ISSN:2320-0790

An International Journal of Advanced Computer Technology

Systematic Review of Metrics in Software Agile Projects

Amrita Raj Mukker[1], Dr. Latika Singh[2], Anil Kumar Mishra[3]

[1]Department of computer science (M.Tech), amritamukker26@gmail.com
[2]Department of computer science (Associate Professor), latikasingh@itmindia.edu
[3]Department of computer science (Assistant Professor), anilkrmishra@itmindia.edu

Abstract: *This is a review paper in which things discussed would be about the various software metrics and about agile methodology. Nowadays Agile practices are increasing popularity in software development communities. This paper is a summary of the various metrics, agile and agile methodology used in software industries. Further this papers shows how Extreme Programming practices (XP) could enhance the development and implementation of a large -scale and geographically distributed systems .Adaptation of Extreme Programming practices in the project has increased the human factor output and its has helped in bringing up promising idea to enhance the conceptualization and implementation as well as future extensions of large scale projects.*

Keywords: *Agile, Methodology, Metrics, Quality, XP.*

1. Introduction

Software systems are becoming more complex and there have been a number of methodologies to deal with the inherent complexity of large software systems, such as agile development processes and component-based development.. The agile methodologies have been adopted rapidly in the last years and have been the dominant development processes.

Recent development efforts have shown a need to frequently reassess requirements for the intended software product and, consequently, replan the project, leading to significant product redesign and refactoring. In this paper I will be discussing about the quality metrics, effect of agile on projects. Agile when used with correct metrics then it leads to a well developed project which satisfies the user with the best quality product.

2. Experimental Software Metrics

A **metric** is a standard for measuring or evaluating something. A **measure** is a quantity, a proportion, or a qualitative comparison of some kind.

- *Quantity*: "There are 25 open defect reports on the application as of today."
- *Proportion*: "This week there are 10 percent fewer open defect reports than last week."
- *Qualitative comparison*: "The new version of the software is easier to use than the old version."

Three kinds of metrics:

All the metrics fall under three main categories. They are:

- Informational – tells us what's going on
- Diagnostic – identifies areas for improvement
- Motivational – influences behaviour

Metrics as indicators

1. Leading Indicator: Suggests future trends or events.
2. Lagging Indicator: Provides information about outcomes.

In fact software metrics is a collective term used to describe the very wide range of activities concerned with measurement in software engineering. These activities range from producing numbers that characterise properties of software code (these are the classic software "metrics") through to models that help predict software resource requirements and software quality. The subject also includes the quantitative aspects of quality control and assurance - and this covers activities like recording and monitoring defects during development and testing.

Metrics can be categorized under the following types of metrics.

