IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 5, Ver. IV (Sep. - Oct. 2016), PP 60-73
www.iosrjournals.org

Process performance model for predicting Delivered Defect Density in a software scrum project

Srijith Sreenivasan .ManimaranSundaram
Research Scholar at Anna University, Chennai, India

Abstract: *Agile scrum projects are gaining popularity for software development and delivery, due to its inherent attributes of flexibility in execution and quicker time to market. However, the critics of Agile have been raising questions on the predictability of success of Agile execution as the emphasis given to quantitative management using predictive techniques for an Agile project may not be to the extent of large mission critical projects which uses in-process leading indicators to predict outcomes. One possible solution to this challenge is to use the best practices from industry adopted models like the Capability Maturity Model Integration® for planning and managing Agile software delivery. In this paper, we present a study were a prediction model was developed for use in Agile projects for systematic in-process monitoring and decision making. The purpose of this paper is to propose a predictive model as a driver for continuous improvement of Delivered Defect Density in the context of a Scrum project.Our findings show that using the prediction model, the success of Agile project could be forecasted and controlled during its planning and execution phases as against waiting for the final release to determine success of delivery.*

Keywords*: process performancemodel;agile;software development; software metrics; scrum*

I. Introduction

In today's rapidly changing business environment, agile projects provide relevant solutions by demonstrating adaptability in execution. In that sense, agile has become highly successful and has been adopted by several organizations as a way of executing projects (Cohn 2010). One of the areas where Agileprojectscan further improve is by gaining quantitative predictability in operations at the same time being flexible in execution. Though there are several approaches suggested within the agile framework itself for scaling up agile operations and handling complexities, managing complex software development by way of having predictable indicators is an opportunity which Agile projects can explore. It is in this context that established models like Capability Maturity Model Integration for Development (CMMI$^{®}$) are extremely useful for managing software development in a methodical way (McMahon 2011). The higher maturity levels (Level 4 and Level 5) of CMMI provides a quantitative approach to monitor and control the project using prediction models. Such prediction models is an aid for Project Managers in predicting the outcomes of their project using in-process factors which are expressed in quantitative terms.

In this paper, a study is performed to predict the Delivered Defect Density (DDD) of an agile project, using a process performance model which is developed based on historical information of the organization. The model is developed based on the guidelines available in the Capability Maturity Model Integration (CMMI), especially on the process areas of Organizational Process Performance and Quantitative Project Management.
The key contributions of this paper are as follows.

- proposal of a model for Agile scrum projects
- evaluation of a model for predicting and improving delivered defect density in SCRUM projects

The paper starts with defining the background and context of the study, research methodology and study of existing literature in this area. A brief description of the major knowledge areas considered in the study namely Agile, CMMI and High Maturity practices in CMMI is then presented. An introduction on measures, metrics and prediction models is also provided. The development of the Agile Prediction Model and the application of the model Agile SCRUM projects is addressed as the main theme of the paper which concludes with the advantages, shortcomings and limitations of the model developed.

1. Background of the study

In organizations where quality of software is a primary goal for the software projects, it is useful to measure and observe Delivered Defect Density as a lagging indicator signifying the quality of developed software. Though this is a good metric to indicate the goodness of the delivered software, the metric lacks the ability to be controlled, as it is an outcome of the software development process and not an in-process metric. In this context, the need for in-process or leading indicators which will help practitioners get visibility on the outcome is important. In this paper, the processes and sub-processes leading to Delivered Defect Density is studied to arrive at the in-process factors having an impact on the outcome metric. The study aims to develop a prediction model using these factors to predict the Delivered Defect Density.

II. Research Methodology

Case study method is used as the methodology for this research. The reason for selecting this method for developing a prediction model is as per the points put forward by Runeson and Höst(2009). As per their study, software engineering research attempts to determine the development and maintenance of software done by software engineers and project managers.They concluded that research questions in software engineering are suitable for case study research, which is especially true for this study as well, since the factors considered includes people, process and technology aspects.

This study was done in the Global In-house Information Technology partner of a major multinational bank in Europe. This organization has development and maintenance projects done in the financial domain, many of which are critical in nature due to the requirements of stringent quality. Hence Delivered Defect Density is a critical measure used by the Project Managers of this organization to determine the quality of software developed and delivered.

Following are the major steps followed in this study.

(i) Case study design:The research objective definition and execution of the case study was planned at this stage.
(ii) Preparation for data collection: In this stage, steps for data collection was arrived at using a data collection plan.
(iii) Collecting evidence: Data was collected based on the data collection plan.
(iv) Analysis of collected data: Data was analyzed and modelled at this stage.
(v) Reporting: Reporting of the results along with a demonstration of usage of the model in an actual project was done at this stage.

2.1 Research Questions

The research questions considered for this study are:

RQ1: What are the processes and sub-process which have a relationship to Delivered Defect Density?
RQ2: What are the factors influencing Delivered Defect Density in this organization?
RQ3: What is the result of using prediction model for Delivered Defect Density in an Agile SCRUM project?

2.2 Data collection and analysis procedures

Data required for the study was collected from the previous projects executed in the organization. A data collection plan was developed for data collection and used for gathering data on the selected sub-process metrics. The operational definition of the metrics was done and used while collecting data to ensure consistency and correctness of the collected data.

Analysis of the data was performed using correlation and regression. Multiple Linear Regression technique was used for modelling and to arrive at the prediction equation.

2.3 Threats to validity

The factors that may have an impact on the validity of this study are presented in this section. These factors are organized as Construct Validity, Internal Validity and External Validity.

2.4 Construct Validity

Though an operational definition exists for each of metrics that were collected, it can be subject to measurement errors. This is especially true since the base measures are collected by Project Managers based on their understanding of the measure. This risk was alleviated by training all Project Managers in data collection and providing an overview on the operational definition of the metric and its intent.

2.5 Internal Validity

The approach used for this study relies on the causal relationship of the factors to Delivered Defect Density. And such relationship is proved using data empirically. However, there could be other factors also, some of which are non-controllable in nature, which could have an impact on the dependent variable used. This threat was overcome by determining the regression co-efficient, R-sq adj. In this study, the R-sqadj value of the equation was around 91%, indicating that 91% of the variation of the outcome can be explained by the factors considered for the study.

