Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

Article

Deep Cross-Project Software Reliability Growth Model using Project Similarity Based Clustering

Kyawt Kyawt San[1], Hironori Washizaki[1], Yoshiaki Fukazawa[1], Kiyoshi Honda[2], Masahiro Taga[3], Akira Matsuzaki[3]

[1]Waseda University; kks@fuji.waseda.jp
[2]Osaka Institute of Technology; kiyoshi.honda@oit.ac.jp
[3]e-Seikatsu Co., Ltd.; masahiro.taga@e-seikatsu.co.jp
* Correspondence: washizaki@waseda.jp

check for updates

Abstract: Software reliability is an important characteristic for ensuring the qualities of software products. Predicting the potential number of bugs from the beginning of a development project allows practitioners to make the appropriate decisions regarding testing activities. In the initial development phases, applying traditional software reliability growth models (SRGMs) with limited past data does not always provide reliable prediction result for decision making. To overcome this, herein we propose a new software reliability modeling method called deep cross-project software reliability growth model (DC-SRGM). DC-SRGM is a cross-project prediction method that uses features of previous projects' data through project similarity. Specifically, the proposed method applies cluster-based project selection for training data source and modeling by a deep learning method. Experiments involving 15 real datasets from a company and 11 open source software datasets show that DC-SRGM can more precisely describe the reliability of ongoing development projects than existing traditional SRGMs and the LSTM model.

Keywords: Software reliability, deep learning, long short-term memory, project similarity and clustering, cross-project prediction

1. Introduction

Reliability is one of the most significant attributes to enhance the quality of the product in the software development process [1–3]. Assessing software reliability is vital to deliver failure free software system. Despite the enormous amount of testing, a number of software defects always occur in the product [4]. Software Reliability Growth Models (SRGMs) express the number of potential errors or defects that might be occurred in the future from analyzing the past data such as the cumulative number of errors, test cases, error rate, and detection time [5]. Therefore, application of SRGMs helps to optimize resource planning and achieve the highly reliable systems.

The SRGMs are not always a reliable indicator to evaluate the situation of an ongoing software project and may even lead to an incorrect plan for testing resources [6]. New projects are not available past data which SRGM use for model fitting. In most studies, SRGM model fitting are relying on past data to predict future for the same project. Cross-project prediction is feasible in such case of requiring past data by applying the other projects. If a source project is dissimilar to the target project, it affects prediction performance and lead to unstable results for the future prediction.

© 2021 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

To address the deficiency of SRGMs and to adopt a more reliable method of software reliability growth modeling, this study [1] introduces a new SRGM method which can be utilized at the beginning stage of ongoing projects. For the target project with an insufficient amount of data, this method acquires the required information and features from similar projects to use in building model. More specifically, a clustering method, k-means is applied according to the features of projects such as the correlation of datasets and the number of bugs to create a new training data source. According to the identified clusters, the included datasets are combined. Prediction modeling is performed by deep long short-term memory (LSTM) model using the merged dataset.

The goals of the study are to:

- Identify the correlation among projects by the number of bugs occurrence pattern and the same attributes of the projects.
- Determine groups of similarity projects from a defect prediction viewpoint.
- Adopt a new approach for SRGM for the initial or ongoing stage of software development projects.

Here, we apply our proposed method called deep cross-project software reliability growth model (DC-SRGM) to 15 actual software projects from a company and 11 open source software (OSS) projects. Then we compare the performance of DC-SRGM with traditional models and the deep learning LSTM models. In our case study, DC-SRGM achieves the best scores in most cases. Hence, it can be regarded as an effective SRGM capable of improving deep learning LSTM models. Additionally, it significantly outperforms conventional SRGMs. Therefore, DC-SRGM method allows software developers and managers to understand project situations in an ongoing stage with limited historical data.

The contributions of this work are as follows:

- A new SRGM method that uses a combination of deep learning and a cluster-based project selection method.
- Experimental comparison to two different models using 15 empirical projects and 11 open source projects to verify the prediction accuracy of the proposed model comparing with two different models.
- Analysis of effective metrics, clustering factors and suitable time to create reliability growth models for industry.