1. Business Metrics
2. Base Metrics
3. Quality Metrics
4. Product Metrics
5. Process Metrics

6. Testing Metrics

TABLE 1
Software Metrics

Metrics	Types	Description
Base Metrics	1. LOC	1. Productivity = KLOC / Person-month 2. Quality = Defects / KLOC 3. Cost = $ / LOC 4. Documentation = pages of documentation / KLOC
	2. Functional Point	FC*VAF(Value Adjustment Factor)
Business Metrics	1. Business Value Delivered	They measure what happened in the last financial period
	2. NPV	Summation Ct/(1+r)^t C = net cash in period t; r = cost of capital
	3.Customer satisfaction	This index is surveyed before product delivery and after product delivery (and on-going on a periodic basis, using standard questionnaires).
	2. Functional Point	FC*VAF(Value Adjustment Factor)
Quality Metrics	1. Complexity	CCM= CCCM+ IMCM
	2. component reusability	CRLLOC=reuse(C)/C*100%
	3. cohesion	
	4. coupling	C(X,Y)=I+N/N+1 N= No of interconnectin betweeb(X,Y), I= Level of highest level of couplingtype found
	5. maintainability	MTTC (Mean time to change) -- Once error is found, how much time it takes to fix it in production.
	6. Integrity	Integrity = Summation [(1 - threat) X (1 - security)]
	7. Reliability	Mean time between failures (MTBF) - Total operating time divided by the number of failures. MTBF is the inverse of failure rate.
	8. Quality of Testing	No of defects found during Testing/(No of defects found during testing + No of acceptance defects found after delivery) *100
	9. Quality of Testing	No. of defects found during Testing/(No. of defects found during testing + No of acceptance defects found after delivery) *100
	10. Correctness	Defects / KLOC or Defects / Function points
Product Metrics	1. Product volatility	Ratio of maintenance fixes,vs. enhancement requests
	2. Defect Density	No of Defects / Size (FP or KLOC)
	3. Mean Time to Failure	It calculates the mean time between two failures
	4. Customer problem metrics	PUM(problem per user menth)= total problem that customer reported foa a time period/ total number of license-month of the software during the period.
	5. Complexity of delivered product	Predicted defects and maintenance costs, based on complexity measures
	6. Defects detected in production	Defects detected in production/system size
Testing Metrics	1. RTF curve	Running means that the features are shipped in a single integrated product. Tested means that the features are continuously passing tests Features means End-user features; pieces of the customer-given requirements
	2. Business value	Highest priority is to satisfy the customer through early and continuous delivery of valuable software.
	3. Velocity	An empirical observation of the team's capacity to complete work per iteration.
	4. Putting Velocity to Work: Burn Charts (Burn-down charts)	1. It shows the estimated effort against time. The units of time are generally the iterations 2. It is likely to be more accurate as at this stage the user stories should have been discussed in detail. 3. It estimations been refined based on additional information
Direct Metrics	1. Size	Size Planned
	2. time	Size Actual Time Meeting
	3. effort	1. Effort Planning 2. Effort Overview 3. Effort Preperation 4. Effort Meeting 5. Effort Rework
	4. Defect	1. Defects Found Major 2. Defects Found Minor 3. Defects Corrected Major 4. Defects Corrected

		Minor
Earned value analysis	1. Planned Value	BAC * Planned Percent Complete
	2. Earned Value	BAC * Actual Percent Complete
	3. Cost Performance Index (CPI) 4. Schedule	EV/AC
	5. Performance Index (SPI)	PV/AC
	6. ETC	(BAC-EV)/CPI
	7. EAC	BAC/CPI OR AC+ETC
REQUIREMENTS QUALITY METRICS	Agile Requirements Ambiguity	Ambiguity = Number of Misinterpreted Requirements / TNR (Note: Ideal=0; Extremely Ambiguous=1)
	Agile Requirements Completeness	Completeness = Number of Correctly Validated Requirements/TNR (Note: Ideal=1; No Validation=0)
	Aspectual Density Per Sprint	NORASP Density Per Sprint (NORASP-DPS) = Σ Number of NORASP / Σ (Number of NORASP + Number of NORARC + Number of NORPOL)
	Agile Requirements Maturity Index (functional)	ARMI Functional (R_i) = Number of AUCs (R_i) – (Number of Changed AUCs (R_i) + Number of Newly Added AUCs (R_i) + Number of Deleted AUCs (R_i)) / Number of AUCs (R_i) (Note: R_i is release i. For functional requirements only.

3. Agile Methodology

Many technological ambitious products were designed with new complex functionality. The demand for functions establishes a need for new software requirements to deliver new functionality. Due to the fast alteration and the high cost of change in the late life cycle phases the agile software development method becomes more important in this field of application. Agile software development methods like eXtreme programming try to decrease the cost of change and therewith reduce the overall development costs. The different cost of change for agile software development in comparison with traditional software development according to the project progress is shown in Figure below.

