# Software Development Metrics Prediction Using Time Series Methods

Michał Choraś[1, 3], Rafał Kozik[1, 3], Marek Pawlicki[1, 3], Witold Hołubowicz[3], and Xavier Franch[2]

[1]ITTI Sp. z o.o., Poland
[2]GESSI, UPC, Barcelona, Spain
[3]UTP Universiy of Science and Technology, Bydgoszcz, Poland

**Abstract.** The software development process is an intricate task, with the growing complexity of software solutions and inflating code-line count being part of the reason for the fall of software code coherence and readability thus being one of the causes for software faults and it's declining quality. Debugging software during development is significantly less expensive than attempting damage control after the software's release. An automated quality-related analysis of developed code, which includes code analysis and correlation of development data like an ideal solution. In this paper the ability to predict software faults and software quality is scrutinized. Hereby we investigate four models that can be used to analyze time-based data series for prediction of trends observed in the software development process are investigated. Those models are Exponential Smoothing, the Holt-Winters Model, Autoregressive Integrated Moving Average (ARIMA) and Recurrent Neural Networks (RNN). Time-series analysis methods prove a good fit for software related data prediction. Such methods and tools can lend a helping hand for Product Owners in their daily decision-making process as related to e.g. assignment of tasks, time predictions, bugs predictions, time to release etc. Results of the research are presented.

**Keywords:** Software Engineering, Software Development, Prediction, Metrics, Time Series

## 1 Introduction and Context

Nowadays, increasing interdependences among software components can be observed in the IT domain. Every industry relying on software solutions is impacted by software development challenges, and with the domain being a growing one, it has a fair share of such issues. Some of them include optimization of the software code development process, minimization of the risk of software failures (quality assurance, code testing/debugging) and the software solutions security. Those aspects - to be addressed during the code development - are critical for businesses, service providers, end customers, and thus for society as a whole.

Choras, M. [et al.]. Software development metrics prediction using time series methods. A: International Conference on Computer Information Systems and Industrial Management Applications. "Computer Information Systems and Industrial Management, 18th International Conference, CISIM 2019: Belgrade, Serbia, September 19–21, 2019: proceedings". Berlín: Springer, 2019, p. 311-323.
The final authenticated version is available online at https://doi.org/10.1007/978-3-030-28957-7_26

A diverse range of interests by different stakeholders is involved in the issues related to code quality and code development optimization, making it a complex, intricate, multifaceted issue. Some of those facets include:

- Growing complexity of software solutions in terms of code lines volume software solutions become increasingly more sophisticated, with the total number of code lines inflating several hundred percent with each subsequent release. Desktop applications, mobile OS or web-based solutions contain millions of lines of code, with e.g. all Google web services exceeding 2 billion lines [11]. The abovementioned trend of growing source code volume poses serious challenges in analysis, processing and reasoning from such data. It becomes a big data exploration problem, especially in the case of near real-time analysis and prediction.
- The emerging trend of including software components in IoT assets, microdevices, smart home systems, smart city systems, etc. adding to that the spike in popularity of cloud-based interconnection of assets.
- The software quality dictates the reputation, market position and competitiveness of the product vendors. The estimates state that a software bug can cause a fall in the product stock price by an average of 4% to 6% (for companies experiencing multiple software failures). This generates almost 3 billion dollars of market losses ( QASymphony 2016, [23] ). Moreover, substandard quality of code impacts the overall cost of the software development, deployment and maintenance [12]. According to QASymphony data, the process of debugging software during its design phase costs 4 to 5 times less than that of fixing bugs after its release.
- The growing cost of software testing and debugging - although the process of debugging software during its design phase costs significantly less than fixing bugs after its release, it is still non-trivial task that consumes a significant part of budgets and companies effort. It might have less of an impact for bulky companies and software houses. However SMEs and small entities providing software solutions, often operating on constrained budgets and finite resources, are becoming increasingly more focused on techniques allowing automation and adequacy of the testing process, to be competitive in relation to big players on the market [9] . The costs of quality assurance and testing in IT are growing from each year. Currently, IT organizations spend approximately 1/3 of their budgets on quality assurance with the trend of inflating this value to approx. 40% in the next three years [20], [5].
- Raising concerns about the software (or solutions including software components) security - software flaws and bugs can impact not only the usability, functional value and user experience, but also the security of users. This is due to the fact that bugs in the design or implementation phase can be exploited by cyber-criminals. According to the article presented by [1] , consequences of cyber attacks cost the British companies about 18 billion per year in terms of lost revenues.

