Strong Agile Metrics: Mining Log Data to Determine Predictive Power of Software Metrics for Continuous Delivery Teams

Hennie Huijgens
Delft University of Technology
Delft, The Netherlands
h.k.m.huijgens@tudelft.nl

Robert Lamping
ING Bank and CGI
Amsterdam, The Netherlands
robert.lamping@ing.com

Dick Stevens
ING Bank
Amsterdam, The Netherlands
dick.stevens@ing.com

Hartger Rothengatter
ING Bank
Amsterdam, The Netherlands
hartger.rothengatter@ing.com

Georgios Gousios
Delft University of Technology
Delft, The Netherlands
g.gousios@tudelft.nl

Daniele Romano
ING Bank
Amsterdam, The Netherlands
daniele.romano@ing.com

ABSTRACT

ING Bank, a large Netherlands-based internationally operating bank, implemented a fully automated continuous delivery pipeline for its software engineering activities in more than 300 teams, that perform more than 2500 deployments to production each month on more than 750 different applications. Our objective is to examine how strong metrics for agile (Scrum) DevOps teams can be set in an iterative fashion. We perform an exploratory case study that focuses on the classification based on predictive power of software metrics, in which we analyze log data derived from two initial sources within this pipeline. We analyzed a subset of 16 metrics from 59 squads. We identified two lagging metrics and assessed four leading metrics to be strong.

CCS CONCEPTS

• **General and reference** → Cross-computing tools and techniques → Metrics

KEYWORDS

Software Economics; Agile Metrics, Scrum, Continuous Delivery, Prediction Modelling, DevOps, Data Mining, Software Analytics.

ACM Reference format:

Hennie Huijgens, Robert Lamping, Dick Stevens, Hartger Rothengatter, Georgios Gousios and Daniele Romano. 2017. Strong Agile Metrics: Mining of Log Data to Determine Predictive Power of Software Metrics for Continuous Delivery Teams. In *ACM Proceedings of the 11th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 2017 (ESEC/FSE'17)*, 6 pages.
https://doi.org/10.1145/3106237.3117779

1 INTRODUCTION

In order to further speed up application deployments, reduce risks of failure, and deliver applications rapid, repeatable, and reliable, ING Bank, a large Netherlands-based internationally operating bank, introduced Continuous Delivery as a Service (CDaaS) and DevOps. ING's continuous delivery cycle includes code, build, deploy, test and release of all software engineering activities, and supports more than 300 software delivery teams - *squads* in ING terminology - that operate primarily on a Linux or a Windows platform. The mindset behind CDaaS is to go to production as fast as possible while maintaining or improving quality, so teams get fast feedback, and know they are on the right track. The continuous delivery pipeline is at the core of a transition that is ongoing within ING towards DevOps.

ING's continuous delivery cycle is now implemented. Two matured CDaaS squads support a huge variety of squads in different business domains and software technologies. And all teams work in an agile (Scrum) way. Yet, now a need is felt to develop a monitor and control capability that fits the different squads in the organization. In this process of setting up a software metrics approach, the company wants to prevent from looking at metrics in isolation, treating a metric (making alterations just to improve a metric), or one trick metrics on the one hand against metrics galore on the other [1]. In line with the fully automated building, testing, and deployment of software in the continuous delivery pipeline three important requirements are applicable. (1) The monitor and control capability is implemented as a fully automated and iterative process, (2) it supports the different squads where possible in improving their deliveries, and (3) the capability must have a high degree of predictive power.

In this paper we describe the initial process to determine strong software metrics - being metrics with high predictive power - in order to be able to support a highly effective monitor and control capability within ING Bank. Because the process to classify metrics will be iterative – adding new metrics to the procedure will lead to a fluid and ongoing redefinition of the concept of 'strong' – we use the terminology of strong *agile* metrics. We perform our

analysis as an exploratory case study, based on two initial data sources from ING's software engineering domain: the Backlog Management (BLM) discipline, and the Continuous Delivery (CDaaS) discipline. Although data is collected on a per squad level, strong metrics are analyzed and reported at an aggregated (average) level and not as such on a per squad level.
Our objectives are to explore whether data mining techniques can help to define such strong agile metrics. In a follow-up case study based at this exploratory study we aim to develop a 1-to-5 star-rating for software metrics, that can be used throughout ING's software organization as a support tool when preparing dashboards and other visualizations. The aspect of automation and building a performance dashboard itself is not within the scope of this paper. We address the following research question:

