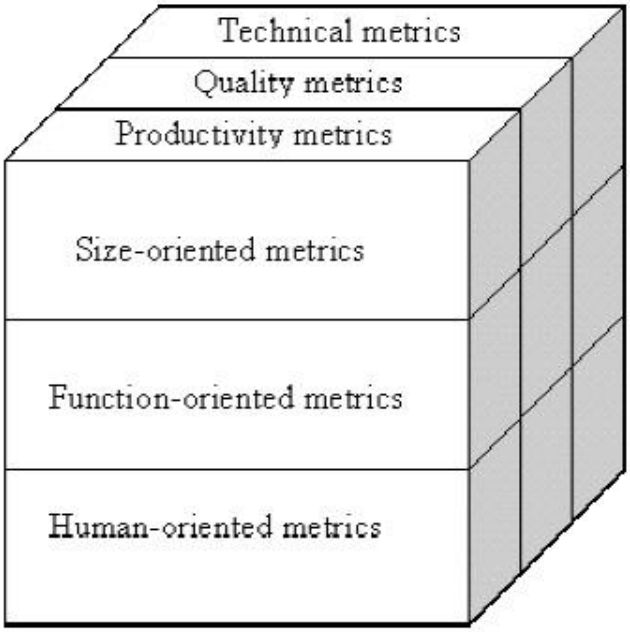

Figure 5. Pressman's Model, 16 years ago.

Some of the studies we reviewed suggested several metrics that were of reduced use and effectiveness due to various reasons. Many of them do not operate in alignment with agile principles; others are redundant or pointless and measured data for measurement's sake [37]. We will present them below.

Metric	**Disadvantage**
Checklist of Documents Completed [37]	Does not provide adequate information of working software produced.
Lines of Code, Total Lines of Code [1, 37]	Works against refactoring for quality design
Number of Tasks Completed [37]	Measures tasks, which can include things other than working software scope
Story Points per Person per Iteration [37]	Advocates competition instead of collaboration
Lines of Code per Developer [37]	Advocates competition instead of collaboration
Bugs Fixed [37]	Does not contribute to the improvement of the software development process.

Table 8. Ineffective and Useless metrics.

4. DISCUSSION

In order to develop real-time software benchmarking and estimation models, measurement specialists must work with and build upon the knowledge and result that has been achieved by others over the years and not start anew with their own set, or interpretation, of rules. In order to surpass this problem, the industry should agree on a set of measurement rules so that everyone is working with the same basis and each individual's results are valid and comparable data that can be utilized by others in the years to come. Considering the ever changing rhythms in which software engineering is evolving nowadays, there are bound to be differences in the practices and procedures applied that make the classification of some new metrics a very complicated process. The iterative development cycles, frequent delivery of product, the continuous requirements adjustments and changes, the test-driven development and pair programming, require different metrics selection and measurement methods. Taking all the above in to consideration the following classification of metrics can be suggested in an attempt to categorize the reviewed metrics in accordance with some core attributes that they share either referring to their utilization or to the nature and application of the results they offer.

Code Metrics
➥ Mean-Time to Repair ➥ Mean-time Before Failure ➥ Cyclomatic Complexity ➥ Class Size ➥ Coupling Between Class Objects ➥ Depth of Inheritance Tree ➥ Weighted Methods per Class ➥ Dead Code ➥ Code Duplication ➥ Coding Standards Violation ➥ Response for a Class ➥ Code Quality
Process Effectiveness Metrics
➥ Iteration length ➥ Planned Iterations ➥ Planned release story points ➥ Pulse ➥ Burn Down ➥ Faults ➥ Rework % ➥ Open and Closed Issues ➥ Changing Request Effort ➥ Defects Carried Over to Next Iteration ➥ User Stories Carried Over to Next Iteration ➥ Obstacles Carried Over to Next Iteration

➥ Obstacles Cleared per Iteration ➥ User Stories Completed per Iteration ➥ Defects Cleared per Iteration ➥ Number of Builds per Iteration ➥ Unit Tests per User Story
Test Efficiency Metrics
➥ Unit tests per Story ➥ Functional Tests per Story ➥ Test Code % ➥ Acceptance Tests per Story ➥ Tests Time to Run ➥ Manual Tests per Story ➥ Time to Fix Tests
Administration Efficiency Metrics
➥ Earned Value Management ➥ Number of Process Improvements Requiring Organizational Support ➥ Number of Process Improvements Not Requiring Organizational Support ➥ Number of Other Improvements Requiring Organizational Support ➥ Number of Process Improvements Not Requiring Organizational Support ➥ Schedule Variance ➥ Facetime
Team Efficiency Metrics
➥ Velocity ➥ Team Morale ➥ Number of Delivered Stories ➥ Team member Loading ➥ Surveys ➥ Interviews
Business Related Metrics
➥ Earned Business Value ➥ Budget at Complete ➥ Return of Investment ➥ Resource Use ➥ Budget Variance ➥ Potential Value Delivered ➥ Product Size ➥ Requirements Volatility ➥ Running Tested Features ➥ Impact of Budget

Table 9. New Metrics Classification

Of all the metrics that were encountered during the conduction of the study there was a small number of them that captivated the attention and left positive or negative impressions upon the researchers. A presentation of the above is offered below in detail.

Earned Business Value (EBV) was definitely a metric around which there is much speculation and discussion. EBV offers a percentage value for each item that the customer has received which represents the relative business value of the said item as defined by the customer. The actual *value* is defined by the customer according to their conception of delivered software importance. Since the organization's interest is to satisfy the customer through swift and frequent delivery of *valuable software,* this metric should be of great importance to any organization, for it is closely tied to the customer, showing customer contentment. However, in order for EBV to work and offer its substantial rewards the project's scope must be well known from the beginning of the procedure and often visible to the customer (who will assign his values to the individual parts to be completed. If such an endeavor is not possible then EBV should be avoided because if the scope deviates or numerous changes are added in the process at some point, the previous measurements will not be compatible with the new ones and this may lead to very unpleasant and false results. One last but important aspect of this metric is that it provides great inspiration and motivation for teams, because they can perceive that they are delivering something of *value* and are contributing towards the organization's goals. For a more complete picture [41] suggests that EBV is useful to agile projects especially when coupled with Earned Value Management (EVM). The Agile Earned Value Management metrics method's validity in accordance to Scrum projects is established both empirically and mathematically [16]. This allows The teams utilizing the Scrum agile method to be able to obtain accurate cost analysis and return of investment estimates and therefore steer the teams efforts more accordingly, in order to better achieve their goals.

Velocity. This very useful metric measures the amount of software that a development team can deliver per iteration. Usually counting story points, velocity can be a very useful metric if applied correctly. There is a thin line between proper and productive use of this metric and many of the organizations fail to discern it. Because velocity is closely tied to the particular team it observes, its results are useful within the team and almost never outside the team scope. It is primarily based on the team's own estimation of how much time they will spend working on said tasks and should not be used as a Time Estimation metric. It forecasts the team's ability to complete their work per iteration and should not be used as a goal to aim for or a comparison between teams. Velocity should be only applied once per team per project and its results should be questioned if changes are made to the team dynamics. Many agile practitioners couple Velocity with Burn Charts when in need to motivate their team. By combining Burn-Down and Burn-Up charts they can present the ratio of work completed / work remaining, which has proven to be an excellent motivator because team members take pride by seeing their velocity charts with high values and aim to retain them.

Facetime. Not considered as a metric by few, Facetime measures the amount that each developer spends with someone else, for example another team member, the team manager or a business representative or stakeholder. This metric, although not in direct numerical value to the organization, represents the main Agile Principle of personal interaction and collaboration. By tracking Facetime, each organization can see whether its software development process is actually following the agile methodology. It is obvious to say that this metric can be used only with co-located

teams and not with dispersed, since communication via electronic means is not an easily or valid measuring factor.

It was thought useful for reference to create a listing of the existing metrics collection tools that were reported earlier, so that referencing can be made easier.