Based on the above considerations, we state that there is a clear need for decision makers (project managers, product owners, development team leaders)

in the software engineering process for an automated quality-related analysis of developed code. This should include code analysis and correlation of development data derived from available tools with management-related issues such as effort allocation, scheduling, workload accumulation and others.

Moreover, we believe that accurate prediction of such indicators can be beneficial in the software development process optimization and can raise the overall quality of software products and reduce further debugging and maintenance costs. In reality, one of the typical needs and challenges for product owners in their daily duties is to plan and predict time estimates for the current and future tasks in backlogs. Of course, those decisions depend on predicting other fundamental aspects like estimated foreseen number of bugs, failed tests as well as other software related and process metrics.

In the previous work of the H2020 Q-Rapids project consortium ( [22], [10], [15], [17], [14]), the concept of quality-aware decision making based on key strategic indicators is proposed. The overall goal of the project is to support strategic decision-making processes by providing strategic indicators in the context of quality requirements in agile and rapid software development. For the purposes of the project, a strategic indicator is defined as a specific aspect that a software development company has to consider as crucial for the decision making process during software development. Aspects such as e.g. time-to-market, maintenance cost, customer satisfaction, etc. can be considered as strategic indicators depending on the context. These strategic indicators are built on top of the measurements and factors calculated on the basis of the software development related data, stored in the management tools such as GitLab or SonarQube.

While in the previous work we focused on the data gathering and processing it towards quality metrics, indicators and requirements generation [21] [7], we hereby target the prediction of the software related metrics.

The paper is structured as follows: in Section 2 an overview of time series methods and models is provided. In Section 3 the sample prediction results are presented. Conclusions are drawn thereafter.

## 2 Time Series Analysis

In this paper we investigate four models that can be used to analyze time-based data series for prediction of trends observed in the software development process, based on the information obtained from mentioned tools (e.g. GitLab). Those models are (1) Exponential Smoothing, (2) Holt-Winters Model, (3) Autoregressive Integrated Moving Average (ARIMA) and (4) Recurrent Neural Networks (RNN).

Time series analysis can be defined as the decomposition of a given $X_t$ signal into a trend $T_t$, a seasonal component $S_t$ and the residual component $e_t$ (an example is shown if Fig.1).

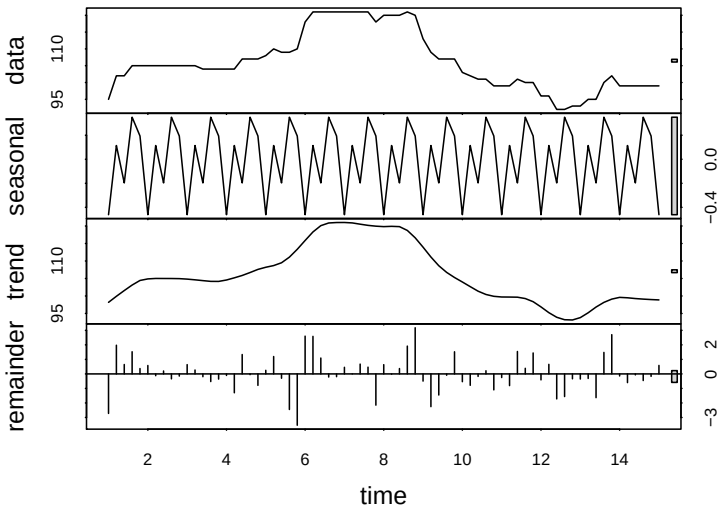


**Fig. 1.** Seasonal decompositioning of signal representing number of open issues in GitLab repository management system.