2.6 External Validity

This study is performed using the data collected from one organization only. The nature and context of this organization can have an impact on the output of this study. However, it was determined that the process followed in this organization is the standard Agile SCRUM execution process without much deviation and tailoring. The organization is also appraised at the Maturity Level 5 of the CMMIfor Development model, indicating that it is benchmarked among the other major Software Development organizations in the Industry. Hence it was determined that this organization can be a typical research specimen where a study on the applicability of Agile and application of CMMI High Maturity prediction model could be carried out.

III. Review of existing literature

In this section, the related work in this field and research gap is addressed. The context of this study with respect to the knowledge gap is briefly addressed.

3.1 Related work

Abrahamsson et al (2007) studied the effort prediction in iterative software development. Their objective was to arrive at a more accurate method of effort estimation in the eXtreme Programming environment, where traditional methods of estimation may not be very accurate. They proposed an incremental model that can be developed without entirely depending on historic data.

Prediction of software defects in various lifecycle models including iterative development is discussed by Fenton et al (2007). They used Bayesian Network to study about software project as a whole and the balance between time, resources and quality. They studied that the factors used for prediction of outcomes in iterative lifecycle, covering both the assessment of process quality and the product quality.

Hearty et al (2009) developed a model for predicting project velocity in eXtreme Programming using Bayesian Belief Network model. The model was validated in an industrial context by the researchers and could demonstrate results. The outcome of their model was the level of functionality delivered over time, which was predicted using factors from early lifecycle stages and knowledge about the presence of an onsite customer. This model used the data from early stages of a project to refine the predictive results in later stages.

A study of software metrics for assessing the phases of an Agile project was done by Concas et al (2012). They modelled the agile software development practices using Java by considering the various agile practices to the outcome of object oriented metrics. Their conclusion was that “good” agile practices are related to “better” values of the object oriented metrics.Eg: when agile practices such as pair programming, test driven development and refactoring are used, the quality metrics shows improvement; when these practices are discontinued, the metrics shows significant worsening. They used case study method for analysis and conclusion.

Li and Leung (2014) developed a prediction model to study the error proneness of agile object oriented system using Bayesian techniques. They proposed Bayesian Network as a technique to predict fault proneness compared to other modelling techniques using study of static metrics. As per their study, Bayesian Networks yielded better predictability when compared to Logistic Regression and Naïve Bayes.

Singh and Verma (2014) developed a fault prediction model using cluster based classification. They argued that using a cluster based approach increases the probability of defect detection which results in more reliable and test effective software. They demonstrated an 83% probability of detection compared with standard methods of defect prediction.

The performance of defect prediction in rapidly evolving software was studied by Cavezza et al (2015). The prediction accuracy on whether or not an attempted commit is going to introduce a bug was studied in a high commit frequency context, depicting rapid evolution of software. They concluded that a dynamic approach to prediction works better than a static prediction model for evolving software. Their study proved useful in development scenarios having shorter release cycles and higher code variability.

A generalized software reliability model for predicting release time for open source software was proposed by Washizaki et al (2015). They recognized that though there is sufficient reliability in Agile owing to shorter release cycles and incremental development, there is a significant challenge on predicting the release date of target software. The model was developed using an ito-type stochastic differential equation extended from nonlinear differential equation. They also evaluated the prediction accuracy using Error Rate, as a function of Predicted and Actual release day. The study reported Error Rates which are very less, hence proving the accuracy of prediction.

3.2 Research Gap

Though there are similar papers which studied the prediction models in Agile as explained in the previous section, a study on predicting Delivered Defect Density using process performance model concept of CMMI in an Agile SCRUM project is not performed and published. The factors considered for the prediction of

Delivered Defect Density, as a combination is not considered for prediction of outcome for agile software development projects. The scope of this study is limited to this area and is very specific in developing such model and determiningits usage in an actual industrial project.

IV. Agile Software Development

The word ‘agile’ means ability to move quickly and easily. Agile found its place in Software development when traditional approaches for software development failed to find answers to some of the most pertinent challenges regarding software development such asSalo et al (2007):

- Lack of ability to handle changes effectively,
- Lack of ability to handle uncertainties,
- Lack of visibility on the working software till the end of software development.

Agile understands and recognizes that Software development is a human intensive activity, which can only be considered as an overall progression of activities to achieve an ultimate goal (Sutherland 2013). The four guiding principles of Agile are the following, which are commonly referred to as the agile manifesto.

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

One of the most popular approaches in Agile is the SCRUM methodology. SCRUM is not just a process or a technique, rather it is a framework within which various processes and techniques can be used for building working software. The SCRUM is an iterative and incremental framework for software development which structures development in cycles called Sprints which are time-boxed. This means that they end on a specific date whether or not the work is completed. The sprints are executed by people who have got a definite role to play. Some of the expected roles in Scrum are the Scrum Master, Product Owner and the Scrum Team.

The input for Scrum team is the product backlog which is a list of items or features for the Software to be developed. This list which is prioritized by Product Owner according to business importance, are broken down into small chunks of work by the Scrum team, further to which each of the team member signs up for their work. Sprints are executed in this fashion for the defined time boxes. During sprint execution, every day the team does a Scrum meeting in which the following three questions are answered.

- What did I accomplish yesterday,
- What will I be working on today,
- Are there any obstacles on my way

At the end of the sprint, two meetings are conducted called the Sprint Review meeting and Sprint retrospective meeting. In Sprint review meeting, the completed work is demonstrated to relevant stakeholders. In Spring Retrospective meeting the team inspects their completed sprint and adapts their performance for future sprints. The Scrum project gets completed, once all the planned sprints are executed.

Cohen and Costa (2003) observed that agile methods are actually a collection of different techniques (or practices) that share the same values and basic principles of traditional development. It is the focus and values behind agile methods that differentiate them from moretraditional methods. The application of agile methods with CMMI is studied by Silva et al (2015). They noted that agile methodologies have been used by companies to reduce their efforts to reach levels 2 and 3 of CMMI, and even to obtain level 5. They observed benefits such as improvements in organizational aspect, greater team and customer satisfaction, further integration, cost reduction, process assimilation, increasing productivity and reducing defects in their study. They argued that the feasibility of using CMMI together with the agile development is manifested on both sides.