RQ *How can we set strong metrics for agile (Scrum) DevOps teams in an iterative fashion?*

As our key contribution we evaluate how to determine the predictive power of software metrics from log data of a toolset of more than 300 different development teams in a large software company, that perform more than 2500 deployments to production each month on more than 750 different applications.

1.1 Background

ING Bank implemented Continuous Delivery based on the model described by Humble and Farley [2], where the CD pipeline is proposed as a deployment pipeline for the whole value stream of software development. ING Bank set up two different pipelines based on the main technologies Linux and Windows. Its' main goal is to support squads in maximizing the benefits of shared use of tools.
In this paper we focus at the CDaaS Linux pipeline. It provides developers with a complete set of standard tools that are supported by a Linux CDaaS squad and available to all squads within the bank. The pipeline fully automates the software release process for Linux based applications. It contains several specialized tools which are connected to each other, such as GitLab (code), Jenkins, SonarQube, and QWasp and Artifactory (build), Nolio (deploy), iTested (test), and iValidate (release).
The final goal of our analysis (yet, out-of-scope for this exploratory study) is to examine 'good' deliveries (being better than average within the scope of a squad) and 'bad' deliveries (being worse than average), as described in previous work on success and failure in software engineering projects [3]. By doing so we expect to identify success factors that help squads to create better deliveries in future releases, and failure factors that help squads to prevent from 'bad' deliveries. However, due to the limited number of initial data sets this final goal is out-of-scope for this paper.

2 RESEARCH APPROACH

We conduct our research as an exploratory case study and we assess the predictive power of software delivery metrics, in which we use data derived from two initial sources. The first data source is log data from ServiceNow, the Backlog Management (BLM) tool that is used by most of the software development squads. The second data source that is used in this study is deployment log data from Nolio.
For our exploratory study we define a limited set of software metrics focused at a delivery scope (e.g. epics, user stories) as a minimum viable product (MVP): a product with just enough features to gather validated learning about the product and its continued development [4]. Based on the initial two data sets we designed four steps to define the MVP: (1) Scope definition, (2) Collect metrics, (3) Analyze for correlations, and (4) Determine prediction power of metrics.

2.1 Scope Definition

In this initial step we define the scope of the software metrics to be analyzed. Within the scope of this exploratory case study we limit the scope of data to log data of two data sources, derived from ServiceNow and Nolio. We combine a snapshot of ServiceNow data from 370 squads with CDaaS data of the deployment file from 101 squads. No time series are included in both datasets. We focus our analysis on squads; teams that deliver sets of user stories, combined in epics, to users within ING Bank or to ING's customers. Due to various missing data in both datasets, linking both datasets result in 59 squads for which all data are present. This is our analysis set.

2.2 Define and Collect Metrics

In a number of meetings with stakeholders definitions of metrics and hypothesis are classified. A draft metrics framework, an inventory of existing dashboards within the company, and an informal literature review are used as a baseline. Our intention is not to come up with a finalized inventory, but instead to set up and maintain a backlog of prioritized software metrics. In order to set up a repeatable future-proof procedure that is fully automated in-the-end, we do not analyze raw data itself. Instead we structure the log data upfront in a dedicated data warehouse with an automated feed from BLM and CDaaS in the background. Within the scope of this study a limited set of one daily download is used for further analysis. In this initial data set observations with missing values are beforehand removed.

2.3 Lagging and Leading Metrics

In this paper we distinguish two types of metrics: lagging and leading metrics. Lagging metrics are output oriented and cannot be directly influenced. A lagging indicator gives a signal *after* the trend or reversal occurs. These metrics are perceived as key indicators for high performing teams. Leading metrics are input oriented and easy to influence. A leading indicator gives a signal *before* the trend or reversal occurs [5].

Table 1. Metrics Descriptions and Descriptive Statistics.