The rest of the paper is organized as follows. Section 2 reviews the related works. Section 3 contains the employment of current project prediction method. Section 4 presents the proposed DC-SRGM framework. Section 5 explains the experimental setup, data and design. Section 6 reports the results and evaluations. Section 7 describes the threats to validity. Section 8 provides conclusions and future work respectively.

2. Related Work

Many SRGMs have been studied to measure the failure process. These models require external parameters to be estimated by the least squares or maximum likelihood estimation to build the relevant parameters [1]. N. Ullah et al. [8] studied different SRGMs using defect data in industrial and open source software and performed comparative analysis between them. To evaluate the qualities of development projects monitored by SRGM applications, K. Honda et al. [6] analyzed the tendencies for unstable situations in the results of different SRGM models. K. Okumoto et al. [4] applied SRGM in developing a reliability assessment automated tool.

---

[1] This paper is extended from our previous study [7]. We conducted additional experiments to investigate the impact of clustering factors, another similarity score using dynamic time warping, applying at different time points of ongoing projects and predictions across organizations.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

Ongoing projects have limited data for use as historical defect data. One alternative is to employ a cross-project prediction, which utilizes external projects to construct a prediction model for the current project [3,9]. In literature, cross-project prediction is a very well-study subject by utilizing project data of different organizations. K. Honda et al. [5] proposed a cross project SRGM model to compare software products within the same company. However, they did not implement cross-project applications of SRGMs for ongoing projects. Remarkably, there are a few studies in SRGM modeling using cross-project data.

The mismatch between the randomly selected source project and the target project affects the cross-project prediction performance and makes unstable results. Earlier studies in [10,11] implied that usage of cross-company data without any modification degrades accuracy of prediction models. Irrelevant source project data may decrease the efficiency of the cross-project prediction model. To overcome this issue, C. Liu et al. [12] considered the Chidamber and Kemerer (CK) metric suite [13] and size metrics to implement a cross-project model which detects change-proneness class files. Source projects were selected by the best-matched distribution. To choose appropriate training data, X. Zhang et al. [14] investigated the efficiencies of nine different relevancy filtering methods. Cross-projects defect prediction model was constructed with a random forest classifier on the PROMISE repository. M. Jureczko et al. [15] also studied a similar project clustering approach using k-means and hierarchical clustering by a stepwise liner regression in the PROMISE data repository. They confirmed that k-means can successfully identify similar project clusters from a defect prediction viewpoint. The above studies with cross-project prediction focused on the clustering or filtering approaches and employed a specific classifier to label defective module or class. None of these methods did not deal with the observed time series failure data.

J. Wang et al. [1] proposed a encoder-decoder-based deep learning model RNN and performed analysis between non-parameter models and parameter models. They applied the cumulative executive time and the accumulated number of defects. However, cross-project prediction model was not implemented. In addition, most of the past studies have not investigated sufficiently in the area of SRGMs modeling that utilize cross-project prediction. This study conducted projects reliability assessment by SRGM modeling with a sophisticated method rather than traditional approaches using cross-project data which was carefully selected with a project similarity method.

3. Current Project Prediction

Current project prediction applies existing project data as training source and then makes prediction for future days. Therefore, the LSTM and SRGM models in this study are created using only the target project's existing data, 50 percent data points. Then these models are used to predict the subsequent days for the rest 50 percent data points.

3.1. Software Reliability Growth Model

Software reliability growth models SRGMs are the Black box approach of software reliability model (SRM) [16] on the basis of failure data regardless of the source code characteristics [8]. The SRGM process are usually with data from testing. After development is done, if the detected failures or defects are resolved which enables system more stable and reliable. Therefore, to understand the underlying condition of the system, such processes are described using a mathematical expression, usually based on parameters such as number of failure or failure density, etc [17]. The literature reports many ways to create models based on the model's assumption of failure occurrence patterns. Similar to a previous study [6,18], we focused on the logistics model which is the most suitable with regard to fitness for the collected experimental datasets. We employed the model using the number of detected bugs and detected time. The Logistics model can be expressed as

$$N(t) = \frac{N_{max}}{1 + exp{-}A(t - B)} \tag{1}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

where $N(t)$ the number of bugs detected by time t. The parameters, N_{max}, A and B were estimated using Nonlinear Least Square Regression (NLR) function [6].