The Capability Maturity Model Integration®

CMMI, which is published by the Software Engineering Institute, part of Carnegie Mellon University provides a landscape for Organizations to move through a progressive and well defined plateaus known as Maturity levels. The model is a collection of best practices Organized systematically into knowledge groups known as Process Areas. Each maturity level with the exception of Maturity level 1 has many Process Areas which needs to be complied with for attaining the Maturity Level (CMMI Product Team2010).

Level 1 of the CMMI model is an ad-hoc level where there are no processes established. Success of projects in a Level 1 organization largely depends on individual heroics, but is not predictable and sustainable. Maturity Level 2 of CMMI, otherwise known as Managed level is one in which basic Project Management processes are in place. Measurements are done at defined intervals and used for in-process course correction. In Maturity Level 3 of the model, a standard way of executing projects is established across the Organization. In this level, the Organization is more proactive and Engineering processes protected by solid project management are effectively implemented. Level 4 of CMMI is also known as quantitatively managed level, indicating that the focus of Organization at this level is quantitative and statistical thinking. Achieving the goals of projects and in turn that of the organization are done through usage of quantitative process performance models, commonly known as Prediction Models. Maturity Level 5 is an Optimizing level in which the focus shifts to constant improvements and innovations directed towards business goals of the Organization. Improvement of the process is inherently part of everybody's role in the process, resulting in a cycle of continual improvement.

There are two representations in CMMI namely staged and continuous. The staged representation enables organizations to achieve maturity levels, whereas continuous representation enables organizations to achieve capability levels. In staged representation, organizations are expected to follow an evolutionary approach to process improvement, moving through each level incrementally. In continuous representation, organizations can select which process areas to improve upon, and decide to which capability level this improvement has to be done. For the purpose of this study, the staged representation of CMMI is considered, as the study is focused on some of the requirements of maturity levels 4 and 5, which corresponds to the staged representation. A brief expectation of maturity levels 4 and 5 of the staged representation is explained in the following section.

High Maturity practices in CMMI

The practices in Level 4 and 5 of CMMI model are commonly referred to as High Maturity practices. There is a reason for combining these two levels and using a single word to signify its commonality. Primarily, the practices of the four Process Areas in these two maturity levels are highly inter-related which makes it logical for Organizations to implement them in unison. Secondly, the maturity level 4 and 5 of the model when implemented consistently, results in a system of prediction, causal analysis and continuous improvement which has to operate seamlessly to obtain desired results. In a High Maturity system, organizations are expected to periodically review business objectives and maintain quality and process performance objectives in accordance with the business objectives. Statistical and other quantitative techniques (including process performance baselines and models) are used to understand process performance and target areas for improvement that would enable achievement of the objectives (CMMI Product team2010).

There are four process areas in maturity levels 4 and 5 together. These are Organizational Process Performance, Quantitative Project Management, Causal Analysis and Resolution and Organizational Performance Management. The purpose of Organizational Process Performance (OPP) is to define an environment for establishing quantitative practices in the organization. The key concepts of this process area are Process Performance Baselines and Process Performance Models.The purpose of Quantitative Project Management (QPM) process area is to define quantitative goals at a project level and to manage projects quantitatively using models and baselines. The Causal Analysis and Resolution (CAR) process area helps the organization to identify the causes of defects, problems and surprise positive outcomes and to initiate corrective actions on the identified causes. The Organizational Performance Management (OPM) process area defines requirements to identify incremental and innovative improvements focused on the business objectives of the organization.

When the requirements of these four process areas are implemented in an organization, the result is a quantitative environment where organizations and projects strive for continuous improvement through the usage of quantitative information for baselining, predictive modelling and usage of prediction models for in-process control of project execution.

The quest for agile teams in achieving CMMI Maturity Level 5 is studied by Cohan and Glazer (2009). Sutherland and Jakobsen (2008) studied the combined use of Scrum and CMMI. They observed that CMMI and Scrum can be successfully mixed. The mix results in significantly improved performance while maintaining compliance to CMMI Level 5 as compared to performance with either CMMI or Scrum alone.

Need for measures, metrics and prediction models

A good quality management system provides warning signs to Project Managers early in the lifecycle of the project and not only towards the end. For this to happen, it is essential to predict outcomes using certain factors during the development phase of the project, so that controlling these factors during project execution will ensure that the final outcome of software development is successful. If these predictions can be made early in the lifecycle, then there is opportunity for Project Managers to make real time decisions which will help in suitable course corrections (Kan2003).

One of the basic problems of software industry is not being able to make a realistic commitment on the effort, schedule and quality of a software project or product (Jalote2005). A probable reason for this is because of the inability of estimation and forecasting techniques to accurately capture the complexities during development processes. Moreover, the lack of reliable past data and metrics leads to poor quality estimates. This results in unreasonable expectations, failed commitments and thereby customer and team dissatisfaction.

Measurements and Metrics contribute in determining the range of expected results that can be achieved by following organization's standard processes. The process capability thus achieved is used by software projects to establish and revise their process performance goals and to analyze the performance of the projects' defined software processes. They also represent the health of a project with respect to the organization's standards.

Major objectives of the metrics are as follows: (Fenton 1997)

- To understand the quality of the product better
- To determine the effectiveness of the software process
- To improve the estimation quality
- To initiate timely corrective actions based on prediction of efforts, schedule and quality

Metrics can be broadly classified as leading and lagging indicators (Regan 2002). Lagging indicators signify the outcome of a process, whereas leading indicators helps in in-process control of the process, which will in-turn influence the outcome. Such an approach of using leading indicators to control and predict lagging indicators is achieved through prediction models.

The usefulness of a model depends on the accuracy of its results. One the determinants of quality of a model is its empirical validity (Kan2003). Empirical validity refers to the situations in which the predictive validity and usefulness of the model is supported by organizational empirical data. The empirical validity will vary drastically across organizations, lifecycle methods and types of projects. Hence it is important for an organization to develop models based on historical data derived from its historical projects.Inorder to test the usefulness of the model proposed, it was validated with an actual industrial project.

The following sections of this paper explains how a prediction model was developed using historical data of an Organization for Agile projects and how it was used in a project to make useful decisions for mid-course corrections and Quantitative Project Management.