Metric*	Source	n	Type	Skewness	Kurtosis	Min	1st Q	Median	Mean	3rd Q	Max
cdaas_cycletime_tst1_prd [Lagging]	CDaaS	59	Days	0.56	-0.19	0.41	11.21	17.22	17.96	25.18	46.52
cdaas_mtb_prd_lst90days	CDaaS	59	Days	0.73	0.96	2.98	23.02	34.16	36.77	49.22	104.90
cdaas_numberofdeploymentsprd	CDaaS	59	Number	4.12	18.77	1.00	3.00	7.00	16.73	17.00	194.00
sprint_averageleadtime	BLM	59	Days	5.19	45.31	-6.50	22.15	31.17	37.65	44.25	370.50
sprint_averagepointsperstory	BLM	59	Ratio	3.37	13.34	0.03	0.08	0.11	0.17	0.17	1.00
sprint_duration	BLM	59	Days	3.14	21.08	5.00	13.00	13.00	14.79	14.00	55.00
sprint_numberofchangemembers	BLM	59	Number	3.76	30.14	0.00	7.00	9.00	9.54	11.00	51.00
sprint_numberofepicslastsprint	BLM	59	Number	1.51	3.77	1.00	7.00	11.00	12.00	16.00	48.00
sprint_numberofsquadmembers	BLM	59	Number	3.56	26.93	2.00	7.00	10.00	10.01	12.00	51.00
sprint_plannedpointscompletionratio [Lagging]	BLM	59	Ratio	-0.62	-0.23	0.00	0.05	0.67	0.64	0.83	1.00
sprint_plannedstoriescompletionratio [Lagging]	BLM	59	Ratio	-0.53	-0.55	0.00	0.44	0.67	0.63	0.86	1.00
sprint_pointscompletionratio	BLM	59	Ratio	0.29	1.75	0.02	0.50	0.72	0.70	0.90	2.18
sprint_remainingtimeratio	BLM	59	Ratio	5.28	29.28	0.00	0.00	0.00	0.03	0.00	1.00
sprint_scopegrowth	BLM	59	Number	16.65	283.94	-112.00	0.00	0.00	2.82	0.00	919.00
sprint_unplannedexistingpointscompletionratio	BLM	59	Ratio	1.53	1.85	0.00	0.00	0.07	0.17	0.27	1.00
sprint_unplannednewpointscompletionratio	BLM	59	Ratio	3.95	18.55	0.00	0.00	0.00	0.07	0.07	1.00

Backlog Management (BLM) log data from ServiceNow and CDaaS log data from Nolio. *A more detailed description of each metric, including extended descriptive statistics is included in the Technical Report. In order to assess distribution we included Skewness and Kurtosis of each individual metric. Lagging metrics are indicated with the text [Lagging] behind their names in the first column of the table above.

2.4 Analysis

We define and collect metrics from two initial data sources, classify metrics, and specify whether metrics are lagging or leading. we examine descriptive statistics and we analyze the total set of metrics from the two initial data sources for statistical correlation. To understand any relations between individual metrics we perform linear regression. For visualization purposes we prepare a correlation matrix that plots positive and negative correlations between all individual metrics; this matrix is not included in this paper, it is to be found in a Technical Report [6].

2.5 Determining Predictive Power of Metrics

To classify software metrics based on their prediction strength with regard to the performance of releases delivered by agile (Scrum) DevOps teams, we use a search algorithm to find the best model, based on forward selection, backward elimination, and stepwise regression. We define strong metrics as leading metrics with strong correlation(s) to lagging metrics in the data set. Although in our exploratory case study we do this in a manual way, we plan for automated methods to identify predictor variables in a future solution. Automated methods are useful when the number of explanatory variables is large, as in our case, and when it is not feasible to fit all possible models. For this purpose we built a new model in R, based on the existing Corrplot package, in which we visualize the outcomes of the multiple linear regression in a Leading Lagging Matrix (see Figure 1).