3.2. LSTM Model

A Recurrent Neural Network (RNN) connects neurons with one or more feedback loops, which is capable of modeling sequential data in sequence recognition and prediction [19,20]. Because it includes high-dimensional hidden states with nonlinear dynamics. These hidden states perform as the memory of the network, and its current state is conditioned on its previous one [21]. In a simple RNN structure, it has an input layer, recurrent hidden and output layers, which accepts the input sequences through time. Consequently, RNNs are capable of storing, remembering and processing data from past time periods, which enables the RNN to elucidate sequential dependencies [19]. However, it comes with the challenges that the memory produced from the recurrent connections may be limited to learning long range sequential data.

An RNN-based LSTM network is designed to resolve that problem. The LSTMs are capable to bridge very long-time lags with an advanced RNN architecture, self-connected units [19,22,23]. The inputs and outputs of hidden units are controlled by gates, which maintain the extracted features from previous time steps [19,23]. LSTM contains an input gate, forget gate, cell state, output gate, and output response. The input gate and forget gate manage the information flow into and out of the cell, respectively. The output gate decides what information is passed to the output cell. The memory cell has a self-connected recurrent edge of weight, ensuring that the gradient can pass across many time steps without exploding [24]. The advantage of an LSTM model is it can keep information over long periods by removing or adding information to the state.

Here, we constructed a LSTM model for SRGM modeling. At each time step, the input layer receives a vector of the number of bugs, pass the data to hidden layers which have four LSTM neurons in each. An output layer generates a single output that gives the predictions for the next time step.

3.3. Application of current project prediction

To allocate optimal testing resources, application of SRGMs are primarily used in cloud service development projects of e-Seikatsu company thus allows managers to assess the release readiness. By current project prediction, traditional SRGMs cannot realize underlying project condition if they are applied at initial stage with limited historical data as shown in Figs. 1a. Therefore, we applied an advanced technique LSTM model with the same amount of data during model construction. Although improvements occur (Figs. 1b), the LSTM model does not always give the accurate results at the beginning in cases with very little data that has different reliability growth pattern.

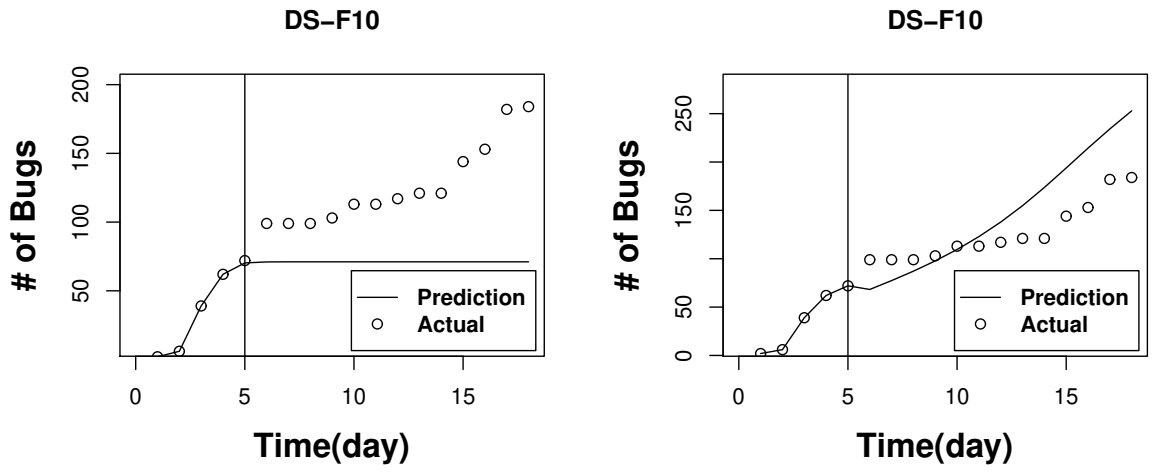
In earlier studies, cross-project predictions models have been utilized to resolve the requirement of a huge historical data. However, one challenge in the cross-project prediction is that the distribution of the source and target project usually differ significantly [14,25]. If the training data contains all the source project data, a poor prediction quality can be resulted. Ideally, one defect prediction model should work well for all projects that belong to a group [15]. Therefore, to eliminate the unrelated data from all source project for each target project, we derived Deep Cross-Project framework which process only the project data with the most common features of the target project.