V. Development of Agile prediction model

Process performance models or prediction models are used to estimate or predict the value of a process performance outcome from the current and historical values of process and product measurements. These process performance models typically use process and product measurement data collected throughout the lifecycle of the project to predict the probability of achieving objectives that otherwise cannot be measured until later in the project's life. The approach which we followed for developing a prediction model consisted of below steps.

(i) High level process mapping of Agile project execution
(ii) Development of Metrics Architecture
(iii) Characterization of processes and sub-processes
(iv) Determination of logical relationship of factors
(v) Operational Definition of metrics
(vi) Statistical validation of relationship
(vii) Conversion to probabilistic model

5.1 High level process mapping of agile project execution

The first activity performed to develop a model was to depict the execution of agile process using a Process map as shown below. Such a process map helped in identifying the processes and sub-processes involved in the agile project. Knowledge of the processes and sub-processes was used to identify the influencing factors at a sub-process level, which will have an impact on the goals of the project.

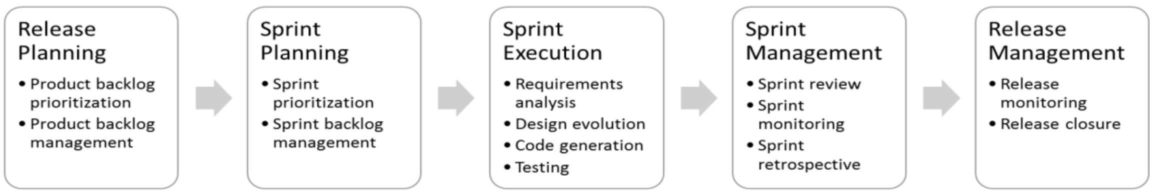


Fig 1 Process and sub-processes of Agile project execution

5.2 Development of Metrics Architecture

The goals which are important at a project level were derived from the organizational business objectives. The processes and sub-processes identified in the previous step were related to the relevant project level objectives. This representation of the relationship between the organizational business objectives, project level objectives, processes and sub-processes was called as a Metrics Architecture, since it provided a conceptual view of the relationship between variables at different levels.

In-order to develop the Metrics Architecture, the business objective of the organization was studied. The business objective was obtained from the organizational vision, which provided a direction for Project Managers to steer their projects. In this case, the business objective identified was related to customer satisfaction, which stated “Achieve Customer Satisfaction through consistent delivery of good quality software, on-time, every time”.

From this high level objective, two lower level goals were identified, one representing quality and another representing process performance. Following the direction provided by the business objective, the quality goal identified was Delivered Defect Density (DDD). This indicated the extent of delivered to the customer after final release. The second goal that was derived representing process performance was Schedule Variance (SV). This goal indicated the extent of delay in calendar days for meeting customer release commitments. Quantitative targets were then set for both these objectives, as defined in Table 1.

Table 1 Quantitative targets for agile project

Metric	Goal
Delivered Defect Density	Less than 0.5 defects per 100 Story points
Schedule Variance	Less than 5%

The quantitative goals defined in Table 1 were arrived at based on the current performance of the organization, which was studied quantitatively using a process performance baseline.The data considered for this study was from the historical projects in the organization executed for the last several months. CMMI defines process performance baseline as a documented characterization of process performance, which can include central tendency and variation. It goes on to mention that a process performance baseline can be used as a benchmark for comparing actual process performance against expected process performance. The mean and standard deviation of process performance of Delivered Defect Density and Schedule Variance, was observed to be as in Table 2.

Table 2 Process performance baselines

Metric	Mean	Standard Deviation
Delivered Defect Density	0.47 defects per 100 story points	0.02 defects per 100 story points
Schedule Variance	3.56%	0.72%

Based on the process performance baselines for DDD and SV, it was inferred by the Organizational management that a goal of less than 0.5 defects per 100 story points and less than 5% is achievable by the projects following similar processes. However, the projects could tailor these defined goals with justifiable rationale and approvals from Senior Management as needed.

Once the goals were defined, the processes and sub-processes which has an impact on the quality and process performance objectives were identified and represented. The purpose is to identify only those relevant and critical processes and sub-process which will have an influence on the project objective and focus on them. Such a representation aided in the next activity of identifying measures corresponding to each of the process and sub-process.

The size measurement used in agile projects is Story Points. Though story points have several faults as observed by Buglione and Abran (2007), story points will give a relative understanding of the size of a feature or user story with respect to the other features of user stories in the application. Hence within the scope of the user stories in an application, story points will serve the purpose of identifying the relative size of features with an acceptable level of accuracy (Cohn 2005). However, when used across projects and applications for the purpose of comparing and making decisions, functional measures such as IFPUG or COSMIC Function Points are considered to be better sizing options (Nasir 2006).

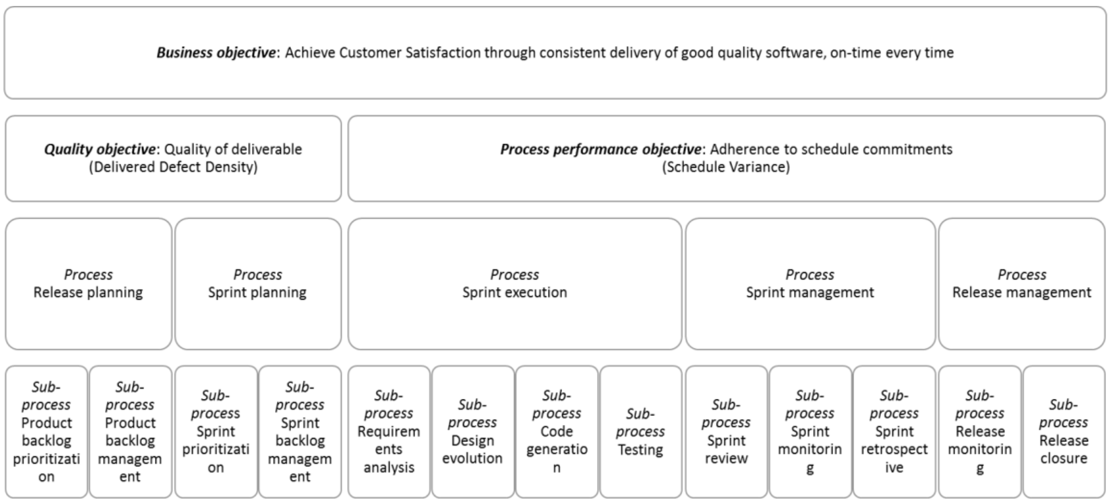

Business objective: Achieve Customer Satisfaction through consistent delivery of good quality software, on-time every time
Quality objective: Quality of deliverable (Delivered Defect Density)
Process performance objective: Adherence to schedule commitments (Schedule Variance)
Process Release planning
Process Sprint planning
Process Sprint execution
Process Sprint management
Process Release management
Sub-process Product backlog prioritization
Sub-process Product backlog management
Sub-process Sprint prioritization
Sub-process Sprint backlog management
Sub-process Requirements analysis
Sub-process Design evolution
Sub-process Code generation
Sub-process Testing
Sub-process Sprint review
Sub-process Sprint monitoring
Sub-process Sprint retrospective
Sub-process Release monitoring
Sub-process Release closure