• Agile task management tool/plug-in • Issues management system • Microsoft Excel • Checkstyle • PMD/CPD • Jester • Findbugs • Simian • Maven website plug-in • Intellij IDEA Inspections • JDepend • Eclipse CAP plug-in • Issues Management System • Special Agile tools/plugins • Physical Task Management Tools	• CruiseControl • TeamCity • Bamboo • Hudson • Continuum • Cobertura • Clover • Maven Dashboard plug-in • FitNesse • Concordion • Selenium • Issues Management System Testing Automation Tools • UnitMetrics

Table 10. Measurement tools listing

The UnitMetrics measurement tool which was developed and implemented into the integrated development environment (IDE) Eclipse, has made the aforementioned able to support agile development. With over 100 downloads and utilizations, initial assumptions can be made as to how to better support agile software development and, in particular, it's possible refactoring. [31]

Past analysis showed that it is actually possible to measure both the quality of a product and the stability of a maintenance process with the use of risk, reliability and test metrics. Since then, many methodologies and ways of applying such metrics have been developed and in our times, almost ten years later, we have the luxury of being able to choose the one best suited for our projects without having to modify (or with minimal modification) factors. Thus, many organizations proceed to obtain a metrics suite instead of deciding first hand which metrics should be used and for what reason. It is natural to observe a tendency towards solid, organized and ready to use metrics, even if that implies the loss of some valuable information that the various packages might not include.

The CMM and CMMI methods typically do utilize formal design and code inspections, which are more than 65% efficient in finding bugs or defects, a value close to twice the efficiency of most forms of testing, which on average can identify

only 30% of the errors present. In 2007 research has showed that the average number of cumulative defect removal efficiency is only about 85% in the US, so both the Agile and CMM ratings are better. The CMM method sports a little higher value than Agile, mainly due to formal inspections, testing specialists, and a more formal quality assurance approach, which agile does not (appears not to) have. The CMM and CMMI are cumbersome when applied to small projects for because they were designed to be used in applications of larger size and scope. They may be tailored or subset to fit small projects, but primarily, they are not effective [11].

The case study conducted by [12] indicate that the CK and QMOOD Object-Oriented (OO) class metrics suites are useful in developing quality classification models to predict defects in agile software development processes for both initial delivery and for multiple, sequential releases. The UBLR analysis for the MOOD metrics indicated that in general, they were not useful as a means to predict the error proneness of object-oriented classes. From the three metrics CM, MOOD and QMOOD, the fist one has proven to be more effective and liable. However, it is also noteworthy that these metrics will not continue to be as reliable throughout time, as the software itself matures over continuous iterations.

Analysis shows that there is not a single methodology that is adequate enough to be universally utilized so that it can bring the required results, guaranteed. There are lots of variables that need to be taken under consideration before deciding which methodology to use, such as project size, time limits and team size and skill [12].

Another interesting fact is the conclusion that application of Extreme Programming to a team of developers for the army [9] showed that despite the fact that the army is a solid and hard to change organization with specific procedures and ways of operation, it is possible to introduce successfully a new methodology which, as it turned out, increased the unit's confidence and ability to make short and long term decisions as well as the management's disposition regarding agile methods. It is surely difficult to apply new methodologies to unwilling recipients as many aforementioned surveys have indicated but if it can be applied in the army, then it can be applied almost anywhere.

In the experiment of [13], cross-referencing the results of Object Oriented and Secure Line of Code metrics when they are both applied to examine the same project we find that with the knowledge of the actual development cost of a project or component, we can view an accurate estimate of the development cost that would occur if we had given a different set of imports and therefore be able to consider if the new parameters are hampering the development process as far as cost is concerned.

According to [43], configuration management and version control metrics applied to an agile environment raise two main concerns: Firstly, agile methodology's approach with very small feedback loops which occur frequently makes for a lot of complex details to attempt to measure in order to gain valid results and secondly, favoring "people and interactions" over "process and tools" makes it very hard to acquire successfully such metrics transparently and unobtrusively. This is doubly true if the organization does not know which metrics to

use and just measures everything in hopes of getting positive results, or when the projects are constantly changing scope and have uncertain goals.

By combining a metric and decision trees one can successfully locate and predict modules with significantly large numbers of undetected faults. Identifying these modules and devoting some additional effort to maintain, enhance and test them out, leads to greater quality and reliability of the software in whole [23].

The Application Development (AD) Metrics are smartly introduced with the Balanced Scorecard model [33], which organizes them into four categories and gives the opportunity for the clients to choose which metrics to use form across them. This way, although appearing costly in the business level, the metrics work very well in the application development level and offer a very realistic and to the point model for metrics selection and definition. In addition it is emphasized that data collection should be as unobtrusive as possible because it is relatively easy for developers to "fake" data in order to get their job done easier and circumvent the process which does not affect them per se.

Many Agile practices such as pair programming, test driven development and planning, offer good techniques and mechanisms to improve agile software development as well as to implement and further test activities of the ISO 12207 development process [19]. Although in almost all cases a significant difference in quality metrics of software developed in the various phases and systematic improvement of software quality metrics, occurred when agile practices are thoroughly used by skilled developers [1] as higher quality and more efficient development led to a reduction in overall project duration, defects and rework. This resulted in reduced costs to build, change and support a new development and production platform [6]. The tracking techniques of agile methodology are not necessary to projects with short-release cycles whereas they are valuable to projects with long iterations. Nevertheless they should be taken under consideration in every project where the control and management shifts from one person to a whole team. Even then however, there is need for a Project Manager, a person who should be able to plan correctly often in the middle of the project's cycle, guide and lead the team to the desired result.

Analysis of environmental programs indicates that there is a very important connection between environmental performance and profitable returns that needs to be further examined and calls for the development of more sophisticated metrics [34]. By using metrics to measure and factor this connection is the only way to show that environmental programs are not a necessary burden. They are not decreases in profitability that must be tolerated for the greater good. They should be viewed as (sometimes) radical innovations that add to profit instead of decreasing it by reducing and minimizing the costs and planning more efficient use of resources.

Process-Oriented Metrics for Software Architecture Adaptability (POMSAA) is a process-oriented framework which calculates the necessary metrics for Adaptability by tracing metrics to their respective requirements, analyzing the reasons for strategic strengths and weaknesses in metrics, evaluates and suggests improvements on the architecture of the framework. These observations are based in an initial study and application of the aforementioned framework which suggests that a lot of research and work is in due in order to certify the operational validity and reliability of POMSAA.

Another metric that should be briefly discussed is the equally loved and frowned upon, Function Points. Throughout the study we have encountered this particular metric in many a case and we saw that it is not one of the metrics that are used *en large* or *at will* by organizations and analysts. However here are some noteworthy observations. The Function Point metric when applied should not only focus on the end product of the measurement system but in each step of the process, which most likely will provide new information. It can also identify which intermediate step is more meaningful and important by measuring the information lost (instead of added) in each step [57]. It is also possible (and effective) to measure Function Points from a system which is expressed in entity relationships(ER) and data flow diagrams (DFD). This automatic measuring process saves lot of human work hours and results are in accord with the traditional human counters, while the IFPUG counting rules are made more solid and rigorous, which helps avoiding confusion between the different counters of the same project [56].

Admittedly, we should not exclude the Naked Objects framework. Naked Objects allow the fast realization of user-stories, which allows better understanding of the demands and changes need to be done. Its experimental implementation [66, 67] has offered some interesting data. Development duration was significantly reduced by -50% while the total effort in hours was reduced by -64%. The total size of the software's code was decreased by -39% and the team productivity substantially increased by 54%. Although these are impressive results, naked objects is still an untested framework and further work is required to assess its overall benefits. Being not yet mature, it should not be considered for application with multiuser security requirements and lots of objects. Additionally, implementation requires high throughput and has been considered difficult.

While agile development is by no means an adequate solution for all projects, findings of [4] indicate that there are a number of different cases where agile development can result in a significant advantage in terms of driving efficiency into the application development process and increasing the time to benefits by reducing errors and changes brought about by unforeseen changes in business requirements.

The threat of mendacity.

The danger of error in the extracted data and deviation in the synthesis of the results is ever present and lurking. In order to uphold the existing research standards and to strengthen our research's framework against this danger, we have developed the research protocol detailed in section two, based upon the guidelines of [77, 78. 79]. Our effort was focused on the goal to set the correct questions that the studies answer to and the various criteria through which the literature we reviewed was filtered. Nevertheless it must be noted that although both authors went through this demanding and arduous procedure, room for data inaccuracy might still exist. This threat to the review's internal or external validity might occur due to some elements such as the inability to accurately and properly compare studies that have notable differences in their respective attributes, or the lack of parts of the attributes of studies what were deemed too important to be discarded.

5. CONCLUSIONS

Metrics are globally utilized in order to help planning the organizations' market strategies, including product development and release. They are effective in estimating and assessing new projects, also identifying and mitigating risks. They offer regular and frequent feedback, and if applied correctly do not pose threats to team collaboration and unity.