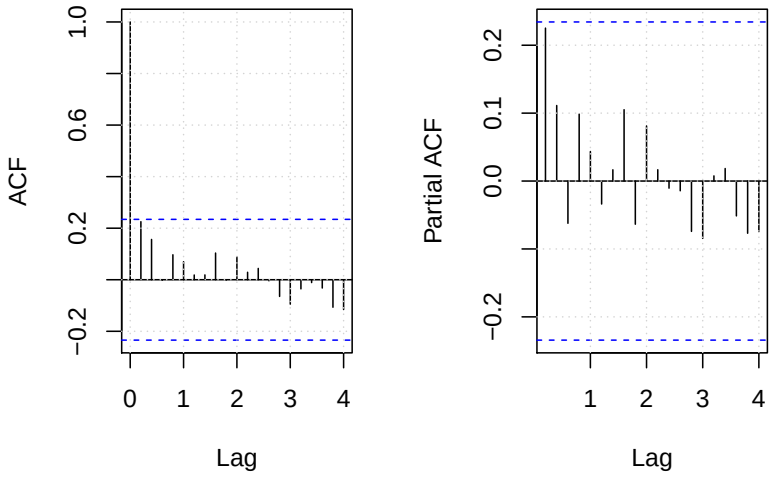


**Fig. 2.** ACF and PACF correlograms

The simplest way to obtain the trend of a signal is to use linear filter on the $X_t$:

$$T_t = \sum_{i=-\infty}^{\infty} \lambda_i X_{t+i} \tag{1}$$

An example of linear filter is a moving average with equal weights:

$$T_t = \frac{1}{2a+1} \sum_{i=-a}^{a} X_{t+i} \tag{2}$$

### 2.1 Exponential Smoothing

Exponential smoothing is a method for refining time series data using the exponential window function. In contrast to the simple moving average technique, the forecasting of data trends using exponential smoothing is characterized by exponentially decreasing weights of past observations, instead of equally weighted observations. This method is especially beneficial in forecasting time series characterized by a low variation of trend and seasonal changes. Numerous recent works show the viability of exponential smoothing for prediction purposes. In the project management area, [3] and [13] propose improvements to well-known Earned Value Management (EVM) project control methodology and Earned Duration Index (EDI) parameter by enhancing it with exponential smoothing to predict project progress and calculate project completion time and cost. In both cases, application of exponential smoothing was easy to incorporate to the traditional forecasting methodologies, raised their accuracy and reduced errors. For example, EVM enhanced by this technique led to the production of 14.8% more accurate time predictions and 25.1% more accurate cost predictions based on the data from 23 real-life projects examined by [3]. Exponential smoothing has been also applied to forecast financial/industrial indicators and variable physical quantities. [29] propose three methods using exponential smoothing to predict solar irradiance over time, based on historical data. Three variations of the method were examined: exponential smoothing with seasonal-trend decomposition, Global Horizontal Irradiance (GHI) time series decomposition (forecasted separately into a direct component and a diffuse component and then recombined) and the regression-based reconstruct of GHI through employing exponential smoothing for cloud coverage forecasting. [26] compared several methods based on exponential smoothing to forecast palm oil production in a given time. Those were: single ES, double ES (Holt Model), triple ES (Holt-Winters Model), triple ES additive and multiplicative. Due to the fact that historical data contain a seasonal component, authors recommend triple exponential smoothing additive as the suggested method for prediction, in this case, giving a smaller error value in the forecasted results.

Exponential smoothing predicts the next value of a given time series using geometric weighted sum of past data samples.

$$\hat{x}_{t=r}(1) = \alpha \cdot x_r + \alpha(1-\alpha) \cdot x_{r-1} + \alpha(1-\alpha)^2 \cdot x_{r-2} + \cdots \tag{3}$$

However, the exponential smoothing model should only be used with signals that have no systematic trend and seasonal components.