TABLE II
Agile Methodologies

TYPE	STRENGTH	WEAKNESS
XP	1. Customer ownership of feature priority, developer ownership of estimates. 2. Frequent feedback opportunities. 3. Most widely known and adopted approach, at least in the U.S. 4. Strong technical practices.	1. Requires onsite customer. 2. Documentation primarily through verbal communication and code. For some teams these are the only artifacts created, others create minimal design and user documentation. 3. Difficult for new adopters to determine how to accommodate architectural and design concerns.
SCRUM	1. Complements existing practices. 2. Self organizing teams and feedback. 3. Customer participation and steering. 3. Priorities based on business value. 4. Only approach here that has a certification process.	1. Only provides project management support, other disciplines are out of scope. 2. Does not specify technical practices. 3. Can take some time to get the business to provide unique priorities for each requirement..
LEAN	1. Complements existing practices. 2. Focuses on project ROI. 3. Eliminates all project waste. 4.Cross-functional teams	1. Does not specify technical practices. 2. Requires constant gathering of metrics which may be difficult for some environments to accommodate. 3. Theory of Constraints can be a complex and difficult aspect to adopt
FDD	1. Supports multiple teams working in parallel. 2. All aspects of a project tracked by feature. 3. Design by feature and build by feature aspects are easy to understand and adopt. 4. Scales to large teams or projects well.	1. Promotes individual code ownership as opposed to shared/team ownership. 2. Iterations are not as well defined by the process as other Agile methodologies. 3. The model-centric aspects can have huge impacts when working on existing systems that have no models.
AUP	1. Robust methodology with many artifacts and disciplines to choose from. 2. Scales up very well. 3. Documentation helps communicate in distributed environments. 4. Priorities set based on highest risk. Risk can be a business or technical risk.	Higher levels of ceremony may be a hindrance in smaller projects. Minimal attention to team dynamics. Documentation is much more formal than most approaches mentioned here
Crystal	1. Family of methodologies designed to scale by project size and criticality. 2. Only methodology that specifically accounts for life critical projects. 3. As project size grows, cross-functional teams are utilized to ensure consistency. 4. The "human"	1. Expects all team members to be co-located. May not work well for distributed teams. 2. Adjustments are required from one project size/structure to another in order to follow the prescribed flavor of Crystal for that project size/criticality. 3. Moving from one flavor

	component has been considered for every aspect of the project support structure. 5. An emphasis on testing is so strong that at least one tester is expected to be on each project team.	of Crystal to another in mid project doesn't work, as Crystal was not designed to be upward or downward compatible.
DSDM	1. An emphasis on testing is so strong that at least one tester is expected to be on each project team. 2. Designed from the ground up by business people, so business value is identified and expected to be the highest priority deliverable. 3. Has specific approach to determining how important each requirement is to an iteration. 4. Sets stakeholder expectations from the start of the project that not all requirements will make it into the final deliverable.	1. Probably the most heavyweight project compared in this survey. 2. Expects continuous user involvement. 3. Defines several artifacts and work products for each phase of the project; heavier documentation. 4. Access to material is controlled by a Consortium, and fees may be charged just to access the reference material.

4. Effect of Metrics on Agile Projects

Good metrics should enable the development of models that are efficient of predicting process or product spectrum. Thus, optimal metrics should be:

- Simple, precisely definable—so that it is clear how the metric can be evaluated;
- Objective, to the greatest extent possible;
- Easily obtainable (i.e., at reasonable cost);
- Valid—the metric should measure what it is intended to measure; and
- Robust—relatively insensitive to (intuitively) insignificant changes in the process or product.

4.1. Scrum Tracking Metrics:

In Scrum, various metrics are used to track the progress of the project and individual performance of team members. Different metrics are used to track the Scrum project.

1) Velocity: Velocity is a metric that is used to track the amount of Product Backlog effort that a team completes in a Single Sprint.

2) Standard violation: The Standard violation metric is used to track the number of standards violated per Sprint.

3) Business value delivered: Business value can be measured in terms of story points, number of stories, or an abstract measure that measures how much value the business attaches to a feature or story.

4) Defects per iteration: This metric is calculated either as a simple count or count weighted by the severity of defects that was introduced during a Sprint. Because the Sprint is of a small duration, defects are very costly in Scrum and hence should not pile up.

5) Number of stories: This metric is calculated as a simple count or count weighted by the story complexity, such as simple, medium, or complex, of the number of stories in a release or a Sprint.

6) Level of automation: The level of automation in testing is one of the key success factors of Scrum.