Fig 2 Metrics Architecture to identify sub-processes

5.3 Characterization of processes and sub-processes

Having identified the processes and sub-processes for agile development, the next step performed was to characterize each of the processes and sub-processes using suitable measures. The measures were chosen after careful analysis as to really represent the process and sub-process from an efficiency and effectiveness point of view.

Table 2 Sub-process metrics mapping based on High level processes

Process	Subprocess	Metrics
Release Planning	Product Backlog prioritization	Release stability Index
	Product Backlog management	Story DD
Sprint Planning	Sprint prioritization	Sprint Stability Index
	Sprint backlog management	Sprint Effort Variance
Sprint Execution	Requirements Analysis	Story Points Coverage, Defects per story
	Design evolution	Design complexity
	Code generation	Cyclomatic Complexity, Code Quality, Other OO metrics
	Testing	RTF, % Test passing, Test productivity
Sprint Management	Sprint review	Team Velocity, Running Tested Features, Average Completion Rate
	Sprint monitoring	Sprint burndown index, Earned Value
	Sprint retrospective	Sprint Reliability, User satisfaction Index, Delivered Defect density
Release Management	Release monitoring	Release burndown index, Release Schedule Variance, SPI, Skill Index
	Release closure	Hard value delivered, Earned Business Value

5.4 Determination of logical relationship of factors

The relationship between the Business Objectives, Quality Objectives, process and sub-processes were then ascertained logically. A review by Subject Matter Experts (SMEs) was conducted to study and verify the logical relationship. Purpose of this review was to ensure that all the measures identified for quantitative modeling was relevant to the processes and sub-processes, and in turn they have a significant influence of the quality and business objectives of the organization.

Two types of relationship were studied between variables.

- Direct relationship,
- Inverse relationship.

It was observed that some of the sub-process metrics identified had a direct influence on the corresponding quality objective, while others were inversely related with the Quality Objective. The below table summarizes the logical relationship and its direction between the sub-process metrics and corresponding Quality Objective.

Table 3 Logical relationship between Quality objective and sub-process metrics

Sub-process Metric	Change in Sub-process Metric	Impact on Quality Objective (Delivered Defect Density)
Story DD	⇧	⇩
Design Complexity	⇧	⇧
Code Quality	⇧	⇩
RTF	⇧	⇩
Skill Index	⇧	⇩

5.5 Operational definition of metrics

The metrics that were selected for statistical validation had to be understood and defined further, so that there is consistency in data analysis and interpretation. For this purpose, the operational definition of these metrics were defined using a Metrics Definition Plan as in Table 4 below.

Table 4 Operational definition of metrics

Sub-process Metric	Operational Definition	Unit of Measurement	Data Source	Level of Control
Story DD	Number of peer review defects per story point of User Stories	Number of Defects	Review Tracker	Statistical
Design Complexity	Average Complexity of design modules on a scale of 1 to 5	Index	Agile Tracking tool	Statistical
Code Quality	% of rule violations in static code analyzers	Percentage	Code Analyzer	Statistical
RTF	% of features in a software which passes all their acceptance test cases, in a given instance	Percentage	Test Tracker	Statistical
Skill Index	Average primary skill rating on a scale of 1 to 5	Index	Skill Tracker	Operational

5.6 Statistical validation of relationship

Once the logical relationship between the variables were determined, the next step was to model and validate this relationship statistically. The purpose was to prove the existence of such a relationship using quantitative data, so that decisions could be taken on relevant factors impacting the goals rather than relying on assumptions and logical inference alone. Historical data for these measures considered were collected from closed projects over a period of past several years and this was organized as a Measurement Repository. The data was validated for relevance and quality prior to use for modelling.

Inorder to develop a model, dependent and independent variables had to be identified. Independent variables are considered as x's influencing the depending variable Y. In this case, DDD is the dependent variable Y, which is influenced by the x-factors which are Sprint DD, Design Complexity, Code Quality, RTF and Skill Index.

Once the variables are determined, suitable statistical technique for modelling had to be selected. This was done based on the table below (Stoddard et al. 2009).

Table 5 Selection of modelling technique

	Continuous (Y)	Discrete (Y)
Discrete (x)	ANOVA Dummy Variable Regression	Chi-Square Logistic Regression
Continuous (x)	Correlation Linear Regression	Logistic Regression

Since both the x's and Y are on a continuous scale, Multiple Linear Regression (MLR) was selected as a technique for statistical modelling. MLR is an approach to modeling the relationship between a dependent variable Y and one or more explanatory variables denoted as x.
The equation for MLR was modelled as below.
Delivered Defect Density = f(Story DD, Design Complexity, Code Quality, RTF, Skill Index)
Prior to performing regression, the relationship of each of the individual sub-process metric on Delivered Defect Density was studied independently using Scatter plots and Correlation. Once the relationships were determined independently, regression analysis was performed. The output of the regression was obtained as follows.
Delivered Defect Density = 0.756 + 0.0150 Story DD + 0.0132 Design Complexity - 0.00101 Code Quality - 0.00160 RTF - 0.00221 Skill Index

The above equation proved that Delivered Defect Density has a direct or positive relationship with Design Complexity and inverse or negative relationship with Story DD, Code Quality, RTF and Skill Index.