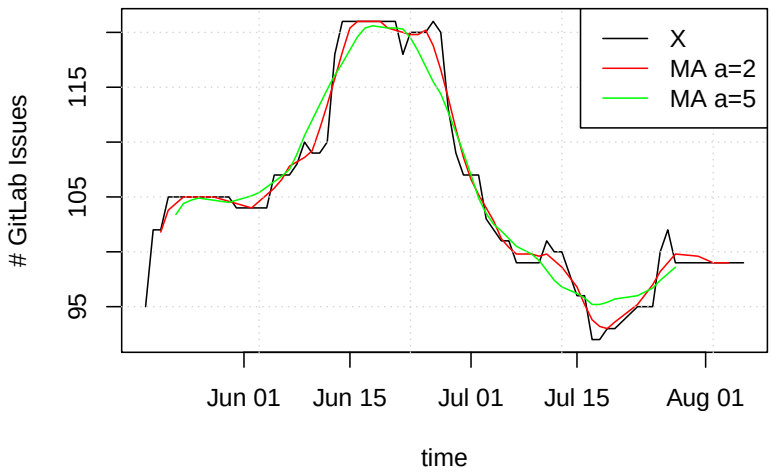


**Fig. 3.** Original signal $X$ and moving averages $a = 2$ and $a = 5$

### 2.2 Holt-Winters Model

Triple exponential smoothing (called Holt-Winters Model) is an example of a method appropriate for forecasting data points in a series that are repetitive over a given period (are seasonal). One of the cases studied in literature and similar to our work, where the seasonal component can impact the overall prediction of a given phenomenon is in software reliability forecasting, addressed by [27]. Authors experimented with non-parametric method for prediction of the number of software bugs in a testing environment, employing the Holt-Winters model and comparing it to well-known parametric software reliability models. According to the study, parametric models developed for prediction of software reliability generally assume that bug detection process is a Markov process or a non-homogeneous Poisson process, that the fault intensity is proportional to the number of remaining faults and fault detection rate is constant. Authors have noticed an advantage of non-parametric methods (such as exponential smoothing) over parametric ones through a decreased volume of historical data and a reduced computational effort necessary to obtain accurate predictions. Specifically, using double or triple exponential smoothing (which addresses seasonal variations

as well) reduces the impact of predicted abnormal data and eliminates the interference. The experimental results show that use of Holt-Winters model pays off not only in better forecasting than double exponential smoothing and than traditional, parametric models, but also that it results in extremely accurate predictions in terms of a precise number of software failures emerging in the future. Apart from software engineering, numerous publications address the problem of forecasting of seasonal variations with the use of the Holt-Winters method in domains other than software engineering. For example, [25] uses triple exponential smoothing to predict cloud resource provisioning to model cloud workload with multi-seasonal cycles, while some other sources examine Holt-Winters model in seasonal sales prediction and tourist attendance peaks forecasting.

As described above, the Holt-Winters model can be used in order to deal with the limitations of the exponential smoothing model. The Holt-Winters model has three smoothing parameters, namely $\alpha$ (for the constant level value), $\beta$ (for the trend), and $\gamma$ (for seasonal variations).

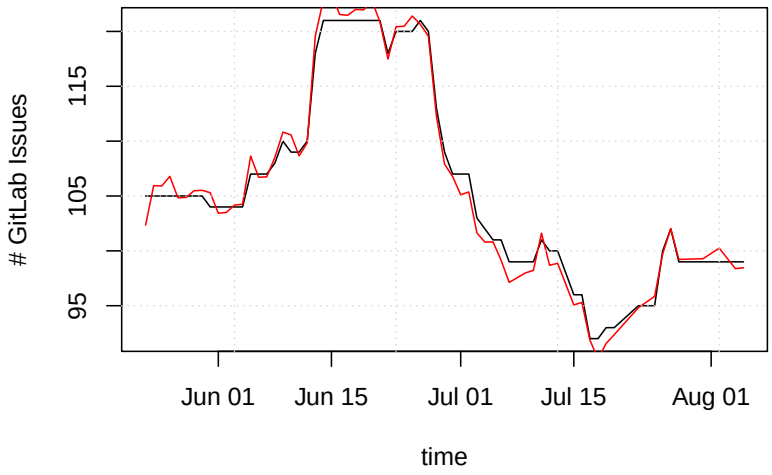


**Fig. 4.** Holt-Winters fitted model (red) for GitLab issues time series (black)

### 2.3 ARIMA

ARIMA stands for Autoregressive Integrated Moving Average. ARIMA is composed of an autoregressive and a moving average model, where the autoregression