When applying metrics it is best to be as clear as possible and consider the transparency degree, if any, and its possible implications on the developers. Metrics should be used as basis for discussion and not to compare developers or teams with each other. The teams may frown upon the usage of metrics and feel monitored and controlled, which could lead to gradual decrease in morale. Therefore it is suggested that team state should be strongly considered along with careful metrics selection that are easy to measure and will not hamper the development process. As [4] suggests. The object of the measurements should be preferred to be outcomes over outputs and results over activity. Last but not least, it is counter-productive to follow numbers instead of trends. Measuring in agile methodology is not a means of corporate control but a useful tool that helps understand the progress of our projects, compare it with other measurements and provide significant information, which can be used to inspire, to motivate and usher a way of acting and thinking that is in accord not only with the organizations' goals but with each individual team's as well.

The cornerstone of successful and beneficial bonding of agile methods within an organization is the communication and collaboration between the organizational level and the project-developing teams, in a constant and uninterrupted basis. Many times, and often while the developing process is still underway, the organization needs to make appropriate decisions and take action to address the problems and/or mistakes that come up, and set the correct course that the developer's actions should then take. It is suggested that the implementation of a project Facilitator in each team would ensure that the team can successfully overcome any obstacles they might find by always performing in tune with the organization's plans. Be that as it may, team and project size are always relevant to whether agile methods can be used.

This systematic literature review offers two main contributions. The enacted review and its presented summary on empirical studies considering agile metrics, can be of importance and use in the industry to provide solid information on quality and utilization of metrics, as well as their respective categories and measurement tools. In addition to the current state of agile implementation and impact on software development, it is hoped that the findings of this review will be useful to future practitioners and researchers who want to stay in tune with and build upon the development of agile procedures and methodologies.

6. REFERENCES

[1] Giulio Concas, Marco Di Francesco, Michele Marchesi, Roberta Quaresima, Sandro Pinna: An Agile Development Process and Its Assessment Using Quantitative Object-Oriented Metrics. XP 2008: 83-93

[2] John Erickson, The Total Economic Impact of ThoughtWorks' Agile Development Methodology. Single Company Analysis – Energy Services, Forrester Consulting, securerespond.com/tworks/pettit/forrester.pdf, etrieved 2009

[3] John Erickson, The Total Economic Impact™ Of Using ThoughtWorks' Agile Approach Single Company Analysis — An Australian Insurance Provider, Forrester Consulting, http://www.thoughtworks.com/sites/www.thoughtworks.com/files/files/forrester_tei_au.pdf, Retrieved 2008

[4] Forrester researchers, The Total Economic Impact™ of Using ThoughtWorks' Agile Development Approach, Forrester Research Inc, Forrester Consulting, http://www.thoughtworks.com/sites/www.thoughtworks.com/files/files/forrester_tei.pdf, Retrieved 2008

[5] John Erickson, The Total Economic Impact™ Of Using ThoughtWorks' "Distributed Agile" Approach Single Company Analysis — Financial Services, Forrester Consulting, http://www.thoughtworks.com/sites/www.thoughtworks.com/files/files/forrester_tei_dist.pdf, Retrieved 2008

[6] Paul Devine, The Total Economic Impact™ Of Using ThoughtWorks' Agile Development Approach, http://www.thoughtworks.com/sites/www.thoughtworks.com/files/files/TEI-media.pdf, 2008. Retrieved 2009

[7] Bryan Campbell, Dr. Glenn Ray, Iterative Development Testing Approaches, http://bryancampbell.com/Files/Iterative_Dev_Testing_Approaches.pdf, Retrieved 2009

[8] Valery A. Martinez, Software Reliability Observations for Software Products Relying on Agile Programming Practices, EEL6883 Software Engineering II., April, 2008

[9] Yael Dubinsky, David Talby, Orit Hazzan, and Arie Keren, Agile Metrics at the Israeli Air Force, http://portal.acm.org/citation.cfm?id=1122089, Retrieved

2009

[10] Outi Salo, Pekka Abrahamsson: Integrating agile software development and software process improvement: a longitudinal case study. ISESE 2005: 193-202, 2005

[11] Capers Jones, Development Practices for Small Software Applications, http://www.stsc.hill.af.mil/crosstalk/2008/02/0802jones.html, Retrieved 2009

[12] Hector M. Olague, Letha H. Etzkorn, Sampson Gholston, Stephen Quattlebaum: Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes. IEEE Trans. Software Eng. 33(6): 402-419, 2007

[13] Wolfgang Holz, Rahul Premraj, Thomas Zimmermann, Andreas Zeller, Predicting Software Metrics at Design Time, http://www.springerlink.com/content/x3w4430557172vu4/, Retrieved 2009

[14] Danilo T. Sato, Hugo Corbucci, Mariana V. Bravo: Coding Dojo: An Environment for Learning and Sharing Agile Practices. AGILE 2008: 459-464, 2008

[15] By Bachir Kane, Estimating and Tracking Agile Projects, Published in PM World Today - May 2007 (Vol. IX, Issue V), 2007

[16] Tamara Sulaiman, Brent Barton, Thomas Blackburn: AgileEVM - Earned Value Management in Scrum Projects. AGILE 2006: 7-16, 2006

[17] Danilo Sato, Alfredo Goldman, Fabio Kon: Tracking the Evolution of Object-Oriented Quality Metrics on Agile Projects. XP 2007: 84-92, 2007

[18] Quentin Hart-Slater, APPLICATION OF THEORY OFCONSTRAINTS METHODOLOGY TO SOFTWARE PROJECT MANAGEMENT, http://www.docstoc.com/docs/2215698/APPLICATION-OF-THEORY-OF, Retrieved 2008

[19] Minna Pikkarainen, Mapping Agile Software Development onto ISO 12207, ITEA, http://www.agile-itea.org/public/deliverables/ITEA-AGILE-D2.9_v1.0.pdf, Retrieved 2009

[20] Aldo Dagnino, Karen Smiley, Hema Srikanth, Annie I. Antón, Laurie A. Williams: Experiences in applying agile software development practices in new product development. IASTED Conf. on Software Engineering and Applications 2004: 501-506, 2004

[21] Michael Karlesky, Mark Vander Voord, Agile Project Management (or, Burning Your Gantt Charts), Embedded Systems Conference Boston (Boston, Massachusetts) ESC 247-267, October 2008

[22] Harald Klein, Sabine Canditt, Using Opinion Polls to Help Measure Business Impact in Agile Development, International Conference on Software Engineering ,Proceedings of the 1st international workshop on Business impact of process improvements Leipzig, Germany, 25-32, 2008

[23] Peter Kokol, Vili Podgorelec, Maurizio Pighin, Using software metrics and evolutionary decision trees for software quality control , http://www.google.gr/url?sa=t&source=web&cd=3&ved=0CC0QFjAC&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.19.476%26rep%3Drep1%26type%3Dpdf&ei=TnOCTMfeEc-TjAf4nJyOCA&usg=AFQjCNGTqz8kMQEFRfHV_W0vgqYpk6JshA&sig2=Zy3gL6TO6OtkUQHGd-2SOA, Retrieved 2009

[24] David J. Anderson: Stretching Agile to fit CMMI Level 3 - the story of creating MSF for CMMI Process Improvement at Microsoft Corporation. AGILE 2005: 193-201, 2005

[25] Norman F. Schneidewind: Measuring and Evaluating Maintenance Process Using Reliability, Risk, and Test Metrics. IEEE Trans. Software Eng. 25(6): 769-781, 1999

[26] Outi Salo, Minna Pikkarainen, Agile Software Deployment of Embedded Systems, ITEA, http://www.agile-itea.org/public/deliverables/ITEA-AGILE-D4.4_v1.0.pdf, Retrieved 2009

[27] Teodora Bozheva, Maria Elisa Gallo: Framework of Agile Patterns. EuroSPI 2005: 4-15, 2005

[28] Alain Abram et al, Full Function Points for Embedded and Real-Time Software, UKSMA Fall Conference, London (UK) October 30-31 1998, http://www.gelog.etsmtl.ca/publications/pdf/379.pdf, Retrieved 2008

[29] Jennitta Andrea, An Agile Request For Proposal (RFP) Process, Proceedings of the Agile Development Conference (ADC'03) 0-7695-2013-8/03, 2003

[30] James P. Andrew, Measuring Innovation 2006, Boston Consulting Group, http://www.scribd.com/doc/6604806/Innovation-2006, Retrieved 2008

[31] Martin Kunz, Reiner R. Dumke, Niko Zenker: Software Metrics for Agile Software Development. Australian Software Engineering Conference 2008: 673-678, 2008