4. Deep Cross-Project Software Reliability Framework

DC-SRGM utilizes a cross-project prediction method that use other projects data as a training data source with the advantageous of LSTM modelling for time series data.

Fig. 2 overviews the proposed model DC-SRGM. It includes three processes, similarity scoring, clustering-based project selection and prediction modeling. Fig. 3 details the process of selecting the most appropriate projects that shares common characteristics with the target project. The core feature of DC-SRGM is that it filters irrelevant projects from training data sources and only selects projects with the most common characteristics as the target project.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1



(a) Logistic Model **(b)** LSTM Model

Figure 1. Applying a Logistics model and LSTM model at day 5 for ongoing project F10.

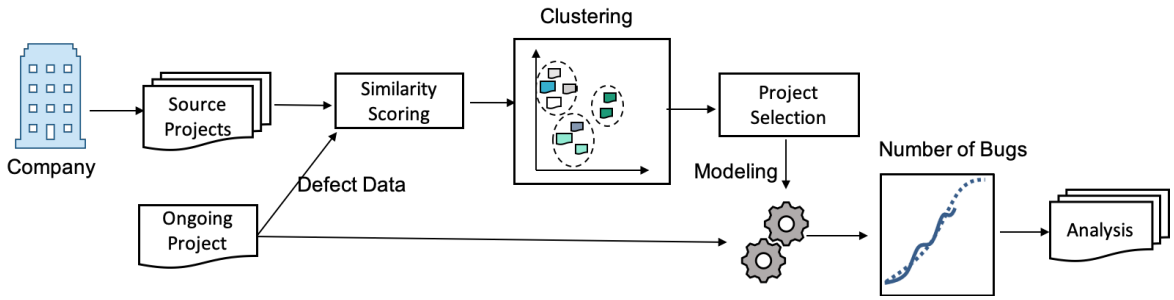


Figure 2. Overview of the DC-SRGM model

4.1. Similarity Scoring

Each project has its own features such as project size and number of bugs [3]. Identifying similarities among the datasets is the basis to eliminate differences between the data across projects. Otherwise, inappropriate source data may be chosen. To exclude irrelevant projects from training data sources, the clustering factors include project similarity scores. In DC-SRGM, cross-correlation is applied to identify the correlation of projects against with the target project.

Cross-correlation: A measure of the similarity among the projects by aligning two time series. The coefficients identify the connections between different time series of datasets [26]. Here, only time series data for detecting the number of bugs is included in the clustering factors scores between the candidate source project and target project. In given time series datasets d in N by the detected number of days for cumulative number of bugs, each dataset is considered as one time series and calculate the cross-correlation function of each pair (di, dj), $1 < i, j < N$, is calculated.