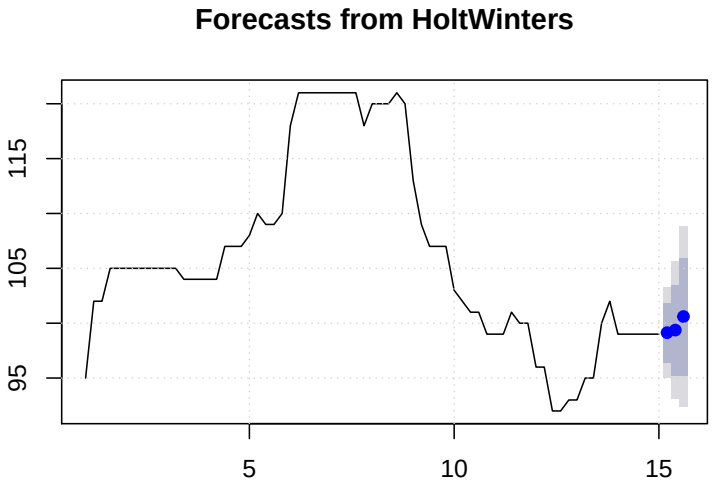


**Fig. 5.** Holt-Winters prediction for next 3 days. The gray bars indicate the confidence levels (95% dark gray and 80% light gray)

model involves regressing the variables based on past values. The model is commonly denoted as ARIMA($p$,$d$,$q$) where parameters p, d, and q are non-negative integers, which control three main components of the model: (i) the order (the number of time lags) of the autoregressive model, (ii) the degree of differencing, and (iii) the order of the moving-average.

It must be noted that ARIMA models are defined for stationary time series. In order to obtain it, the non-stationary time series has to be differenced until the non-stationarity is eliminated. The number of differencing iterations corresponds to the $d$ parameter of the model. In order to chose the other two parameters ($p$ and $q$) we must analyze the autocorrelation diagram (ACF) and the partial autocorrelation diagram (PACF). From the diagram shown in Fig.2, we may conclude that $p$ will be 0, because ACF function drops below the significance threshold (dashed line) after lag 0. Similarly, we can estimate the $q = 0$ value using the PACF diagram.

Autoregressive Integrated Moving Average (ARIMA) is a technique successfully applied for the prediction of non-stationary time series in different areas (e.g. finances or cybersecurity [2]). QoS in a cloud environment is a case for examination of the ARIMA model for cloud workload prediction purposes [4]. Authors concluded that their model is able to improve the prediction accuracy of up to 91%, contributing to a better cloud resources allocation with no impact on the cloud environment performance. ARIMA has also been studied for the pur-

poses of prediction in software engineering. For example, a comparative analysis of ARIMA, neural networks and a hybrid technique for Debian bug number prediction can be found in [19], where a model including a combination of ARIMA an Artificial Neural Networks has been presented as a promising approach to improve prediction accuracy. On the other hand, [16] shows that hybrid model based on the Singular Spectrum Analysis (SSA) and ARIMA outperforms other models in prediction of software failures during the software testing process.

Furthermore, Recurrent Neural Networks (RNN) gain traction in the context of forecasting and prediction based on time series data. A number of publications describe the use of the RNN for such purposes, including [6] Chang et al. 2014 for flooding water level forecasting, [24] Rout et al. 2017 for financial time series data prediction (forecasting of stock market indices) or [18] Muller-Navarra et al. 2015 for sales forecasting using partial recurrent neural networks. RNNs have some advantages over other methods used for time series prediction, namely: robustness to noise in a dataset, robustness to the incompleteness of the data, a better ability to predict from big datasets, a better ability to approximate non-linear functions and learning temporal dependencies among the data [8].

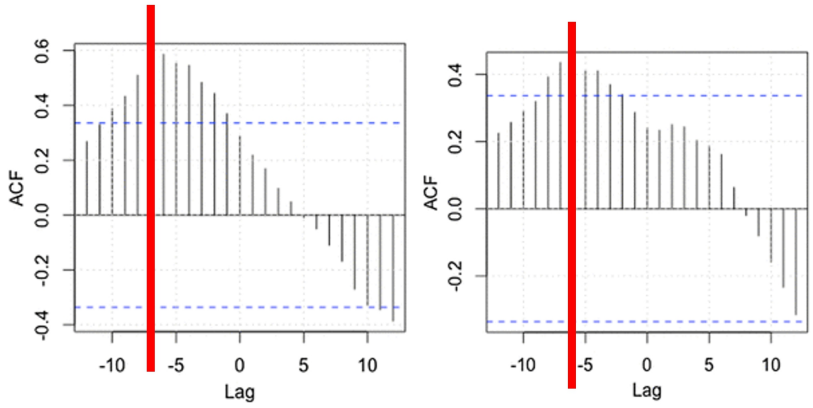


**Fig. 6.** Identified correlated facts: backlog size and the cognitive complexity (left), backlog size and the number of duplicated lines (right).