[32] Dr. Linda Rosenberg, Ted Hammer, Jack Shaw, SOFTWARE METRICS AND

RELIABILITY,
http://www.google.gr/url?sa=t&source=web&cd=2&ved=0CCYQFjAB&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.104.4041%26rep%3Drep1%26type%3Dpdf&ei=jm6CTLqSAZm8jAeWoaCbCQ&usg=AFQjCNHZ9bW9j8rlhjKq6EU_GqDKYKHVXQ&sig2=ZDBwCDnOVBZUOYIBtt019A, Retrieved 2009

[33] Liz Barnett, Metrics For Application Development, Forrester Research Inc, http://www.forrester.com/rb/Research/metrics_for_application_development/q/id/35916/t/2, 2005

[34] Global Environmental Management Initiative, MEASURING ENVIRONMENTAL PERFORMANCE: A Primer and Survey of Metrics In Use, GERMI, http://www.gemi.org/resources/MET_101.pdf, Retrieved 2008

[35] Deborah Hartmann, Robin Dymond: Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value. AGILE 2006: 126-134, 2006

[36] Mike Burba, Four Myths of Agile Development A Real-world “Enterprise Agile” Case Study, Compuware, http://www.agilejournal.com/articles/columns/case-studies/187-case-study-four-myths-of-agile-development, Retrieved 2009

[37] Deborah Hartmann, Robin Dymond, Appropriate Agile Measurement. http://www.berteigconsulting.com/AppropriateAgileMeasurement.pdf, both retrieved 2009

[38] Abrahamsson, P., AGILE Software Development of Embedded Systems, ITEA2 symposium, Berlin, Germany, 18-19 October 2007 http://www.agile-itea.org/public/papers/ITEA-Symposium_oct-07.pdf, Retrieved 2008

[39] Outi Salo, Pekka Abrahamsson: An iterative improvement process for agile software development. Software Process: Improvement and Practice 12(1): 81-100, 2007

[40] Stephanie Moore, Liz Barnett, Offshore Outsourcing and Agile Development, Forrester Research Inc, http://www.xgenta.com/docs/Forrester04_AgileOffshoring.pdf, Retrieved 2009

[41] Daniel Rawsthorne, Monitoring Scrum Projects with AgileEVM and Earned Business Value (EBV) Metrics, Danube Technologies, http://danube.com/system/files/CollabNet_WP_AgileEVM_and_Earned_Business_Value_Metrics_032510.pdf, Retrieved 2009

[42] Brad Appleton, Robert Cowham,Steve Berczuk, Lean-based Metrics for Agile CM Environments, cm crosroads http://www.cmcrossroads.com/agile-

scm/7820-lean-based-metrics-for-agile-cm-environments, Retrieved in 2009

[43] VersionOne, 3rd Annual Survey: 2008 “The State of Agile Development”, VersionOne, http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf, Retrieved 2009

[44] VersionOne, 2nd Annual Survey: The State of Agile Development, VersionOne, http://www.versionone.com/pdf/StateOfAgileDevelopmet2_FullDataReport.pdf, Retrieved 2009

[45] VersionOne, Survey: The state of Agile Development, VersionOne, http://trailridgeconsulting.com/surveys/state-of-agile-development-survey-2006.pdf, Retrieved 2009

[46] Ade Miller, Distributed Agile Development at Microsoft patterns & practices, Microsoft Corporation, download.microsoft.com/.../distributed_agile_development_at_microsoft_patterns_and_practices.pdf, Retrieved 2009

[47] Shine Technologies Pty Ltd, AGILE METHODOLOGIES Survey Results, SHINE TECHNOLOGIES, http://www.shinetech.com/attachments/104_ShineTechAgileSurvey2003-01-17.pdf, 2003

[48] Mike Griffiths, Using Agile Alongside the PMBOK, Originally published as part of 2004 PMI Global Congress Proceedings - Anaheim, California, 2004

[49] Michael James, An Agile Approach to “Metrics”: Applied Macromeasurements to Ensure On-time Delivery: Copyright ©2007 – 2008 Danube Technologies, Inc. http://danube.com/system/files/CollabNet_WP_Macromeasurements_061710.pdf, Retrieved 2009

[50] Santhana Krishnan, Agile Workshop: Agile Workshop: Agile Metrics http://www.slideshare.net/Siddhi/agile-workshop-agile-metrics, Retrieved 2009

[51] Mishkin Berteig, Agile Work Uses Lean Thinking, Berteig Consulting. http://www.berteigconsulting.com/Whitepaper%20-%20Agile%20Work%20Uses%20Lean%20Thinking.pdf, Retrieved 2009

[52] Carol Dekkers, Use Cases and Function Points -- Where's the Fit?, QUALITY PLUS TECHNOLOGIES, INC. Software and Technology Solutions, Published in IT Metrics Strategies, January 1999

[53] Rosenberg Linda, Parolek Frank and Botzum Steve, “The Role of Metrics in

Risk
Management across the Software Development Lifecycle", NASA GSFC., 2001

[54] Nary Subramanian, Lawrence Chung: Process-Oriented Metrics for Software Architecture Changeability. Software Engineering Research and Practice 2004: 83-89

[55] Robert C. Martin, PERT, CPM and Agile Project Management, Objectmentor.com http://www.objectmentor.com/resources/articles/PertCpmAgile.pdf, Retrieved 2009

[56] Evelina Lamma, Paola Mello, Fabrizio Riguzzi: A System for Measuring Function Points from an ER-DFD Specification. Comput. J. 47(3): 358-372, 2004

[57] Alain Abran, Pierre N. Robillard: Function Points Analysis: An Empirical Study of Its Measurement Processes. IEEE Trans. Software Eng. 22(12): 895-910, 1996

[58] Nicole Rauch, Eberhard Kuhn, Holger Friedrich (2008) Index-based Process and Software Quality Control in Agile Development Projects, Andrena Objects, http://comparch2008.ipd.kit.edu/fileadmin/user_upload/comparch2008/iert-rauch-andrena.pdf, Retrieved 2009

[59] James P. Andrew, Harold L. Sirkin, Knut Haanes, Davind C, Michael, Measuring Innovation 2007, The Boston Colnsulting Group, http://209.83.147.85/publications/files/Measuring_Innovation_Aug_2007.pdf , Retrieved 2009

[60] O. Salo and P. Abrahamsson, "Empirical Evaluation of Agile Software Development: A Controlled Case Study Approach," 5th International Conference on Product Focused Software Process Improvement, Japan, 2004.

[61] Salo, O. (2004) Improving Software Process in Agile Software Development Projects: Results from Two XP Case Studies. In: EUROMICRO 2004, France.

[62] Abrahamsson, P. & Koskela, J. (2004) Extreme Programming: A Survey of Empirical Data from a Controlled Case Study. In ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2004), Redondo Beach, CA, USA, 2004.

[63] Koskela, J. & Abrahamsson, P. (2004) On-Site Customer in an XP Project: Empirical Results from a Case Study. In: EuroSPI 2004, Trondheim, Norway., 2004

[64] Pikkarainen, M. and Passoja, U. "An Approach for Assessing Suitability of Agile

Solutions:A Case Study" The Sixth International Conference on Extreme Programming and Agile Processes in Software Engineering, Sheffield University, UK, 2005

[65] Pikkarainen, M., Salo, O. & Still, J. (2005). Deploying Agile Practices in Organizations:
A Case Study. In: European Software Process Improvement and Innovation (EuroSPI 2005), Budapest, Hungary, 9-11 November, 2005,

[66] Keränen, H., & Abrahamsson, P. (2005) Naked Objects versus Traditional Mobile Platform Development: A comparative case study, Proceedings of the 2005 31st EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA'05)
0-7695-2431-1/05, 2005

[67] Keränen, H., & Abrahamsson, P. (2005) A Case Study on Naked Objects in Agile Software Development, XP 2005, Sheffield University, UK. H. Baumeister et al. (Eds.): XP 2005, LNCS 3556, pp. 189–197, 2005.
© Springer-Verlag Berlin Heidelberg 2005

[68] Scott Ambler, Survey Says: Agile Works in Practice, Dr.Dobb's Journal http://www.drdobbs.com/architecture-and-design/191800169;jsessionid=PKDAU11XXKSZPQE1GHOSKHWATMY32JVN?queryText=Survey+Says%3A+Agile+Works+in+Practice, Retrieved 2009

[69] Erik Arisholm, Hans Gallis, Tore Dyba, Dag I.K. Sjøberg, Evaluating Pair Programming with Respect to
System Complexity and Programmer Expertise, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 33, NO. 2, 2007

[70] Lucas Layman, Laurie Williams, Lynn Cunningham, Exploring Extreme Programming in Context: An Industrial Case Study,
Proceedings of the Agile Development Conference (ADC'04) 0-7695-2248-3/04, US, 2004.