7) Number of tests: A measure of the number of tests that have been developed, executed, and passed to validate a story, epic, or the entire release.

Quality metrics are indeed helpful in bringing to focus defect as they occur and prompt the need to comply with project requirements thus preventing avoidable rework at a later stage of the project. Whatever areas a project management chooses to focus on the project team will begin to work seriously and shift their emphasis to perform well against that metrics. There are several metrics that agile project team must follow to measure the project progress, at the same time the agile team have to determine which of the metrics are more important to project sensor, project team, project manager and other involved in the project to achieve this it is necessary to study a few agile project metrics. XP and Scrum are the most commonly used agile methodologies in the software industries.

REFACTORING:

To support agile software development and especially refactoring, mainly source-code based product metrics are beneficial to increase quality and productivity. Primarily internal quality attributes have to be ensured to control the source-code quality and to evaluate refactoring steps. If we combine refactoring with software measurement we can give advice about the following aspects:

- Appropriate point in time for necessary refactoring steps
- Significance of a refactoring step and the apparent quality effect
- Side effects of refactoring steps

With these three aspects one can ensure quality along refactoring steps. The metrics should deliver indices for distinct refactoring steps and they should be easily interpretable. The measurement results should be a trigger or activator for useful refactoring steps and they should avoid quality loss through refactoring steps.

4.2. Method Designs for XP: (critical Factors)

1. *Team productivity:* SR(success rate) is directly proportional to (Team productivity).
2. *Number of user stories implemented:* SR is directly proportional to (number of user stories implemented).
3. *Pair programming percentage:* SR is directly proportional to (Pair programming %)
4. *Number of Post-release Defects:* SR is inversely proportional to (Number of Post-release Defects).

5. *Customer involvement percentage:* SR is directly proportional to Customer involvement
6. *Total work effort:* SR is inversely proportional to (Total work effort).
7. *Number of Post-release enhancement suggestions:* SR is directly proportional to (Number of Post-release enhancement suggestions).

- **Two Modelling Approaches for XP:**

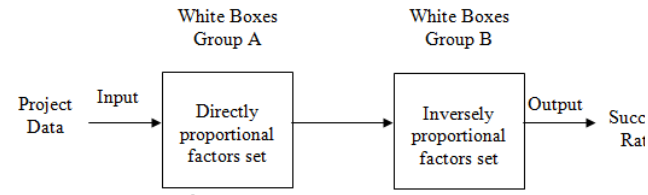

Figure 1: First approach steps

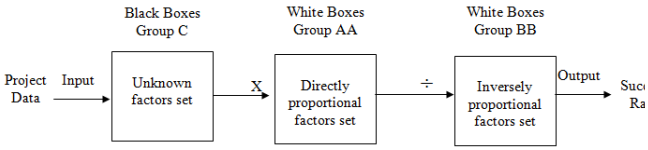

Figure 2: Second approach steps

4.3. "UnitMetrics" – Measurement Tool

To support agile software development at the origin a tool was implemented as a Plug-In for an Integrated Development Environment (IDE). Eclipse, because of the preconditions and its open source character [Eclipse], was chosen. Especially because of the many iterations of a product until it reaches final status the integration in a development environment instead of a stand-alone realization is recommendable.

The project was realized as an open source project and published under sourceforge.net [UnitMetrics]. Since August 2007 the plug-in was downloaded over 300 times but empirical information about the usage of the tool is not yet available [Kunz 2008].

The general goal was to create a measurement tool which expands the Eclipse-Development-Environment to provide source-code analysis by the use of appropriate metrics.

Four major features should be supported:

- Continuous analysis
- Fundamental interpretation
- Interactive visualization
- Extensibility

4.4. NORPLAN (Non-Functional Requirement planning):

Most of the project management metrics include the actual value as well as a second value that captures the impact of that metric. The impact of each metric captures the weight of how important that metric is to the NORPLAN (Non-Functional Requirement planning) algorithm used in risk calculation. The total impact of all metrics combined is equal to 100 points. For instance, if one metric has a very high impact and represents a 10% importance compared to all other metrics, then this metric should be assigned an impact value of 10 (leaving 90 points of impact for all other metrics). In this manner, different metrics could be assigned different weights depending on the specific nature of the agile team, the project, and the complexity of the system being developed.