3 RESULTS

In Table 1 we inventory descriptive statistics of the BLM metrics and CDaaS metrics in scope. To test whether the data in our datasets is normally distributed or not, we used a skewness and kurtosis test. As values for skewness and kurtosis between -2 and +2 are considered acceptable in order to prove normal univariate distribution [7], we assess the majority of metrics in both subsets to be not normally distributed.

3.1 Analysis of Predicting Variables

We perform pairwise correlation in order to find any relations between individual metrics. Because a small majority of the metrics are assessed to be not normally distributed we use the method Spearman. A visualization in the form of a correlation matrix is included in the Technical Report [6]. Variables that have no significant correlation in a 1-to-1 analysis, may act differently in multiple regression. Remaining time ratio is an example of such a variable.

We are modelling lagging indicators in terms of leading variables. In Figure 1 the impact of leading metrics on the set of lagging metrics is visualized in a *Leading Lagging Matrix*. The figure shows for each lagging variable (horizontal axis) what the impact is of each leading variable (vertical axis). The color of each circle indicates whether the impact is positive (blue) or negative (red). The size of a circle indicates whether impact is strong (large impact) or weak (small impact). Same size circles on the same row do not mean they are equal: for each lagging variable the leading variables are calculated using multiple linear regression, subsequently all multiple linear regression coefficients are rescaled to a

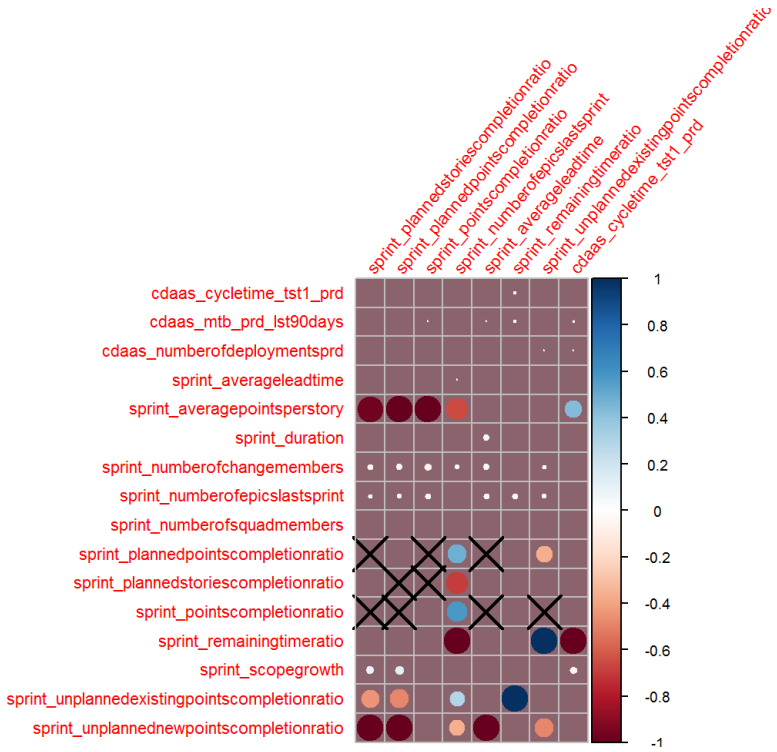

Figure 1: *Leading Lagging Matrix* showing the impact of leading metrics (vertical axis) on lagging metrics (horizontal axis).

scale of -1 to 1. Empty squares indicate a coefficient of zero. Crossed out variables are excluded from the lagging model in order to avoid collinearity (independent variables that are highly correlated).

In this first exploratory study we used a pragmatic approach to determine which variables are leading or lagging in our model. We argued that in this first analysis three metrics are assessed to be lagging. (1) *Planned stories completion ratio*; the number of planned stories that were completed in a sprint divided by the number of planned stories. (2) *Planned points completion ratio*; the number of completed planned story points divided by the number of planned story points. (3) *CDaaS cycle time*; the mean time from first test deployment after last production has been done until the next production deployment for all applications of a squad.

The choice for these three lagging metrics is mainly driven by the assumption that they are typically output related and cannot easily be planned upfront. For analysis purposes we included a reference set of five other metrics on the x-axis of the matrix, although these were not assumed to be lagging.