[71] Matthias M. M¨uller Walter F. Tichy, Case Study: Extreme Programming in a University Environment, Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), 2001
0270-5257/01

[72] P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, "New Directions on Agile Methods: A Comparative Analysis," International Conference on Software Engineering, 2003

[73] A. Cockburn, Agile Software Development. Boston: Addison-Wesley, 2002.

[74] P. Abrahamsson, "Extreme Programming: First Results from a Controlled Case

Study," presented at 29th Euromicro Conference, 2003.

[75] I. Stamelos and P. Sfetsos, “Agile Software Development Quality Assurance”, IGI Publishing, 2007, ISBN: 978-159904216-9.

[76] Minna Pikkarainen, Outi Salo: A Practical Approach for Deploying Agile Methods. XP 2006: 213-214

[77] Tore Dyba, Torgeir Dingsøyr, Empirical studies of agile software development: A systematic review, 0950-5849/$ - see front matter 2008 Elsevier B.V. All rights reserved. doi:10.1016/j.infsof.2008.01.006, 2008

[78] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, Mohamed Khalil, Lessons from applying the systematic literature review process within the software engineering domain, 0164-1212/$ - see front matter 2006 Elsevier Inc. All rights reserved. doi:10.1016/j.jss.2006.07.009, 2006

[79] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, Stephen Linkman, Systematic literature reviews in software engineering – A systematic literature review, Information and Software Technology 51 (2009) 7–15

7. APPENDIX A

Systematic Literature Review Data Synthesis Table.

ID	Authors	Topic	Institution/Company	Country	Type of study	Aproach	Population	Remarks
S1	Concas et al 2008	An Agile Development Process and Its Assessment Using Quantitative Object-Oriented Metrics	Universita di Cagliari	Italy	Experiment	Combined	N/A	• Significant difference in quality metrics of software developed duting the various phases with agile implementation. • Systematic improvement of software quality metrics when agile practices are thoroughly used by skilled developers.
S2	John Erickson 2008	The Total Economic Impact of ThoughtWorks' Agile Development Methodology. Single Company Analysis – Energy Services	North American-based Energy services organisation	US	Case Study	Combined	1	Within the five years of the application of Agile methodology the short iterative development cycles, empowered teams, transparency, constant reprioritization of requirements, and strong business participation, all lead to increased benefits: • Increasing team efficiency reduced both development and maintenance costs, and freed the development organization to address other projects. • Delivering core requirements in shorter timeframes greatly increased financial benefits. • Return of Investment (ROI) without the implementation of agile methodology: 14% • Return of Investment (ROI) with the implementation of agile methodology: 50%
S3	John Erickson	The Total Economic	Australian Insurance	Australia	Case Study	Combined	1	Agile methodology provided important decrease to costs and increased quality and speed of development over four years:

	2005	Impact™ Of Using ThoughtWorks ’ Agile Approach Single Company Analysis — An Australian Insurance Provider	provider					• Return of Investment (ROI) without the implementation of agile methodology: 16% • Return of Investment (ROI) with the implementation of agile methodology: 56% Not every project is a good fit for Agile development, but ThoughtWorks has demonstrated that applying Agile processes and strong project management can potentially be a benefit to high-risk projects, although highly-skilled staff will need to be hired and retained in order for this to succeed.
S4	Forrester Research Inc 2004	The Total Economic Impact™ of Using ThoughtWorks ' Agile Development Approach	Four Companies	International	Case Study		4	The enterprises discovered an expected return even on a risk adjusted basis, which exceeded their standard ROI: • Return of Investment on non risk-adjusted basis: 29% - 64% • Return of Investment on the Thoughtworks agile risk-adjusted basis: 31% - 66% Although agile practices are not suitable for every single project their implementation offers: • driving efficiency into the application development process • Reducing errors and changes caused by unforeseeable factors.
S5	John Erickson 2004	The Total Economic Impact™ Of Using ThoughtWorks ’ “Distributed Agile” Approach Single Company	Leading Insurance and Financial services organisation	US	Case Study	Combined	1	• projects were finished in almost half the time it took previously. • cash benefits were noticeably lower in the first year that Agile was applied but unexpectedly higher the following three years in the four-year cycle of the study. • Return of Investment (ROI) without the implementation of agile methodology: 4% • Return of Investment (ROI) with the implementation of agile methodology: 94%

		Analysis — Financial Services						
S6	Paul Devine 2008	The Total Economic Impact™ Of Using ThoughtWorks' Agile Development Approach Single Company Analysis — Media	Media Company	International	Case Study	Combined	1	• Higher quality and more efficient development led to a reduction in overall project duration, defects and rework. • This resulted in reduced costs to build, change and support a new development and production platform. • ROI without risk-adjustment: 64% • ROI with risk-adjustment: 40%
S7	Bryan Campbell, Dr. Glenn Ray 2009	Iterative Development Testing Approaches	Large Fortune 500 Company	US	Experiment	XP	11	• The project developed an iterative testing framework to support the iteration schedule developed for the project beforehand, identifying errors and addressing them respectively with each iteration, eventually reducing and minimizing them.
S8	Valery A. Martinez 2008	Software Reliability Observations for Software Products Relying on Agile Programming Practices	Sears Home Improvement Products.	US	Experiment	XP	4	• When it comes to XP, It is better for a task to be less large and complex or many problems should occur, especially when the team is pressed by time. • It would be wise to assign the junior developers with small and relatively simple sub-tasks and leave the complex parts to the senior developers. • In 28% of the total time a junior developer would introduce an error, whereas a senior developer in only 4%.

S9	Yael Dubinsky, David Talby, Orit Hazzan, and Arie Keren 2005	Agile Metrics at the Israeli Air Force	MAMDAS - a software development unit in the Israeli Air Force	Israel	Experiment	XP	60	• It is possible to introduce successfully a new benefactory methodology in a hard to change organization such as the army, overally increasing the confidence and ability to make short and long term decisions.
S10	Outi Salo, Pekka Abrahamsson 2005	Integrating Agile Software Development and Software Process Improvement: a Longitudinal Case Study	VTT, Technical Research Centre of Finland	Finland	Case Study	Coombined	5	• Successful implementation of agile methods within an organization is the communication and collaboration between the organizational level and the project-developing teams. • The organization needs to make appropriate decisions and take action to address the problems and/or mistakes that come up, and set the correct course that the developer's actions should then take. • The implementation of a project Facilitator in each team would ensure that the team can successfully overcome any obstacles they might find by always performing in tune with the organization's plans.
S11	Capers Jones 2007	Development Practices for Small Software Applications	N/A	US	Case Study	Combined	40/16 000	• For the Function Point method of cost estimation shows that there is not a single methodology that is adequate enough to be generally (universally) utilized. ••• Formal inspections are more than 65% efficient in finding bugs or defects, which is about twice the efficiency of most forms of testing. • Capability Maturity Model Integration (CMMI) and Agile approach are more efficient in defect and bug removal than the U.S. average • CMMI has a little higher ratings than Agile in the aforementioned, because of its more formal approach.
S12	Hector M. Olague, Letha H. Etzkorn, Sampson	Empirical Validation of Three Software Metrics Suites	IEEE	US	Case Study	Combined	4	• Chidamer and Kemerer's Metric Suite (CK) has proven to be more effective, reliable than the MOOD and QMOOD metrics and therefore better in predicting fault tendencies and frequency. •These metrics will not be so reliable for a long time since over the course of the continuous iterations of the software development, the software itself matures.