## 3 Prediction of software metrics using time series analysis

In our research we have used real data coming from SW development team working on commercial and research IT products (web applications) from several projects. The software team consists of up to 10 individuals, and they extensively use GitLab and SonarQube. According to the procedure described in the previous publications concerning Q-Rapids approach to data gathering and analysis, we have created connectors to those software management tools in order to calculate

metrics, factors and strategic indicators. Hereby, our work facilitates adding the prediction aspect to help product owners in their daily decision making.

### 3.1 Predicting Metrics Trends

In our research we have attempted the application of numerous Time Series prediction methods with regard to software quality and fault prediction. The first of the discussed methods utilizes simple moving averages. The moving average is a useful tool that smooths out input signal. The algorithm calculates the average over a specific period of time. Among others, it helps to eliminate the noise from the input time series. In principle, a moving average with a short time frame will react faster to signal changes than a moving average with a longer time period. Two moving averages (for a time period a=2 and a=5), are shown in Fig.3. In the presented example we have used a moving average to analyse a number of issues opened in a repository manager (in our case GitLab). The common use case of two moving averages is as follows. When the moving average with a shorter period crosses above the moving average with a longer one it may indicate that the trend of the observed signal will be rising. For example, we can see this for June 14 in Fig.3. A more sophisticated approach for time series analysis is the Holt-Winters model. The method determines the trend of the data it operates on, and estimates the seasonality of the series. The fitted Holt-Winters model of GitLab issues time series has been shown in Fig.4. In contrast to the moving average approach, the Holt-Winters algorithm provides a better fit for the original data. Moreover, the model also uses a more formal mathematical tool suit to provide the prediction capabilities. The predictions for the next three days for the same data have been presented in Fig.5. The procedure constitutes an approximation with a certain confidence level of a given time series based on the raw data.

### 3.2 Forecasting Metrics with ARIMA

In the literature, we can find numerous examples [28] that the Autoregressive Integrated Moving Average often outperforms the Holt-Winters algorithm. Therefore, forecasting using the ARIMA model has also been attempted in this work.

In general, we have observed that ARIMA seasonal model allowed us to achieve a higher (than the Holt-Winters algorithm) fitting degree and lower forecasting error. For example, in Fig. 6 we have shown, that for a short time frame prediction, the ARIMA model allowed us to identify that the increasing trend of number of items in sprint backlog may cause deterioration of the code quality (increased cognitive complexity and number of duplicated lines). In that case we have used 3 ARIMA models: one to predict the backlog size, one to predict the cognitive complexity, and one to predict the number of duplicated lines. Statistically significant correlations have been spotted with auto correlation function (ACF).

## 4 Quantitative Evaluation of Prediction Models

In this section we have formally evaluated different prediction models used to forecast various projects and software quality metrics. We have used commonly adapted forecast accuracy metrics, such as RMSE (Root Squared Mean Error), MAE (Mean Absolute Error), MASE (Mean Absolute Scaled Error), and MAPE (Mean Absolute Percentage Error). For each metric we define error as forecast error, which is a deviation $e_{T+h}$ of the forecast $\hat{y}_{T+h}$ from the observation $y_{T+h}$ within a predefined time lag $h$:

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h} \tag{4}$$

Moreover, here we assume that we split the time series $y$ in to a training set $\{y_1, \cdots, y_T\}$ and testing set $\{y_T, y_{T+1} \cdots\}$. In our experiments we have evaluated the prediction effectiveness of following models, namely (described in previous section) ARIMA and Holt-Winters, and random walk forecast. The effectiveness results for different scenarios have been reported in Tab.1,2,3.