Certain parameters of Regression were studied to determine the goodness of the above equation. These were ANOVA p-value, R-sq adjusted value, Variance Inflation Factor, p-value of factors and normality of residuals. The resulting interpretation is summarized in the below table.

Table 6 Process performance baselines for sub-process metrics

Regression Analytic output	Observed Value	Inference
ANOVA p-value	0.01	Since the ANOVA p-value is less than 0.05, the Regression test is significant at 95% confidence interval
R-sq adjusted	91.3%	This implies that the factors considered explains more than 91% of the variation in output, i.e DDD.
Variance Inflation Factor (VIF)	Less than 5	A high value of VIFfor factors would indicate that the factors are inter-related to each other. In this case, since the VIF is less than 5, there is no relationship among factors, and each factor influences the output independently.
p-value of coefficients	Less than 0.05	Since the p-value is less than 0.05, the factors are significant to influence the output, at 95% confidence interval.
A-D testfor residuals	p>0.05	The residuals of the regression is normally distributed

This information helped the practitioners to precisely identify what are the contributing factors which will help them to achieve Delivered Defect Density. This implies that, from a Project Management perspective, inorder to reduce Delivered Defect Density, practitioners shall work towards reducing Design Complexity and increase Story DD, Code Quality, RTF and Skill Index. These sub-process metrics, also known as leading indicators are the ones which could be controlled during the course of execution of project.

5.7 Conversion to probabilistic model

In order to find out the probability that developed model will be accurate, Monte Carlo simulation technique was used. Monte Carlo methods are a class of computational algorithms that rely on repeated random sampling to compute their results. The input required for this simulation technique are the values of independent variables (x's), corresponding dependent variable (Y) and the relationship connecting the two.

The first step in developing the probabilistic model was to define assumptions on the distribution patterns of x-variables. A detailed study of the x-variables was done to understand its type of distribution based on historical data. A process performance baseline was created for each x-variable to understand its type and nature of distribution. CMMI defines Process Performance baseline as a documented characterization of process performance, which can include central tendency and variation.
The process performance baselines defined for the x-variables are as in Table 7.

Table 7 Process performance baselines for sub-process metrics

Variable	Mean	Standard Deviation	Distribution Type
Story DD	0.25	0.09	Normal
Design Complexity	3.06	0.702	Normal
Code Quality	81.78	5.15	Normal
RTF	76.97	3.64	Normal
Skill Index	3.48	0.93	Normal

Table7 hasparameters defined as assumptions in the Monte-Carlo simulation. The mean and standard deviation of the x-variables were modelled and assumed as inputs. The distribution type was defined as Normal for all the x-variables. Using this information and the regression equation, a relationship was drawn between the x-variables and Y, which is Delivered Defect Density. After this, the Delivered Defect Density was defined as output and a simulation run was performed with 10000 trials. The resultant output is as follows.

Table 8 Simulation output for Delivered Defect Density

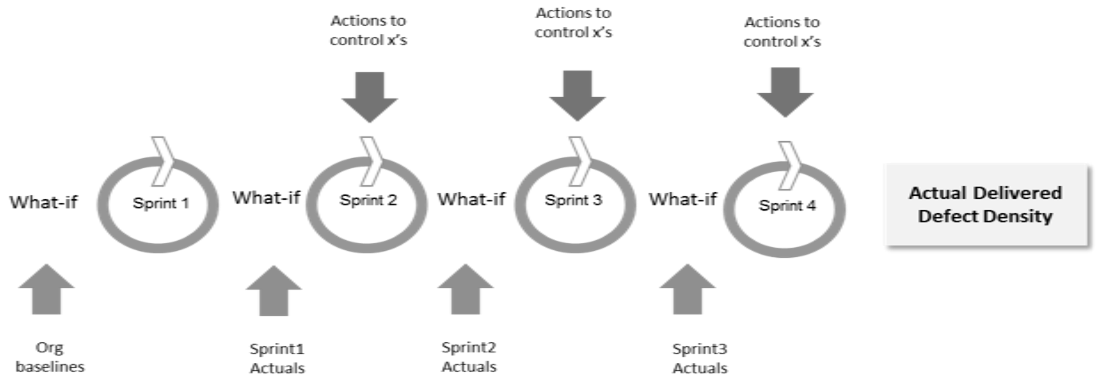
Name	Maximum	Minimum	Mean	Std. Dev.	Certainty @ 0.6
Delivered Defect Density	0.64	0.56	0.59	0.012	64.9%

The certainty value of Delivered Defect Density to meet a goal of less than 0.6 defects per 100 Story points as observed from the simulation result is 64.9%. This implied that, there is a 35.1% risk of not meeting the project goal of Delivered Defect Density, if the Agile project is following the same mean and standard variation performance of sub-process metrics as per the Organizational baselines. A better performance of sub-process metrics was required to meet the project goal with higher level of certainty. Actions were warranted during the project execution phase to improve the sub-process metrics performance.

VI. Usage of the prediction model during project execution

The prediction model developed out of the statistically significant metrics architecture was used in the Agile project during planning and execution phases. An agile project demands flexibility, transparency and improved efficiency. Better predictability of operations will help Scrum team to achieve this more systematically.

Fig. 3 In-process monitoring during execution of Agile project using the prediction model



During the planning stage of the Agile project, the Scrum team had to determine the optimum planning parameters to be used to run the project. Among the various parameters to be considered while planning, the following were considered as critical sub-process metrics to be given priority while planning. This was based on the derivation from statistically validated Metrics Architecture.

- Skill Index of the SCRUM Team
- Story Defect Densities
- Design Complexity
- Code Quality
- Running Tested Features

By giving priority to critical factors over other parameters gave Scrum team better focus on what is more important and significant to achieve the overall objective of the Scrum project, namely reducing Delivered defect density and reducing Schedule Variance as per overall release plan.