	Gholston, Stephen Quattlebaum 2007	to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes						
S13	Wolfgang Holz, Rahul Premraj, Thomas Zimmermann, Andreas Zeller 2007	Predicting Software Metrics at Design Time	Saarland University, Germany and University of Calgary, Canada	Germany, Canada	Experiment		89	• Knowing of the actual development cost of a project or component, an accurate estimate of the development cost that would occur if we had given a different set of imports can be measured. • Results in knowing whether the new parameters are hampering the development process as far as cost is concerned.
S14	Danilo Sato, Dairton Bassi, Mariana Bravo, Alfredo Goldman, Fabio Kon	Experiences Tracking Agile Projects: an Empirical Study	University of São Paulo	Brazil	Case Study	XP	7	• The projects had a higher desired score which underlines the team's increased willingness to adopt agile methodology. • The daily meetings of the team as suggested by the agile methodology (referred to as Retrospectives here) helped the teams to understand and stay in tune with the project's pace. • Retrospectives proved important and were also performed by the teams that could not follow the pace, nevertheless improving their performance in general. • The classes created with the agile approach resulted in classes of the same

	2008							complexity as before agile, albeit significantly smaller in size and therefore less prone to error and faults.
S15	Bachir Kane 2007	Estimating and Tracking Agile Projects	Ecole Superieure de Commerce de Lille	France	Case Study	XP	1	• Agile tracking techniques are not necessary to projects with short-release cycles whereas they are valuable to projects with long iterations. • They should be taken, however, under consideration in every project where the control and management shifts from one person to a whole team. • In every case, there is need for a Project Manager who should be able to plan correctly often in the middle of the project's cycle, guide and lead the team to the desired result.
S16	Tamara Sulaiman, Brent Barton, Thomas Blackburn 2007	AgileEVM – Earned Value Management in Scrum Projects	N/A	US	Experiment	Scrum	2	The Agile Earned Value Management metrics method allows teams utilizing the Scrum agile method to be able to obtain • accurate cost analysis • return of investment estimates and therefore steer the teams efforts accordingly.
S17	Danilo Sato, Alfredo Goldman and Fabio Kon 2007	Tracking the Evolution of Object-Oriented Quality Metrics on Agile Projects	University of São Paulo	Brazil	Case Study	XP	7	• The project without and agile methodology implementation turned out to be larger in size, more complex and significantly more prone to errors and maintenance. • High correlation between size, coupling and complexity metrics noted.
S18	Quentin Hart-Slater 2003	APPLICATION OF THEORY OF CONSTRAINTS METHODOLOGY TO SOFTWARE	University of Milwaukee	US	Case Study		6	• The overall capacity of the team developing a project is not utilized because it is limited by various constraints. • The management of the team should be able to identify and remove those constraints and therefore not only improve the team's productivity but that of the whole system as well. • Many constraints are abstract in definition and identifying them is difficult

		PROJECT MANAGEMENT						• Theory of Constraints helps management to identify the areas in the project than need to be addressed with higher priority.
S19	Minna Pikkarainen 2006	Mapping Agile Software Development onto ISO 12207	ITEA	Netherlands	Case Study	Combined	4	• Agile practices offer good techniques and mechanisms to improve software development as well as to implement and further test activities of the ISO 12207 process.
S20	Aldo Dagnino, Karen Smiley, Hema Srikanth, Annie I. Antón and Laurie Williams 2004	EXPERIENCES IN APPLYING AGILE SOFTWARE DEVELOPMENT PRACTICES IN NEW PRODUCT DEVELOPMENT	ABB Inc.	International	Experiment	ADEPT	2	• The Agile Development in Evolutionary Prototyping Technique (ADEPT) project satisfied 100% more customers with -50% the effort in documentation. • The traditional Incremental Development Model (IDM) project did not perform as well but was completed in 300 work hours less . • The team could identify and adapt to changes efficiently, while augmenting their communication and group skills.
S21	Michael Karlesky, Mark Vander Voord 2008	Agile Project Management (or, Burning Your Gantt Charts)	N/A	US	Case Study		N/A	Agile project management, in contrast to traditional project management of existing methodologies can offer: • Better risk and scope management. • Efficient budgets and schedules to create valuable products. • Lightweight and flexible documentation. • Early and effective integration error detection via the multiple iterations.
S22	Harald Klein, Sabine Canditt	Using Opinion Polls to Help Measure Business	Siemens	Germany	Case Study		340	• 49% said that the cost was reduced significantly with the introduction of agile practices. • 83% stated that agile practices offered a much better degree of business satisfaction.

	2008	Impact in Agile Development						• There are many communication-related issues which are important enough to cause instability and failure to a software development team. • Agile methodology works not with the individual but the team as a whole, making it an entity that is able to communicate and collaborate with other such entities.
S23	Peter Kokol, Vili Podgorelec, Maurizio Pighin 2001	Using software metrics and evolutionary decision trees for software quality control	N/A	Slovenia	Case Study		217	• 70% of the modules were correctly classified when only "α" metric was used. • 80% of the modules were correctly classified when the "α" metric was used in conjunction with decision trees. • Combining the "α" metric and decision trees one can successfully locate and predict modules with significantly large numbers of undetected faults. • Devoting some additional effort to maintain , enhance and test them out, leads to greater quality and reliability of the software in whole.
S24	David J. Anderson 2005	Stretching Agile to fit CMMI Level 3 - the story of creating MSF for CMMI® Process Improvement at Microsoft Corporation	Microsoft	US	Experiment		8	• Agile processes had to be enhanced in order to be combined with the Capability Maturity Model Integration. A successful modification has reduced the overall overhead/heaviness of the project by 85%. • Results in a CMMI approach that was more agile, adaptive and lightweight.
S25	Norman F. Schneidewind 1999	Measuring and Evaluating Maintenance Process Using Reliability, Risk and Test Metrics	NASA	US	Case Study		17	• Risk, reliability and test metrics can be used to measure both the quality of a product and the stability of a maintenance process. • Nowadays the challenge is to be able to choose the one best suited for our projects without having to modify (or with minimal modification) factors.
S26	Outi Salo, Minna	Agile Software Deployment of	ITEA	Netherlands	Case Study	Combined	N/A	• The Agile Deployment Model (ADM) helps to identify the correct and most useful agile methods and practices for each individual organization, by examining 6

	Pikkarainen 2005	Embedded Systems						key challenges. • ADM both introduces new (agile) techniques and utilizes traditional methods, applied in an agile context.
S27	Teodora Bozheva and Maria Elisa Gallo 2005	Framework of Agile Patterns	European Software Institute	Spain	Experiment	XP	13	• Productivity increased up to 73%. • Schedule deviation reduced by 7% - 38%. • Cost deviation decreased up to 31%. • Defect rates reduced by 10% - 83%. • Only one company decreased productivity and increased cost deviation. • Success of implementing agile methodology depends on factors generally related to development, Testing, Team effort, Management and Customer, and even then, its not fit for everyone.
S28	Alain Abram et al 1998	Full Function Points for Embedded and Real-Time Software	Software Enfineering Management Research Laboratory, SELAM	Canada	Case Study		N/A	• In order to develop real-time software benchmarking and estimation models, specialists must work with and build upon the knowledge that has been achieved by others over the years. • It is unwise, time-, and effort-consuming for them to start anew with their own set, or interpretation, of rules. • The industry should agree on a set of measurement rules so that everyone is working with the same basis and each individual's results are valid and comparable data that can be utilized by others in the future.
S29	Jennitta Andrea 2003	v	ClearStream Consulting	US	Experiment	XP	1	• The Request For Proposal is a process a company executes to find the vendor and/or product that best meet their criteria instead of developing new softwarein order to reduce ownership costs. • Through agile implementation there can be better guidance on how to identify and evaluate the company's key requirements, leading to better vendor choices. • The "planning" part of the Extreme Programming agile method, can be useful outside its development/programming context.
S30	James P. Andrew 2006	Measuring Innovation 2006	The Boston Consulting Group	International	Case Study		269	• Dissatisfaction with Return of (Innovation) Investment (ROI) decreased significantly over the years 57% - 48%, despite the improvement. • Over 50% of the companies measure Post-Launch Impact infrequently, or never. • 78% of responders relate developer incentives to Innovation metrics

								• Most useful and popular metrics were Time to Market, ROI and New Product Sales
S31	Martin Kunz, Reiner R. Dumke, Niko Zenker 2008	Software Metrics for Agile Software Development	Software Engineering Group University of Magdeburg	Germany	Experiment	XP	1	• The UnitMetrics measurement tool has made the integrated development environment(IDE) Eclipse, able to support agile development. • With over 100 downloads and utilizations, initial assumptions can be made as to how to better support agile software development and, in particular, its possible refactoring. • Very User-friendly Interface
S32	Dr. Linda Rosenberg, Ted Hammer, Jack Shaw 1999 (?)	SOFTWARE METRICS AND RELIABILITY	NASA	US	Case Study		56	• The Automated Requirements Measurement (ARM) can parse requirement documents and assess the vocabulary of the document as well as each individual specification statement. • In addition, it assesses the structure of the requirement document by identifying the number of requirements at each level of the hierarchical numbering structure. • An inconsistent or absent structure affects the software reliability. For example, by making it difficult to make changes later on.
S33	Liz Barnett 2005	Metrics For Application Development	Forrester Research Inc	US	Case Study		20	• The Balanced Scorecard model organizes Application Development (AD) Metrics into four categories and lets clients to choose which metrics to use form across them. • Although appearing costly in the business level, the metrics work very well in the application development level and offer a very realistic model for metrics selection and definition. • Data collection should be as unobtrusive as possible because it is relatively easy for developers to "fake" data in order to get their job done easier and circumvent the process not directly affecting them.
S34	Global Environmental Management Initiative	MEASURING ENVIRONMENTAL PERFORMANCE: A Primer and	Global Environmental Management Initiative	US	Case Study		41	• There is a very important connection between environmental performance and profitable returns that needs to be further examined and calls for the development of more sophisticated metrics. • By using metrics to measure and factor this connection we can show that environmental programs are not a necessary decreases in profitability that must be tolerated for the greater good.