Dynamic Time Warping (DTW): A well-known technique to measure the optimal alignment or similarity between time series sequences of different lengths with respect to the shape of information and patterns [27]. It calculates the minimal distance to observe dissimilarities among the datasets according to the projects scale and distribution. Here, it is used to compare the performances of DC-SRGM.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

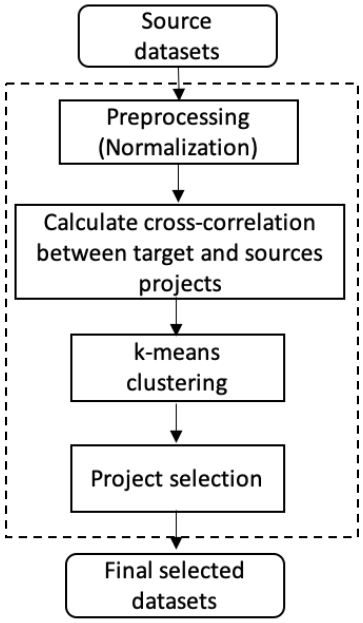


Figure 3. Project selection process

4.2. Project Clustering

Project clustering groups similar projects together using k-means algorithm with the following clustering factors:

- Cross-correlation similarity scores between the number of bugs growth patterns
- Normalized values of the maximum number of bugs
- Normalized values of the maximum number of days

The clustering results indicated three groups. Each group includes projects with characteristics similar to the target project according to the cross-correlation scores and the distribution of the projects such as the number of bugs and the number of days.

Table 1 summarizes the clustering factors, which are the cross-correlation similarity score, maximum number of bugs and maximum number of days. Table 2 summarizes the project clustering results in the industrial datasets. For each target dataset, the number of projects in each group differ slightly based on the similarity scores between the candidate target and source datasets. Table 2 details of each cluster such as the range of the number of bugs, number of days, and overall number of bugs of the included projects. "Grad" indicates a gradual increase in the detected number of bugs. "Expo" refers to an exponential increase in bug growth. "Expo & Grad" denotes both an exponential and gradual increase in the number of bugs.

Table 3 shows the clustering results by projects, where "Cluster" represents the cluster containing the target project. Projects applied for model building are presented in Table 2 according to the expressed cluster name. "Actual Growth" describes the bug growth of each project. "Prediction Result" shows the growth of the number of bugs by the prediction model created by clustered projects.

In this study, since the maximum number of bugs, maximum number days, and cross-correlation scores for the connections between projects are used as clustering factors, the obtained clusters are always three main groups depending on these factors, their similar attributes and data patterns. The first cluster denotes a group with moderate to strong correlation scores. The second cluster is influenced by exponential growth of the number of bugs. The third cluster is grouped by the distribution of the number of days of the projects. For example, F01 and F02 projects have the same distribution scales and a moderate cross-correlation score. Hence, they are grouped in the same cluster. On the other

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

Table 1. Summary of the clustering factors.

Similarity	Max Bugs	Max Days
0 ∼1	47 ∼ 752	14 ∼ 36

Table 2. Summary of the clustering results. Projects are generally clustered into three groups according to similarity scores and the projects scales. Grad, Expo & Grad, and Expo indicate the growth of number of bugs is gradually increasing, exponentially and gradually increasing, and exponentially increasing.

Cluster	Clustered projects	Max Bugs	Max Days	Growth	Type
C1	F01, F02, F04, F05, F07, F08, F09, F10, F11	91 ∼ 188	14 ∼ 22	Grad	Similarity
C2	F12, F15	540 ∼ 752	18 ∼ 24	Expo	# Bugs
C3	F03, F06, F13, F14	47 ∼ 331	22 ∼ 36	Expo & Grad	# Days

hand, the F12 project shows exponential growth for the number of bugs and a different data occurrence pattern. Building a model for the F01 project using F12 would overestimates the prediction result. Hence, DC-SRGM achieves better performance when applying in the middle of the projects to build a model using a similar group of projects.

4.3. Selection

To investigate whether a cluster for SRGM modeling exists, a prediction model is created by the datasets from each same cluster. According to our initial analysis, the cluster from the number of bugs prediction viewpoint exists only in the group with the target project itself. Because each group shares the most common attributes of the projects such as failure occurrence pattern and only those within the same group are appropriate to model for each project. In addition, only a cluster that belongs to the target project is selected. All the containing projects in that cluster are combined but the target project itself is excluded when merging the data. Eventually, the merged group of projects which have been eliminated the irrelevant training data, is used for model training.