3.4 Key Findings

When examining the *Leading Lagging Matrix* as depicted in Figure 1, we observe the following:

Observation 1. Higher average story points (5th row) have a negative impact on the planned completion ratio (either points or stories) and on the total completion ratio. Besides that, higher average story points lead to a longer CDaaS cycle time (from test to production).

Observation 2. If the planned points completion ratio (10th row) goes up, also the number of epics increases. At the same time unplanned backlog work (unplanned existing) decreases.

Observation 3. The planned stories completion ratio (11th row) shows an opposite effect: if this variable goes up, the number of epics decreases. The different behavior of both metrics need to be examined further in follow-up research.

Observation 4. If the points completion ratio (12th row) increases, also the number of epics goes up, an effect that is similar to Observation 2.

Observation 5. If the remaining time ratio (13th row) goes up, the number of epics and the CDaaS cycle time decreases. Furthermore, if there is time left after all planned work is done, we see that squads pick up backlog work.

Observation 6. When unplanned existing work (15th row) (e.g. picked up from backlog) pops up this has a moderate negative impact on the planned work and the ability to pick up backlog work. In this case the number of epics increases moderately, while remaining time ratio increases strongly.

Observation 7. Average lead time is negatively impacted by unplanned new work (e.g. incidents) (bottom row), possibly because it delays planned work that was already started in the current sprint or earlier and that cannot finish in the planned time. Average lead time is not tightly linked to a sprint, but more to a user story.
Observation 8. No significant impact is caused in our models by the three CDaaS variables, and the remaining BLM variables.

4 DISCUSSION

As explained in paragraph 3.3 we assumed upfront three metrics to be lagging: planned stories completion ratio, planned points completion ratio, and CDaaS cycle time. The first two metrics are both about the completion of planned work, so to say the predictability of delivery of squads. We prefer planned stories completion ratio because it has the advantage that scope growth (measured in story points) has no influence on the ratio value. Leading variables for this metric are unplanned new points completion ratio, unplanned existing points completion ratio, and average points per story. With regard to this lagging metric we assess these three leading variables as 'strong metrics'.
The third lagging variable, CDaaS cycle time, can be influenced by the leading variables remaining time ratio and average points per story. With regard to this lagging metric we assess both leading variables as 'strong metrics'.
Besides these three variables we observe that also the number of epics (4th column from the left) can be influenced strongly by planned points completion ratio, points completion ratio, remaining time ratio, unplanned existing points completion ratio, and unplanned new points completion ratio. However, because we assume that this variable can be easily planned upfront we do not assess it as a lagging variable as such.
From an overall point of view we argue that average points per story is influencing both preferred lagging metrics, and due to that is to be assessed as the most powerful metric in the actual subset.

4.1 Implications

Our model, based on an initial subset of BLM and CDaaS data, indicates that squads can improve their planned stories completion ratio and reduce their CDaaS cycle time by slicing their deliverables in smaller user stories. Squads can reduce their CDaaS cycle time by keeping open space in their sprint planning (e.g. increasing their remaining time ratio). Finally, by reducing unexpected unplanned work squads can increase their predictability of delivery (e.g. planned stories completion ratio). These implications are also identified by Humble and Farley [2] as key measures for implementing Continuous Delivery and DevOps. Our research herewith substantiated these measures based on statistical analysis of ING's Continuous Delivery cycles.

4.2 Threats to Validity

With regard to construct validity we are aware that the use of story points for comparison purposes over squads might be spurious in a way. However, to prevent from differences in ranges used by different squads we calculated all measurements with story points involved to indexes. A second threat that we take into account is the way we picked metrics to be included in the study and the choice of lagging versus leading metrics. In our actual approach we inventoried metrics in related work and existing dashboards within ING bank, and mapped these on a metrics framework that we based on previous work [8]. We realize that some systemic bias might play a role here and are looking for ways to mitigate this in a more mature approach. A third threat with regard to construct validity is that we did not exclude outliers from our research dataset. Although we realize that this is important for a future approach we did not include this in this exploratory study. Finally we recognize challenges with data quality as a threat to validity. Especially to link BLM data on squads and applications with the CDaaS dataset was a blockade in some cases.
A threat to internal validity that we acknowledge is the fact that 'fishing for p -values' might hold a risk that some of the correlations we find are a coincidence. Although we acknowledge the fact that in a future approach corrections (e.g. Benjamini–Hochberg [9]) are to be implemented, we did not apply any of such corrections in this initial and exploratory study.
Concerning external validity we argue that results from our study are not to be generalized to other companies than ING Bank. We assume that different companies have their own specific metrics patterns. We expect that our approach to derive strong metrics by mining log data from software delivery pipeline tools can be successfully used in practice by other companies too. To encourage reuse and further improvement of our approach, we share a subset of the R-code developed by us in the Technical Report [6].