**Table 1.** Effectiveness comparison of forecast models (Sprint Backlog Size)

| | RMSE | MAE | MASE | MAPE |
|---|---|---|---|---|
| ARIMA | 0.448 | 0.281 | 9.258 | 16.830 |
| Random Walk | 0.496 | 0.335 | 11.267 | 19.492 |
| Holt-Winters | 0.570 | 0.409 | 14.669 | 24.526 |

In the first experiments we have used different forecasting models to predict amount of task in a sprint backlog. The results show that the ARIMA model achieves the lowest error rates. Surprisingly, we may notice that the highest errors are reported for Holt-Winters prediction model.

**Table 2.** Effectiveness comparison of forecast models (Tasks in progress)

| | RMSE | MAE | MASE | MAPE |
|---|---|---|---|---|
| ARIMA | 0.268 | 0.160 | 11.339 | 12.561 |
| Random Walk | 0.286 | 0.215 | 14.773 | 17.995 |
| Holt-Winters | 0.308 | 0.235 | 16.836 | 19.124 |

We have identified the similar results also for other metrics, namely the number of task under development (tasks in progress) and the number of delayed tasks.

## 5 Conclusions

In this paper we have addressed the problem of the analysis of the software development and quality data. Our thesis was that the time-series analysis methods

**Table 3.** Effectiveness comparison of forecast models (Delayed tasks)

| | RMSE | MAE | MASE | MAPE |
|---|---|---|---|---|
| ARIMA | 0.369 | 0.161 | 223.958 | 7.208 |
| Random Walk | 0.487 | 0.251 | 171.567 | 13.963 |
| Holt-Winters | 0.671 | 0.461 | 55.947 | 24.017 |

are well suited to the challenge of software related data prediction. Such methods and tools (e.g. offered by H2020 Q-Rapids Project) can be helpful to Product Owners in their daily decisions related to e.g. assignment of tasks, time predictions, bugs predictions, time to release etc. We conducted our experiments on real case data in order to forecast various project-related characteristics. We have compared different forecasting tools such as ARIMA, Random Walk, and Holt-Winters. Our experiments have shown that ARIMA model performs well on our data. It achieved the lowest prediction errors among compared ones.

## 6 Acknowledgments

This work is funded under Q-Rapids project, which has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 732253.

## References

1. Tovey A. Cyber attacks cost British industry 34bn a year, online publication. `http://www.telegraph.co.uk/finance/newsbysector/industry/defence/11663761/Cyber-attacks-cost-British-industry-34bn-a-year.html`, 2018.
2. Choras M. Kozik R. Andrysiak T., Saganowski L. Network traffic prediction and anomaly detection based on arfima model. *in Jos Gaviria de la Puerta et al., Advances in Intelligent Systems and Computing*, vol. 229:545–554, 2014.
3. Jordy Batselier and Mario Vanhoucke. Improving project forecast accuracy by integrating earned value management with exponential smoothing and reference class forecasting. *International Journal of Project Management*, 35(1):28 – 43, 2017.
4. R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya. Workload prediction using arima model and its impact on cloud applications qos. *IEEE Transactions on Cloud Computing*, 3(4):449–458, Oct 2015.
5. Capgemini. Capgemini: World Quality Report 2016-17. `https://www.capgemini.com/world-quality-report-2016-17/`, 2017. (Accessed 9 Oct 2017).
6. Fi-John Chang, Pin-An Chen, Ying-Ray Lu, Eric Huang, and Kai-Yao Chang. Real-time multi-step-ahead water level forecasting by recurrent neural networks for urban flood control. *Journal of Hydrology*, 517:836 – 846, 2014.
7. Michał Choraś, Rafał Kozik, Damian Puchalski, and Rafał Renk. Increasing product owners' cognition and decision-making capabilities by data analysis approach. *Cognition, Technology & Work*, 21(2):191–200, May 2019.