Table 9 What-if analysis based on planned and actual values during sprint execution

Subprocess Metric	Planned value	Actual value (x) – Sprint 1	Actual value (x) – Sprint 2	Actual value (x) – Sprint 3	Actual value (x) – Sprint 4
Story DD	0.25	0.43	0.22	0.21	0.18
Design Complexity	3.06	3.01	3.2	2.9	2.7
Code Quality	81.78	71.2	80.2	83.6	82.3
RTF	76.97	73.5	77.7	75.4	81.2
Skill Index	3.48	3.1	3.21	3.35	3.35
Predicted Delivered Defect Density	0.59	0.61	0.59	0.58	0.57

Out of the two goals, Delivered Defect Density was considered more important as the product quality was critical to client. During the planning phase, the SCRUM Master wanted to know whether he would be able to achieve the goals on Delivered Defect Density, considering the resources available. This was done by using the Organizational baseline values for the planning parameters in the prediction model as shown in Table 7. Further to this after each sprint, the actual values of the sub-process metrics were substituted in the prediction model equation to determine the predicted value of Delivered Defect Density. The result of this what-if analysis is summarized in Table 7 below.

Planning and in-process monitoring was performed before and after each sprint to ensure that the SCRUM Master has an ongoing control on the overall execution of the project. The actions taken for each sprint is explained below.

6.1 Planning phase

The Scrum Team had a four sprint project to be executed. During the planning phase, the Scrum Master set goals for the project. The overall goal for the project was set for Delivered Defect Density, which was a lagging indicator. Achieving this goal indicated the success of the Scrum project. However, Delivered Defect Density is not an in-process metric and cannot be measured during sprint execution. Inorder to achieve control on the Delivered Defect Density, Scrum Master had to focus on the sub-process metric, since those were directly correlated to the outcome measure of Delivered Defect Density. Hence, by monitoring the sub-process metrics of Story DD, Design Complexity, Code Quality, RTF and Skill Index during sprint execution, Scrum Master could have an impact on Delivered Defect Density.

With this awareness, sub-process metric goals were defined. These goals were set at the baseline values itself as defined in Table 5. However, the simulation exercise showed that if the sub-processes were operating as per baseline values, the probability of meeting the DDD goal is only around 65%. Being the first sprint, Scrum Master was willing to accept this risk and proceed with execution, with an intent to determine the actual values of sub-process performance after Sprint 1.

6.2 Execution phase

During this phase, the Scrum Master meticulously monitored the actual performance of sub-process metrics and made adjustments to the performance of these sub-process factors by applying rootcause analysis techniques and professional judgment. The advantage of using a prediction model for in-process monitoring was that the Scrum Master was certain on which all factors to be addressed, rather than trying to improve multiple factors without a focus on the end goal of Delivered Defect Density.

Fishbone and 5-Why analysis techniques were used to perform root cause analysis.The root cause analysis helped scrum master to identify the reasons for sub-process performance not within the expected limits. Timely actions were identified on such causes identified to improve the sub-process performance.

The actual performance of sub-process metrics were monitored after the completion of each sprint during the Sprint Retrospective meeting and analyzed.

6.3 Completion of Sprint 1

As per the prediction model, lower values of Story DD and Design complexity will lead to better Delivered Defect Density. At the end of Sprint 1, it was observed that the actual value of Story DD was higher than planned value, though Design Complexity was lesser. Code Quality, RTF and Skill Index should have higher values to reduce Delivered Defect Density. The actual values indicated that these metrics did not achieve the planned value. The predicted value of Delivered Defect Density using these values was 0.61, which gave an early warning signal to the Scrum team that the goal may not be achieved, going by the current performance of sub-processes.

At this stage, the Scrum Master had to initiate actions on all sub-process metrics except for Design Complexity. Rootcause analysis was performed to understand why these metrics did not meet the planned values and subsequently corrective actions were implemented. To reduce Story DD, specialized training was provided to the Business Analyst who defined the user stories. Inorder to improve Skill Index, it was decided to include a subject matter expert whose competency would raise the average value of the team performance. RTF and Code Quality were acted upon by improving the coding process and testing process respectively.

6.4 Completion of Sprint 2

At the end of Sprint 2, the actual values indicated that Story DD has reduced to 0.22, which was better that the planned value. Code Quality and RTF were now better than the planned values. However, Skill Index had to further improve and Design Complexity had to reduce to meet the overall goal of Delivered Defect Density. The predicted value of Delivered Defect Density using this actual performance came to 0.59, which met the goal. But the team had to constantly monitor the sub-process metrics to ensure that the overall goal is met and was under control. Actions were initiated on those metrics which did not meet the planned values.

6.5 Completion of Sprint 3

After the completion of Sprint 3, all the metrics except for RTF further improved. The predicted value of Delivered Defect Density using these values was 0.58, which was better than the project goal. The Scrum Master still had one more sprint to initiate actions on RTF which could further improve Delivered Defect Density. Rootcause analysis was done on how to improve RTF further and actions such as self-check of features by developers were implemented.

6.6 Completion of Sprint 4

The actual values of Sprint 4 indicated that the sub-process metrics were better than the planned values. Skill Index could not achieve the planned value being an operation measure and that resources could not be changed drastically after sprints. However, this was compensated by better performance of other sub-process metrics. The final predicted value of Delivered Defect Density was at 0.57, with the actual value of sub-process metrics from Sprint 4.

6.7 Project Closure

At the close of the project, the actual value of Delivered Defect Density was computed. This came to 0.581, which was much better than the project goal of 0.6. This was possible because of systematic actions initiated by the Scrum team and the end of each sprint with the help of quantitative early warning indicators of sub-processes, rather than waiting till the end of projects, were no actions would have been possible and effective.

VII. Conclusion

Agile software development gained increasing popularity in the industry, because of its adaptability to dynamics of software development and quicker execution time. The methodology of agile is flexible that it is compatible with other methodologies also. One of the most widely adopted model in software development is the Capability Maturity Model Integration (CMMI), which provides and evolutionary plateau for process improvement. In the maturity levels 4 and 5 of CMMI, the model urges software project managers to use process performance model, which will provide predictive control to the development process. When such a process performance model is combined with the inherent discipline of Agile, the result is a flexible and quantitative execution process for software.

This approach was studied by developing a process performance model based on historical data of project execution. The process level and sub-process level was modelled using regression and the resultant model was used for performing what-if analysis during the planning and execution phases of the project. The output of what-if analysis helped the scrum team to take meaningful decisions during the course of project execution and control the performance of the project.