5 RELATED WORK

Where from the 80s onwards software companies used to follow a software process improvement (SPI) approach with varying success [10] [11], since the start of this millennium an industry-wide transformation towards agile development methods is obvious. Although several studies are performed on the success and failure of agile methods [12] [13] [3], a clear definition of success is difficult to find. A number of researchers and practitioners come up with terms such as hyperproductivity [14] [15] [16], usually with a limited focus on best practices in a Scrum environment, described as for example 'the most productive Java projects ever documented' [14]. To define definitions for hyperproductivity and accompanying software metrics we based our research besides the above on metrics for continuous delivery as mentioned by Humble and Farley [2] and on Puppet's State of DevOps Report [5].
The effect on strong positive correlations between *Project Size*, *Project Cost*, and *Number of Defects* is known from related work [3] [17] [18]. Also the effect of project size as a risk factor is described earlier. Smaller projects tend to have lower cancellation rates [19]. Smaller projects tend to perform better in terms of quality, being on budget, and being on schedule [19] [20]. Project size is found to be an important risk factor for success [21] [22] [23].

In previous work we found strong effects by comparing quantitative metrics such as cost, duration, number of defects, and size with qualitative metrics like stakeholder satisfaction and perceived value [8]. A recent guest editorial by Mäntylä et al. [24] mentions that, although "many papers investigate success and failure of software projects from diverse perspectives, leading to a myriad of antecedents, causes, correlates, factors and predictors of success and failure", a solid, empirically grounded body of evidence enabling actionable practices for increasing success and avoiding failure in software projects is not yet found.
Premrai et al. [25] investigated how software project productivity had changed over time, finding that an improving trend was measured, however less marked since 1990. The trend varied over companies and business sectors, a finding that matches the result of our previous research with regard to differentiation over business domains [3].

6 CONCLUSIONS AND FUTURE WORK

We analyzed a dataset built from BLM and CDaaS data from ING Bank in order to identify strong metrics; metrics with high predictive power towards a subset of lagging variables. We found two lagging metrics and four leading metrics that are assessed to be strong.
In future research we plan to extend the number of data sources - e.g. availability, squad decomposition, business process performance, customer experience, incidents - and due to that the number of variables in our model. We also plan to examine how lagging metrics and strong leading metrics can be identified in an automated procedure. Furthermore we intend to automatically set targets on the strong agile metrics, based on the performance of high performing teams within ING Bank. Our final objective is to use our findings to define relevant lagging metrics and related strong leading metrics to enable squads and management to steer on performance by delivering strong agile metric dashboards.

ACKNOWLEDGMENTS

Our sincere thanks to ING Bank for offering us the opportunity and the confidence to perform research in their challenging software development team environment.

REFERENCES

[1] E. Bouwers, J. Visser en A. van Deursen, „Getting What You Measure," *Communications of the ACM*, vol. 55, nr. 7, pp. 54-59, 2012.

[2] J. Humble en D. Farley, Continuous Delivery, reliable software releases through build, test and deployment automation, Addison-Wesley, 2010.

[3] H. Huijgens, R. van Solingen en A. van Deursen, „How to build a good practice software project portfolio?," in *ACM Companion Proceedings of the 36th International Conference on Software Engineering (ICSE)*, 2014.

[4] J. Münch, F. Fagerholm, P. Johnson, J. Pirttilahti, J. Torkkel en J. Jäarvinen, „Creating minimum viable products in industry-academia collaborations," in *Lean Enterprise Software and Systems*, Springer Berlin Heidelberg, 2013, pp. 137-151.