	1998	Survey of Metrics In Use						• They should be viewed as (sometimes) radical innovations that add to profit instead of decreasing it by reducing and minimizing the costs and planning more efficient use of recourses.
S35	Deborah Hartmann, Robin Dymond 2006	Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value	-	US	Case Study		N/A	• A project which delivers high-value features to a customer early in the project may become self funding during the course of the project • There can be one "key" metric chosen and distinguished form the others, by which all aspects of the company can be measured, and it should be one that is closely related to the economics of the company. • The other metrics are renamed to "diagnostics" and their value is to supplement, diagnose and improve the key metric.
S36	Mike Burba 2007	Four Myths of Agile Development A Real-world "Enterprise Agile" Case Study	Compuware	US	Case Study		N/A	• Agile development iterations lead to features that answer closely to the business requirements and help root out the functions that would be unnecessary. • Planning in agile methodology is done throughout the development in different intervals and that helps the team overcome changes and problems. • Agile management is not more difficult than the traditional methods. With the right approach the agile methods scale without affecting the agile practices of the teams. • Aspects of agile methodology can be used outside the scope of programming, in more complex projects
S37	Serena 2007	Agile in the enterprise	Serena	N/A	Case Study		N/A	• 17% of the enterprises use Agile practices • 34% are aware of what the Agile methodology is • In order for company to adopt agile methods
S38	Pekka Abrahamsson, Ko Doom 2007	AGILE Agile software development of embedded systems	ITEA	Finland	Experiment	Combined	68	• 73% of the industrial projects with agile approach were considered successful or very successful • Analysis indicated that a team consisting on 17 developers using agile methodology developed software 8 times better and 3.5 times faster that the average measurements of the industry.
S39	Pekka Abrahamsso	An iterative improvement	ITEA	Finland	Case Study	Scrum	35	• 60% did not make use of any available agile practice • Almost 80% were not even remotely familiar with the agile method, Scrum

	n, Outi Salo 2007	process for agile software development. Software Process: Improvement and Practice						• 77% of those that later tried and experienced Scrum practices found them beneficial.
S40	Stephanie Moore, Liz Barnett 2004	Offshore Outsourcing and Agile Development	Forrester Research Inc	US	Case Study		N/A	• Most Indian companies are opposed to Agile methods, as they are antithetical to less-disciplined development processes. • The few Indian firms that have adopted Agile, was because of their customers' demand and not of their own initiative. • Offshore projects can benefit from Agile methods, but introduction must be gradual. • Excellent team communication and individual resolve is required for the success of such endeavors.
S41	Daniel Rawsthorne 2008	Monitoring Scrum Projects with AgileEVM and Earned Business Value (EBV) Metrics	Danube Technologies	US	Case Study	Scrum	1	• The Earned Value Management(EVM) metrics can be applied to agile projects, but in order to give valid results they should be coupled with the Earned Business Value(EBV) metric. • On the other hand, the EBV could be utilized in the absence of agile metrics to offer substantial overall results.
S42	Brad Appleton, Robert Cowham,Steve Berczuk 2009	Lean-based Metrics for Agile CM Environments	CM Crossroads	US	Case Study		1	Configuration management metrics applied to an agile environment raise two main concerns: • Agile methodology's approach with very small feedback loops which occur frequently makes for a lot of complex details to attempt to measure in order to gain valid results. • Agile methodology's approach favoring "people and interactions" over "process and tools" makes it very hard to acquire successfully such metrics transparently and unobtrusively.

S43	VersionOne 2008	3rd Annual Survey: 2008 “The State of Agile Development”	VersionOne	US	Survey		2319	• The two most important barriers ion the road to agile adoption is the unwillingness to change 44%, and the lack of ability to change the organizational culture • 57% of responders have their Agile teams distributed. • The organization's greatest concerns regarding Agile implementation is the lack of up-front planning 46% and loss of management control 37%. • 49% of responders prefer Scrum as their preferred Agile practice. • 55% of responders state that over 90% of their agile projects have been successful • 17.4% stated that their agile projects have been 100% successful. • 23% stated that the reason for their failure in agile projects was that the organization's philosophy and culture were at odds with the core agile values.
S44	VersionOne 2007	2nd Annual Survey: “The State of Agile Development”	VersionOne	US	Survey		1681	• The greatest obstacle in the road for agile adoption is the company's un-willingness to change • 57% of responders have their Agile teams distributed • 37% of responders prefer Scrum as their preferred Agile practice. • 30% of responders stated that they have considered Agile in order to benefit in the management of changing priorities. • The organization's greatest concern regarding Agile implementation is the lack of up-front planning. 34% • 33% of the organizations have adopted Agile in over 75% of their software development projects. • 84% of the Organization gave adopted agile methods in some or all parts of their software development process. • Agile implementation increased productivity by 90% and reduced software defects by 85% • Use of agile methodology has accelerated time-to market by 83% and reduced the overall cost by 66%
S45	VersionOne	Survey: The	VersionO	US	Survey		722	• 84% of the Organization gave adopted agile methods in some or all parts of

	2006	state of Agile Development	ne					their software development process. • Ability to manage changing priorities enhanced by 92% with Agile. • Increased team morale and productivity as well as software quality by 74% • Reduced risk and increased the time-to-market by 72% • 40% or responders prefer use Scrum as their preferred Agile practice. • The organization's greatest concern regarding Agile implementation is the lack of up-front planning. 20%
S46	Ade Miller, 2008	Distributed Agile Development at Microsoft patterns & practices	Microsoft	US	Case Study		N/A	Before distributed agile development is adopted, some important features must be taken under consideration: • Ability to organize teams at distance and making communication efficient. • Decreased team function and performance due to distance results in increased delivery times and increased chance of failure. • Team members need to be resolute about their work so that they will not fall back on their tasks, since there is less pressure.
S47	Shine Technologies 2003	SHINE TECHNOLOGIES AGILE METHODOLOGIES Survey	Shine Technologies	Australia	Survey		181	• 84.7% of respondents had average or greater knowledge of Agile methods. • 46% stated that costs were unchanged with implementation of Agile. • 93% stated that productivity was better or significantly better. • 88% stated that quality was better or significantly better. • 83% stated that business satisfaction was better or significantly better. • 59% or respondents utilize and favor XP over other agile practices.
S48	Mike Griffiths 2004	Using Agile Alongside the PMBOK	Quadrus Development	US	Case Study		N/A	• To measure how well Agile can be used Project Management Body of Knowledge (PMBOK) we need metrics that are relatively simple and relevant to the Goal. • Agile project management offer tracking and reporting metrics which do not hamper the workload of a project. • Agile should be used alongside traditional project management techniques on high-execution risk projects.
S49	Michael James	An Agile Approach to	Danube Technolo	US	Case Study	Scrum	N/A	• Macromeasurements like Velocity and Earned Business Value metrics for Scrum and Running Tested Features metrics for XP should be measured once per iteration.

	2008	"Metrics": Applied Macromeasurements to Ensure On-Time Delivery	gies					• Product and release plans can be adapted using empirical data, without affecting the teams' individual organization.
S50	John D. McGregor 2005	JOURNAL OF OBJECT TECHNOLOGY	Luminary Software	US	Case Study		N/A	• Selective choice of strategic metrics can lead to summarized information, that is otherwise too widespread for one person to examine personally • Global quality metrics provide information to determine what should be measures regardless of the process model utilized
S51	Mishkin Berteig	Agile Work Uses Lean Thinking	Berteig Consulting Inc.	Canada	Case Study		8	• Agile implementation minimizes setbacks like barriers and obstacles when the team is trying to hasten their efforts. • In case of setbacks like documentation policies and corporate standards the Process Facilitator helps the team overcome and work around them. • A high degree of trust must be developed between customers employees and management in order for all of them to pursue the goal of becoming Agile.
S52	Carol Dekkers 1999	QUALITY PLUS TECHNOLOGIES, INC. Software and Technology Solutions: Use Cases and Function Points -- Where's the Fit?	QUALITY PLUS TECHNOLOGIES	US	Case Study		N/A	• Use cases require project teams to devote more time in planning and documentation of requirements in earlier stages of development. • Function points will supplement the utilization of use cases by identifying lack of clarity in them, as well as assist to define certain ambiguous points which would have passed undetected
S53	Dr. Linda H. Rosenberg, Frank	The Role of Metrics in Risk Management	NASA	US	Case Study		N/A	• When supposting a risk-management program with the correct selection of metrics like the Requirement and Product Quality and the Test and Process Efficiency, the development is enhanced.