The prediction model developed from historical data enabled the Scrum team in taking real time decisions which helped the project achieve its overall objective through focused actions on in-process metrics. This way, the project had all the advantages of agility, flexibility and lean processes, at the same time achieving predictability in operations through quantitative monitoring of statistically significant sub-process metrics

The model developed is based on the data gathered from one organization. Hence the coefficients of the factors in the equation may not be same for other organizations who wish to use the model. However, the approach for modelling, the process and sub-processes would be of interest to scrum masters wanting to predict Delivered Defect Density in their projects.

The major challenge in using such model comes from the source data used for development as it is organization specific. Practitioners shall determine the quality of source data and ensure its integrity before using for developing the model using the approach specified in this paper. As such, the approach does not have any significant limitations. However, it could so happen that variables available and the type of data available in certain organizations could be of different from the ones mentioned in this study. As an example, the data type of x's and Y could be different, in which case other modelling techniques such as ANOVA, Binary Logistic Regression, Logitetc to be used instead of Multiple Linear Regression. The selection of such technique is explained in Table 5.

The key strength of the model is that is having the backing of historical data and its usage is validated in an industrial project. The approach to modelling is flexible to the extent that a given organization can create its own metrics architecture to determine the processes and sub-process that will have an impact on the Critical to Quality objective, which in this case is Delivered Defect Density.Since not many papers describe the development of prediction models for Agile projects, the knowledge in this paper and its approach may be used for developing prediction models for other lagging metrics in Agile. This also gives a scope for future work, where researchers can study the possibility for prediction models for other critical to quality variables in Agile such as Velocity, Earned Business Value, Agility Index etc.

References

[1]. Abrahamsson, P., Moser, R., Pedrycz, W., Sillitti, A., &Succi, G. (2007, September). *Effort prediction in iterative software development processes--incremental versus global prediction models*. In Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on (pp. 344-353). IEEE.

[2]. Buglione, L., &Abran, A. (2007). *Improving Estimations in Agile Projects: issues and avenues.* In Proceedings of the 4th Software Measurement European Forum (SMEF 2007), Rome (Italy) (pp. 265-274).

[3]. Cavezza, D. G., Pietrantuono, R., & Russo, S. (2015). *Performance of Defect Prediction in Rapidly Evolving Software.*

[4]. Cohan, S., & Glazer, H. (2009). *An agile development team's quest for CMMI® maturity level 5*. In Agile Conference, 2009. AGILE'09. (pp. 201-206). IEEE.

[5]. Cohen, D., Lindvall, M., & Costa, P. (2003). *Agile software development*. DACS SOAR Report, (11).

[6]. Cohn, M. (2005). *Agile estimating and planning*. Pearson Education.

[7]. Cohn, Mike. (2010).*Succeeding with agile: software development using Scrum*. Upper Saddle River, NJ: Addison-Wesley.

[8]. Concas, G., Marchesi, M., Destefanis, G., &Tonelli, R. (2012). *An empirical study of software metrics for assessing the phases of an agile project*. International Journal of Software Engineering and Knowledge Engineering, 22(04), 525-548.

[9]. CMMI Product Team. (2010)*CMMI® for Development, Version 1.3*. CMU/SEI-2010-TR-033

[10]. Fenton, N., Neil, M., Marsh, W., Hearty, P., Marquez, D., Krause, P., & Mishra, R. (2007). *Predicting software defects in varying development lifecycles using Bayesian nets.* Information and Software Technology, 49(1), 32-43.

[11]. Fenton, Norman E., & Shari Lawrence Pfleeger. (1997). *Software metrics: a rigorous and practical approach*. 2. ed. London: Intern. Thomson Computer Press [u.a.].

[12]. Hearty, P., Fenton, N., Marquez, D., & Neil, M. (2009). *Predicting project velocity in XP using a learning dynamic Bayesian network model*. Software Engineering, IEEE Transactions on, 35(1), 124-137.

[13]. Jalote, Pankaj. (2005).*An integrated approach to software engineering*. 3rd ed. New York: Springer.

[14]. Kan, Stephen H. (2003). *Metrics and models in software quality engineering*. 2. ed. Harlow: Pearson Professional Education.

[15]. Levin, Richard I., & David S. Rubin. (1998).*Statistics for management*. 7th ed. Upper Saddle River, N.J.: Prentice Hall.

[16]. Li, L., & Leung, H. (2014). *Bayesian Prediction of Fault-Proneness of Agile-Developed Object-Oriented System. In Enterprise Information Systems* (pp. 209-225). Springer International Publishing.

[17]. McMahon, Paul E. (2011). *Integrating CMMI and agile development: case studies and proven techniques for faster performance improvement*. Upper Saddle River, NJ: Addison-Wesley.

[18]. Nasir, M. (2006, June). *A survey of software estimation techniques and project planning practices*. In Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2006. SNPD 2006. Seventh ACIS International Conference on (pp. 305-310). IEEE.

[19]. Regan, Gerard. (2009).*A practical approach to software quality*. New York: Springer, 2002

[20]. Runeson, P., &Höst, M. (2009). *Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering*, 14(2), 131-164.

[21]. Salo, Outi. &Abrahamsson, Pekka. (2007), *An iterative improvement process for agile software development,* Wiley.

[22]. Silva, F. S., Soares, F. S. F., Peres, A. L., de Azevedo, I. M., Vasconcelos, A. P. L., Kamei, F. K., & de LemosMeira, S. R. (2015). *Using CMMI together with agile software development: A systematic review*. Information and Software Technology, 58, 20-43.

[23]. Singh, P., &Verma, S. (2014). *An Efficient Software Fault Prediction Model using Cluster Based Classification*. International Journal of Applied Information Systems (IJAIS), 7(3), 35-41

[24]. Stoddard et al., A mini tutorial for building CMMI process performance models, Software Engineering Institute.

[25]. Sutherland, Jeff. (2013).*Scrum: A Revolutionary Approach to Building Teams, Beating Deadlines, and Boosting Productivity*, Random House.

[26]. Sutherland, J., Jakobsen, C. R., & Johnson, K. (2008, January). *Scrum and cmmi level 5: The magic potion for code warriors*. In Hawaii International Conference on System Sciences, Proceedings of the 41st Annual (pp. 466-466). IEEE.

[27]. Washizaki, H., Honda, K., & Fukazawa, Y. (2015, August). *Predicting Release Time for Open Source Software Based on the Generalized Software Reliability Model*. In Agile Conference (AGILE), 2015 (pp. 76-81). IEEE