[5] „State of DevOps Report," Puppet, 2016.

[6] H. Huijgens, R. Lamping, D. Stevens, H. Rothengatter en G. Gousios, „Strong Agile Metrics - Technical Report TUD-SERG-2017-010," Delft University of Technology, 2017.

[7] D. George en M. Mallery, SPSS for Windows Step by Step: A Simple Guide and Reference, 17.0 update (10a ed.) red., Boston: Pearson., 2010.

[8] H. Huijgens, A. van Deursen en R. van Solingen, „The Effects of Perceived Value and Stakeholder Satisfaction on Software Project Impact," *Information and Software Technology*, 2017.

[9] W. Hopkins, A new view of statistics, Internet Society for Sport Science, 2000.

[10] T. Hall, A. Rainer en N. Baddoo, „Implementing Software Process Improvement: An Empirical Study," *Software Process Improvement and Practice*, vol. 7, pp. 3-15, 2002.

[11] T. Dyba, „An Empirical Investigation of the Key Factors for Success in Software Process Improvement," *IEEE Transactions on Software Engineering*, vol. 31, nr. 5, pp. 410-424, 2005.

[12] T. Chow en D.-B. Cao, „A survey study of critical success factors in agile software projects," *The Journal of Systems and Software*, vol. 81, pp. 961-971, 2008.

[13] S. C. Misra, V. Kumar en U. Kumar, „Identifying some important success factors in adopting agile software development practices," *The Journal of Systems and Software*, vol. 82, pp. 1869-1890, 2009.

[14] J. Sutherland, A. Viktorov, J. Blount en N. Puntikov, „Distributed Scrum: Agile project management with outsourced development teams," in *IEEE 40th Annual Hawaii International Conference on System Sciences (HICSS)*, 2007.

[15] M. Beedle , M. Devos, Y. Sharon, K. Schwaber en J. Sutherland, „SCRUM: An extension Pattern Language for Hyperproductive Software Development," in *Pattern Languages of Program Design*, Addison-Wesley, 2000, pp. 637-651.

[16] J. Sutherland, N. Harrison en J. Riddle, „Teams that Finish Early Accelerate Faster: A Pattern Language for High Performing Scrum Teams," in *47th Hawaii International Conference on System Science*, 2014.

[17] K. El Emam en A. Günes Koru, „A replicated survey of IT software project failures," *IEEE software*, vol. 25, nr. 5, pp. 84-90, 2008.

[18] M. Bhardwaj en A. Rana, „Key Software Metrics and its Impact on each other for Software Development Projects," *ACM SIGSOFT Software Engineering Notes*, vol. 41, nr. 1, pp. 1-4, 2016.

[19] D. Rubinstein, „Standish group report: There's less development chaos today," *Software Development Times*, vol. 1, 2007.

[20] R. Sonnekus en L. Labuschagne, „Establishing the Relationship between IT Project Management Maturity and IT Project Success in a South African Context," Proc. 2004," *PMSA Global Knowledge Conf., Project Management South Africa*, pp. 183-192, 2004.

[21] H. Barki, S. Rivard en J. Talbot, „Toward an assessment of software development risk," *Journal of Management Information Systems*, vol. 10, pp. 203-223, 1993.

[22] J. Jiang en G. Klein, „Software development risks to project effectiveness," *Journal of Systems and Software*, vol. 52, nr. 1, pp. 3-10, 2000.

[23] R. Schmidt, K. Lyytinen, P. Cule en M. Keil, „Identifying software project risks: An international Delphi study," *Journal of management information systems*, vol. 17, nr. 4, pp. 5-36, 2001.

[24] M. V. Mäntylä, M. Jørgensen, P. Ralph en H. Erdogmus, „Guest editorial for special section on success and failure in software engineering," *Empirical Software Engineering*, vol. April, pp. 1-17, 2017.

[25] R. Premrai, M. Shepperd, B. Kitchenham en P. Forselius, „An Empirical Analysis of Software Productivity Over Time," in *IEEE International Symposium Software Metrics*, Como, Italy, 2005.