8. Georg Dorffner. Neural networks for time series processing. *Neural Network World*, 6:447–468, 1996.
9. Michael Felderer and Rudolf Ramler. Risk orientation in software testing processes of small and medium enterprises: An exploratory and comparative study. 24:519–548, 08 2015.
10. X. Franch, C. Ayala, L. López, S. Martínez-Fernández, P. Rodrguez, C. Gómez, A. Jedlitschka, M. Oivo, J. Partanen, T. Räty, and V. Rytivaara. Data-driven requirements engineering in agile projects: The q-rapids approach. pages 411–414, Sept 2017.
11. Desjardins J. Qasymphony, how many millions of lines of code does it take? "`https://informationisbeautiful.net/visualizations/million-lines-of-code/`", 2017.
12. Capers Jones and Olivier Bonsignour. *The Economics of Software Quality*. Addison-Wesley Professional, 1st edition, 2011.
13. Homayoun Khamooshi and Abdollah Abdi. Project duration forecasting using earned duration management with exponential smoothing techniques. 33:04016032, 07 2016.
14. Puchalski D. Renk R. Kozik R., Choras Michal. Platform for software quality and dependability data analysis. *in: Zamojski, W. et al. (Eds.): Contemporary Complex Systems and Their Dependability Proceedings of the Thirteenth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX*, pages 306–315, July 2-6 2018.
15. Guzman L, Oriol M, Rodrguez P, Franch X, Jedlitschka A, and Oivo M. How can quality awareness support rapid software development – research preview. *In International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages pp. 167–173, 2017.
16. G. Liu, D. Zhang, and T. Zhang. Software reliability forecasting: Singular spectrum analysis and arima hybrid model. pages 111–118, Sept 2015.
17. Lidia López, Silverio Martínez-Fernández, Cristina Gómez, Choraś Michał, Rafał Kozik, Liliana Guzmán, Anna Maria Vollmer, Xavier Franch, and Andreas Jedlitschka. Q-rapids tool prototype: Supporting decision-makers in managing quality in rapid software development. *J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018*, LNBIP 317:200–208, 2018.
18. Moritz Müller-Navarra, Stefan Lessmann, and Stefan Voß. Sales forecasting with partial recurrent neural networks: Empirical insights and benchmarking results. *2015 48th Hawaii International Conference on System Sciences*, pages 1108–1116, 2015.
19. J. Pati and K. K. Shukla. A comparison of arima, neural network and a hybrid technique for debian bug number prediction. pages 47–53, Sept 2014.
20. Jorgensen PC. Software testing: a craftsman's approach, 2016.
21. Kozik R. Choraś M. Puchalski D., Renk R. Q-rapids framework for advanced data analysis to improve rapid software development. *Journal of Ambient Intelligence and Humanized Computing*, 10:1927–1936, 05 2019.
22. Q-Rapids. Q-rapids h2020 project (2018). `http://www.q-rapids.eu/`, 2018. "Project website".
23. QASymphony. QASymphony, The Cost of Poor Software Quality. `https://www.qasymphony.com/blog/cost-poor-software-quality/`, 2016. online publication.
24. Ajit Kumar Rout, P.K. Dash, Rajashree Dash, and Ranjeeta Bisoi. Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach. *Journal of King Saud University - Computer and Information Sciences*, 29(4):536 – 552, 2017.

25. Ashraf A. Shahin. Using multiple seasonal holt-winters exponential smoothing to predict cloud resource provisioning. *CoRR*, abs/1701.03296, 2016.
26. B Siregar, I A Butar-Butar, RF Rahmat, U Andayani, and F Fahmi. Comparison of exponential smoothing methods in forecasting palm oil real production. *Journal of Physics: Conference Series*, 801(1):012004, 2017.
27. Jinyong Wang, Zhibo Wu, Yanjun Shu, Zhan Zhang, and Lixing Xue. A Study on Software Reliability Prediction Based on Triple Exponential Smoothing Method (WIP). pages 61:1–61:9, 2014.
28. Wang, Zhi-Hui, Lu, Chen-Yang, Pu, Bin, Li, Gui-Wen, and Guo, Zhen-Jiang. Short-term forecast model of vehicles volume based on arima seasonal model and holt-winters. *ITM Web Conf.*, 12:04028, 2017.
29. Dazhi Yang, Vishal Sharma, Zhen Ye, Lihong Idris Lim, Lu Zhao, and Aloysius W. Aryaputera. Forecasting of global horizontal irradiance by exponential smoothing, using decompositions. *Energy*, 81:111 – 119, 2015.