4.4. Training

To employ the LSTM model, the input to the network at each time step is a vector of the number of bugs, and the single output is the number of bugs for the next time step. Fig. 4 shows the process of LSTM training at each time step. Because the ranges of the input values can vary, the values of bugs are scaled into the range of 0 to 1. By considering the prediction process as a time series, the input layer receives the values of number of bugs for 9 days and the single output node produces the next day prediction. By shifting by 1 in each time step, the model is trained to the maximum days of the training dataset. The model is trained with 300 epochs because the results are similar to those using 500 epochs. The stochastic gradient descent method is employed using the mean squared error loss function. For a target project prediction, the trained model is employed with fifty percent data points of its project to predict next fifty percent data points because we considered a project to be ongoing.

5. Experiments Design

Experiments were conducted to answer the following research questions. Fig. 3 overviews the evaluation design for each research question.

- **RQ1: Is DC-SRGM more effective in ongoing projects than other models?**

  This question evaluates the effectiveness of the DC-SRGM model compared to the Logistic model and LSTM model (Fig. 5, RQ1). That is, does the proposed method correctly describe ongoing projects' reliability despite insufficient data to apply in a prediction model. Specifically, we used a case study to compare the performance of different models for 15 industrial projects with a

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

Table 3. Summary of the clustering results by project. Grad, Expo & Grad, Expo and Const indicate the growth of number of bugs is gradually increasing, exponentially and gradually increasing, exponentially increasing and constantly increasing.

Project	Max Bugs	Days	Cluster	Actual Growth	Prediction Result
F01	91	19	C1	Grad	Grad
F02	137	22	C1	Grad	Grad
F03	47	36	C3	Grad	Grad
F04	122	17	C1	Grad	Grad
F05	188	19	C1	Grad	Grad
F06	263	26	C3	Expo	Expo
F07	146	15	C1	Grad & Expo	Grad
F08	97	17	C2	Grad	Grad
F09	99	16	C3	Expo & Grad	Grad
F10	184	18	C4	Grad	Grad
F11	74	14	C5	Grad	Grad
F12	540	24	C2	Expo	Expo & Grad
F13	187	22	C3	Expo & Grad	Expo & Grad
F14	752	18	C3	Expo & Grad	Const
F15	331	35	C2	Expo	Expo