OUTCOME:

This case study involved three experiments that used three different priority schemes for calculating the requirements implementation sequence (NORPLAN). The first scheme was based upon prioritizing requirements according to the highest business value. The second scheme prioritized requirements according to the highest calculated technical and project management risks (riskiest requirements first). The third scheme was based upon prioritizing requirements according to the lowest calculated risk (i.e. riskiest requirements last).

It resulted that:

- If requirements were to be implemented according to the originally-requested priority sequence (highest business value first), it would take 6 releases (21 sprints),
- When the riskiest requirements were planned and implemented first, the overall duration of the implementation was shortened by almost 2 months. In this experiment, only 5 releases (17 sprints) were computed
- When the riskiest requirements were planned and implemented last, the overall duration of the implementation was shortened by one month. Only 5 releases were computed, which was similar to the second experiment, but it took 19 sprints (1 month longer) instead of 17 sprints.

4.5. Extreme Programming Applied in a Large-scale Distributed System:

4.5.1 TRAFFIC VIOLATION CLEARANCE SYSTEM:

The traffic violation clearance system is a web-based system that integrates with existing state traffic systems and other violation source mediators to handle over-speed and red light rushing violations by prohibiting any further violating vehicle operations on the state server and sending an SMS to the registered vehicle owner notifying them about the violations they have committed. When vehicle owner comes to the annual renewal of their vehicle license or any other operation on the vehicle, the traffic authority operator receives a message on the violations committed for the registered owner of the vehicle, and the operation cannot be completed until the registered owner clears their violations. When violations are cleared, vehicle operations are allowed on the state server, and an SMS will also be sent notifying the registered owner about the clearance, and subsequently they can complete the original process he requested from the traffic authorities.

4.5.1.1 Environment setup for XP methodology:

Having no central repository imposed extra challenges on how violation clearance has to be approached. The most challenging issue for the system was to come up with a suitable design to process violations on such diversified, geographically distributed environment. Moreover, bringing together a team of developers and customers to sit together and form the requirements collectively imposed another dimension of communication management and took a significant amount of time in the start up of the project in order to bring the team to an alignment with XP practices.

The team had only worked with a disciplined approach similar to waterfall software development methodology in their previous projects and it was their first interaction with XP. They were trained on agile software development methodologies in general, with extensive consideration for migration to XP methodology.

On project initiation, practices took place such as team formation where a team of 5 developers and 4 customer representatives were seated together in an open space large office. An environment preparation process was taken place where Client/Server source code control tool "SVN" was setup in addition to an agile project management tool "Mingle" and a continuous integration server "CruiseControl" was installed. The work office space was organized to fit both teams: developers and customers.

4.5.1.2 Outcome:

When agile methodology was applied to the project the quality of the project was enhanced. Collaboration and satisfaction was high. The development team's interest (buy-in) was high and the velocity was increased by 3%. The number of iteration was 6 iterations and 3 releases.

TABLE IV
IMPACT OF EXTREME PROGRAMMING PRACTICES ON THE PROJECT

	Release 3
Code gallery exhibt	4
User stories completed/velocity	40
User stories increase*	3%
Story points	650
Story points increase*	3.7%

*Increase percentage depends on the first column as a baseline i.e. release 2 user stories increase is calculated by 13/10=1.3, and story points increase is calculated as 220/140=1.5.

TABLE V
Below is a comparison table of various project.

NAME	TEAM MEMBERS	DURATION	COMMENTS
INTERNAL PROJECT			
Project A (legacy traffic system)	12	2 years	Large number of developers were working on requirements gathering
Project B (foreigners registration system)	6	6 months	Most of projects time was in requirements gathering and no single line of code was written
EXTERNAL PROJECTS			
Project C (National Health Insurance Fund)	18	6 months	Completed on time, using crystal clear methodology
Project D (University registration and management system)	6	3 months	Completed before time using XP

Furthermore, a comparison has been carried out between the current project and other company projects accordingly and information is summarized in Table V.