	Parolek, Steve Botzum 2001	Across the Software Development Lifecycle						• Such metrics offer important information for decision-making and support the risk-management program by measuring the risk status and the results of the mitigation processes.
S54	Lawrence Chung, Nary Subramanian 2004	Process-Oriented Metrics for Software Architecture Adaptability	N/A	US	Experiment		2	Process-Oriented Metrics for Software Architecture Adaptability(POMSAA) is a process-oriented framework which calculates the necessary metrics for Adaptability: • Traces metrics to their respective requirements • Analyzes the reasons for strategic strengths and weaknesses in metrics • Evaluates and suggests improvements on the architecture of the framework
S55	Robert C. Martin 2003	PERT, CPM, and Agile Project Management.	Object Mentor Inc.	US	Case Study		N/A	• The CPM method is not suitable to accurately represent dependencies because it doesn't support displaying more tasks than those that are currently underway • The PERT method is good for large-scale projects but not if we want to manage projects at the "per day" or "per person" level. • The Agile Project Management (APM) method is useful for it is suitable to measure the kinds of tasks that software projects incorporate and also it displays the uncertainty and randomness often associated with such endeavors.
S56	Evelina Lamma, Paola Mello, Fabrizio Riguzzi 2004	A System for Measuring Function Points from an ER–DFD Specification	The British Computer Society	Italy	Case Study		7	• It is possible and effective to measure Function Points from a system which is expressed in entity relationships(ER) and data flow diagrams(DFD) • This automatic measuring process saves lot of human work hours and results are in accord with the traditional human counters. • The IFPUG counting rules are made more solid and rigorous, which helps avoiding confusion between the different counters of the same project.
S57	Alain Abran, Pierre N. Robillard 1996	Function Points: A Study of Their Measurement	Universite du Quebec a Montreal,	Canada	Case Study		N/A	• The Function Point metric when applied should not only focus on the end product of the measurement system but in each step of the process, which most likely will provide new information. • It can also identify which an intermediate step is more meaningful and important by measuring the information lost (instead of added) in each step.

		Processes and Scale Transformations						
S58	Nicole Rauch, Eberhard Kuhn, Holger Friedrich 2008	Index-based Process and Software Quality Control in Agile Development Projects	andrena	Germany	Case Study	Scrum	N/A	• The ISIS system for quality management has the project logbook in its core. • Inaccurate and false trends can be counteracted against with haste. • Assumes to be able to draw conclusions about the whole project while examining only parts of it.
S59	James P. Andrew, Harold L. Sirkin, Knut Haanes, Davind C, Michael 2007	Measuring Innovation 2006	The Boston Consulting Group	International	Case Study		377	• Less than 50% of respondents are satisfied with their return of innovation investment. • 42% of responders stated that their organization is not planning to increase its innovation investments. • 69% of responders stated that they develop deep understanding of customers and ensuring high-level sponsorship. • 58% stated that they provide strong support to project developer teams.
S60	Outi Salo, Pekka Abrahamsson 2004	Empirical Evaluation of Agile Software Development: the Controlled Case Study Approach	VTT Technical Research Centre of Finland	Finland	Experiment	XP	4	• Recruiting more experienced subjects was worth the effort due to their higher level of knowledge and skills • Development standards should have been designed for the team prior to the project • The team presence factor (the time the team spent within project facilities) could have an influence on the eXpert project outcome.
S61	Outi Salo	Improving Software	VTT Technical	Finland	Case Study	XP	2	• Examination in post-iteration workshops revealed that in the second project(zOmbie) there were significantly more positive as well as negative findings in

	2004	Process in Agile Software Development Projects: Results from Two XP Case Studies	Research Centre of Finland					relation to the first one(eXpert). • All top 5 most important positive findings in eXpert are related to XP practices whereas in zOmbie on human and environmental practices. • 20% of zOmbie's negative findings were related to its Off-shore Customer factor and the communication problems that are implied. • Estimation of the tasks was problematic in both projects and contributed by above 15% on their negative findings. • Decrease of negative findings towards the end of the projects leads to satisfaction, adaptation and improvement of the team and its efforts.
S62	Pekka Abrahamsson, Juha Koskela 2004	Extreme Programming: A Survey of Empirical Data from a Controlled Case Study	VTT Technical Research Centre of Finland	Finland	Case Study	XP	6	• Customer was present at 80% of the development time, which proved to be a great motivating factor for the team. • From the above percentage, 21% was devoted to assist with development, 42.6% in planning and 29.9% in acceptance and testing. • In the first release of the project 81.7% of the programming was done in pairs, whereas in the second it decreased to 75.9%
S63	Juha Koskela, Pekka Abrahamsson 2004	On-Site Customer in an XP Project: Empirical Results from a Case Study	Sheffield University	UK	Case Study	XP	1	• On-site customer involvement is critical to the success of XP. • The ability to contact the customer at will had a great impact in team effort as they delivered 250% more Value for the Customer, within initial schedule.
S64	Minna Pikkarainen, Ulla Passoja 2005	An Approach for Assessing Suitability of Agile Solutions: A Case Study	VTT Technical Research Centre of Finland	Finland	Case Study		3	• Agile assessment assists in finding the appropriate agile practices that are needed by an organization in order to improve a specific aspect of its software development process. • Process assessment can be performed fairly easily, by using simple documentation, close communication and rapid feedback. • Accurate expected results after agile implementation can be found and examined through assessment.
S65	Minna	Deploying	F-Secure	US	Case		1	• Improvement Actions were much more than the negative experiences, 32 vs

	Pikkarainen, Outi Salo, Jari Still 2005	Agile Practices in Organizations: A Case Study	Corporation		Study			11. • The pilot projects provide the organization with valuable feedback on implementing the agile process. • Negative experiences were always more than positive experiences and improvement actions taken.
S66	Heikki Keränen, Pekka Abrahamsson 2005	Naked Objects versus Traditional Mobile Platform Development: A Comparative Case Study	Sheffield University	UK	Case Study	XP	2	Naked Objects implementation on the zOmbie project led to many interesting data: • Development duration was significantly reduced by -50%. • The total effort in hours was reduced by -64% • Team productivity substantially increased by 54% • The total size of the software's code was decreased by -39%.
S67	Heikki Keränen, Pekka Abrahamsson 2005	A Case Study on Naked Objects in Agile Software Development	Helsinki Stock Exchange	Finland	Case Study		1	• Naked Objects framework is not yet mature and should not be considered for application with multiuser security requirements and lots of objects. • Naked Objects allow the fast realization of user-stories, which allows better understanding of the demands and changes need to be done. • Implementaion requires high throughput and has been considered difficult.
S68	Scott Ambler 2006	Survey Says: Agile Works in Practice	N/A	Us	Survey		4232	• 54% Have limited knowledge of agile methods • 60% of responders reported increased productivity and 66% reported higher quality. • 58% of responders reported that their stakeholder were satisfied.
S69	Erik Arisholm, Hans Gallis, Tore Dyba, Dag I.K. Sjøberg	Evaluating Pair Programming with Respect to System Complexity	N/A	International	Experiment	XP	295	• Increased efforts to correctly perform the tasks by 84% • Results from measuring required time to perform tasks and percentage of correct solutions offered did not offer significant differences.

	2007	and Programmer Expertise						
S70	Lucas Layman, Laurie Williams, Lynn Cunningham 2004	Exploring Extreme Programming in Context: An Industrial Case Study	Sabre Airline Solutions	US	Case Study	XP	1	• Developer productivity increased by 50% • Software quality before release increased by 65% • Software quality after release increased by 35%
S71	Matthias M. Muller, Walter F. Tichy 2001	Case Study: Extreme Programming in a University Environment	Computer Science Department Universitat, Karlsruhe	Germany	Experiment	XP	12	• 48% enjoyed working in accord with pair programming principles. • 96% found and gave support to their partner in order to find solutions • XP is best adopted by small teams, due to the higher communication requirements, which would disorganize larger teams.
S72	Forrester's Executive Research Panel 2004		Forrester Research Inc	US	Survey		115	• Over 40% of responders value the results of quality and productivity metrics • Over 66% of organizations stated that the project management and cost management metrics are very valuable to them. • 63% stated that there is initiative to improve such metrics in the company.