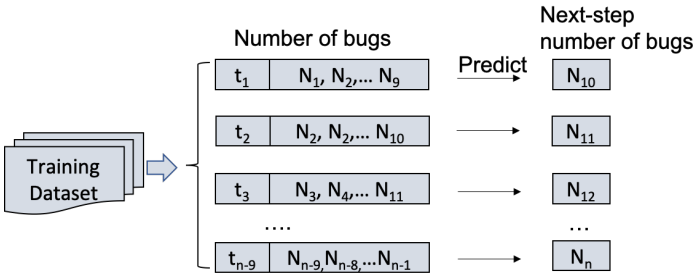


Figure 4. Model training process

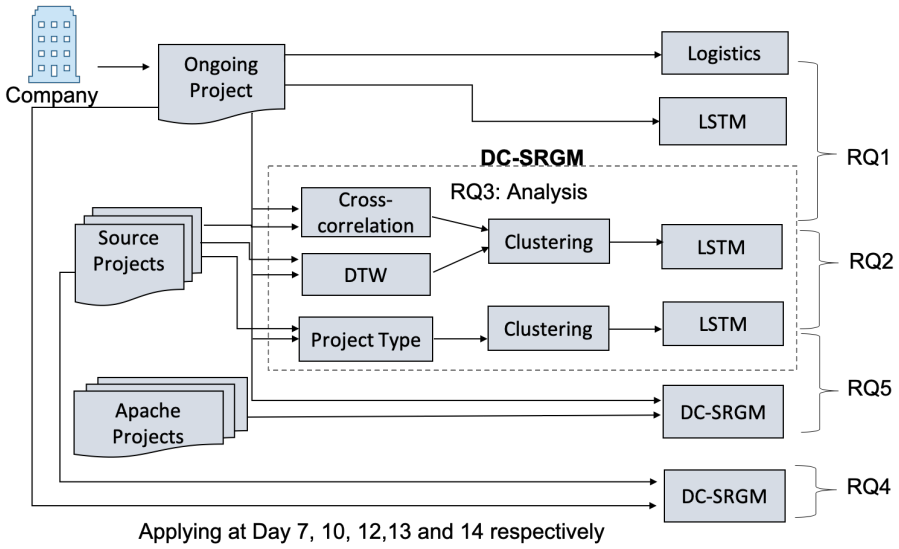


Figure 5. Overview of the evaluation design (Research Questions)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

duration longer than 14 days and 11 OSS projects. Because the target is an ongoing project, the first half of its data is used to obtain the similarity scores as well as for input data. Then the models are used to predict the second half of the target data. The results should reveal whether cluster-based similar project selection improves the LSTM model performance relative to that of a traditional Logistics model.

- **RQ2: What factors influence the performance of DC-SRGM?**

  This question examines the performance of DC-SRGM upon applying a different clustering factor to the similarity scores of the projects. Domain experts indicated that the projects are clustered according to the project domain type and the same types of projects are applied as the training source projects for modeling. We compared the prediction results with the results of similarity scores in term of AE values to reveal how different clustering factors influence the prediction results. This RQ help to assess whether DC-SRGM can be utilized when the same type of other projects is not available.
- **RQ3: Do different similarity measurements affect the prediction quality of DC-SRGM?**

  This research question investigates the performances of DC-SRGM based on cross-correlation and Dynamic Time Warping (DTW) to determine the impact of the similarity measurement techniques on the model (Fig. 3, RQ3). We analyzed the impact of the similarity measurement on DC-SRGM by comparing the performance of two methods in terms of AE values by model. In general, AE $>$ 0.10 indicated a satisfactory model.
- **RQ4: Can DC-SRGM precisely describe an ongoing projects' status?**

  This research question explores the relation of the amount of utilized project data and the model's prediction capability for new initial stage projects. It aims to determine to determine if there is a suitable time for managers to begin to evaluate projects with acceptable accuracy by DC-SRGM. Therefore, we applied the DC-SRGM model at different time points in ongoing projects to assess the prediction performance and the impact of target project's past data usage.
- **RQ5: Can DC-SRGM trained with OSS datasets indicate the industrial projects' situation?**

  Even if previous source projects data is unavailable, this RQ evaluates whether DC-SRGM created with OSS datasets can predict the conditions for an industrial project. We used open source datasets to create DC-SRGM with the same setting and procedure performed on industrial datasets. Then the results are compared to those predicted using industrial datasets.

5.1. Initial analysis

To identify similar groups, the initial analysis used cosine similarity and DTW. However, the similarity measurements and the prediction performance were not correlated. Therefore, the k-means clustering method was applied. Then the optimum number of clusters, k was determined by the Elbow method. Initially, the clustering produced biased results on the number of days. After the addition of cross-correlation coefficients in clustering factors, the same characteristics of the projects were classified.

5.2. Performance Measure

We evaluated the prediction capability in terms of accuracy by considering the ratio between the difference in the error values and the prediction over a time period namely average error (AE) [1]. AE is defined as:

$$AE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{U_{ij} - D_j}{D_j}\right| \tag{2}$$

where U_{ij} denotes the c number of predicted bugs by time t_j, D_j represents the accumulative number of detected bugs by time t_j, and n is the project size [1]. A value closer to zero indicates a better the prediction accuracy.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

Table 4. Industrial projects details.

Project	Days	# of Bugs
F01	19	91
F02	22	137
F03	12	47
F04	17	259
F05	19	188
F06	26	263
F07	15	146
F08	17	97
F09	16	99
F10	18	184
F11	14	74
F12	25	351
F13	22	187
F14	34	331
F15	18	752

Table 5. OSS projects details.