Thus we conclude the adaptation of XP practices to this project has enhanced the "buy-in" of all team members, and as a result the human factor in the development process was maximized. This led to customer demand on adoption of the XP Methodology in the current development of the legacy Traffic distributed system as well.

5. MYTHS ABOUT AGILE

Common Myths And Reality:

Myths are widely accepted but mistaken beliefs. There are some myths about Agile:

- Requires no documentation and works informally on trust.
- Requires mature teams that are co-located.
- Allows no time for designing.
- Cannot work with CMMI or other process models.

But the reality is different:

- Agile requires just enough documentation.
- Co-located, mature teams help in communication, but are not a prerequisite.
- Agile requires iterative, incremental design and not an all-encompassing rigid design.
- Most CMMI level 5 and ISO certified teams use the agile methodology.

6. CONCLUSIONS AND FUTURE WORK

After the study of various metrics and agile methodology two main things were concluded:

i. XP and Scrum are the two main methodologies of Agile which are popular and are used in most of the software projects.

ii. When agile methodology is applied to large scale and small scale projects then the quality of the product increases which we can measure by using quality metrics explained in table 1.

Further we will apply agile methodology (XP) to a real project and compare their quality and productivity by using quality metrics and product metrics and this would be the future work.

REFERENCES

[1] Divya Chaudhary* , Prof. Rajender Singh Chhillar, "International Journal of Advanced Research in Computer Science and Software Engineering", Volume 3, Issue 7, July 2013, www.ijarcsse.com

[2] Ozgur Aktunc, "Entropy Metrics for Agile Development Processes", 2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops.

[3] Jean-Marc Desharnais, Buğra Kocatürk, Alain Abran, "Using the COSMIC Method to Evaluate the Quality of the Documentation of Agile User Stories", 2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement.

[4] Chaelynne M. Wolak, "Extreme Programming (XP) Uncovered", A paper submitted in fulfillment of the requirements for DISS 725 – Research Paper Four DISS 725 Spring 2011

[5] Reiner R. Dumke, Andreas Schmietendorf, Martin Kunz, Konstantina Gorgieva, "Software Metrics for Agile Software Development", Source: Gilb, T.: Estimation or Control? – Thesis paper, URL: http://www.dasma.org, December 2007.

[6] D. I. Heimann, P. Hennessey, and A. Tripathi, "A Bipartite Empirically- Oriented Metrics Process For Agile Software Development," ASQ Software Quality Professional, vol. 9, no. 2,2007

[7] Norman E Fenton, Martin Neil, "Software Metrics: Roadmap", 44 (0)208 530 5981 norman@dcs.qmw.ac.uk, 44 (0)1784 491588 martin@dcs.qmw.ac.uk.

[8] Panos Kourouthanassis[1], Diomidis Spinellis[1], George Roussos[2], and George M. Giaglis, "MyGROCER Ubiquitous Computing Environment".

[9] Tor Stålhane[1], Geir Kjetil Hanssen,"The application of ISO 9001 to agile software development", The Norwegian University of Science and Technology SINTEF ICT.

[10] Weam M. Farid, Frank J. Mitropoulous, "NORPLAN: Non-functional Requirements Planning for Agile Processes", 978-1-4799-0053-4/13/$31.00 ©2013 IEEE

[11] Elmuntasir Abdullah, El-Tigani B. Abdelsatir "Extreme Programming Applied in a Large-scale Distributed System", 2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING(ICCEEE)

[12] *H. Kiwan, Y. L. Morgan, Luigi Benedicenti, "* Two Mathematical Modeling Approaches For Extreme Programming", 2013 26th IEEE Canadian Conference Of Electrical And Computer Engineering (CCECE)

[13] Monika Agarwal, Prof. Rana Majumdar, "Tracking Scrum Projects Toola, Metrics and Myths about Agile", International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 3, March, 2012)