Project	Days	# of Bugs	Studied Version	
Camel	36	32	2.15.1	2.15.2
Ignite	48	149	2.5	2.6
Jclouds	175	25	2.1.0	2.1.1
Karaf	56	64	4.1	4.2
Lucene	91	6	6.6.0	6.6.1
Maven	160	22	3.5.1	3.5.2
Shiro	30	6	1.3.0	1.3.1
Spark	99	185	2.3.1	2.3.2
Syncope	80	36	2.0.2	2.0.3
Tez	120	27	0.6.0	0.6.1
Zookeeper	86	14	3.4.12	3.4.13

We employed the Frideman test with the Nemenyi test as a post-hoc test to evaluate the statistically significant difference in performances between DC-SRGM and the base line methods because it is better suited for non-normal distributions.

5.3. Data Collection

The datasets were from 15 projects data with a duration longer than 14 days from real cloud services development projects. Each dataset consisted of the time series number of bugs per the testing day. The domains of the projects were property information management, customer relationship management, contract management, money receipt/ payment management and content management systems [6]. To derive more generalize results, we aimed to include as many software projects as possible. Thus, 11 datasets from Apache open source projects were also collected from apache.org using a bug tracking system, JIRA, to study reliability growth modeling. For each project, all the issues reported in two minor versions, which were declared as bugs or defects excluding any other categories, were collected. Table 4 and 5 describe details of each dataset.

6. Results and Discussion

6.1. RQ1: Effectiveness of DC-SRGM

To generalize the finding, the experiments in RQ1 compared DC-SRGM to the Logistics and LSTM models. Table 6 and Table 7 present the AE values of the three models for the industrial datasets and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

OSS datasets respectively. Table 8 describes the results of the statistical test between DC-SRGM and the two other models. For the industrial datasets, DC-SRGM yielded the largest improvement. On average, it improved the AE by 24.6% and 50% compared to the LSTM and Logistic model, respectively. Table 6 compares the number of datasets where each model obtained better or worse (win or lose) scores across datasets. If a model obtained a score below the threshold (0.1), it was considered as an indicator of good accuracy. In most cases, DC-SRGM achieved better AE values. Fig. 6a also expresses the median of AE values among the three models. The red line represents the threshold. The DC-SRGM model had a lower AE values with a median below 0.1, implying a higher accuracy than the other two models. The LSTM models was close to the threshold, but the Logistics model had at the worst performance.

In case of the OSS datasets (Fig. 6b and Table 7), the results slightly differed, which is most likely due to the difference in the project nature between industrial and OSS projects. DC-SRGM achieved the best score. It had 65.4% improvement compared to the Logistic model in terms of AE average and better scores in terms of W/L. However, the performance with the LSTM model did not pass the significant test and its boxplot was bigger than the LSTM model. The LSTM model increased its accuracy in the OSS environment due to larger amount of training data. OSS datasets have a different development environment and style, specifically having a larger project size provides better accuracy for the LSTM model using the current project prediction method.

There are two exceptional cases where the proposed DC-SRGM was less accurate: F03 and F14 prediction. In the clustering result, the F03 project was grouped in the third cluster, which was grouped according to the number of days despite having a strong correlation with the projects in the first cluster. This impacted F03 modeling and is the reason DC-SRGM provided less accurate results than the LSTM model and Logistics model. In the F14 project, its domain differed from the other projects and it had a long duration according to the domain experts of these experimental projects.

Figs. 7a-7d plot the results when applying DC-SRGM, LSTM and Logistics models to the on F02, F03, F04 and F10 datasets at the middle of the projects, respectively. The predicted number of bugs by DC-SRGM described the potential number of bugs more correctly than the other two models. Hence, the results from both industrial and OSS datasets indicated that DC-SRGM outperformed LSTM and Logistics model and improved the prediction accuracy when applied in an ongoing stage of industrial development. For OSS projects, DC-SRGM significantly outperformed the Logistics model and, on average, was better than LSTM. However, its performance slightly decreased in the industrial environment while the performances of the LSTM model increased.

RQ1: Is DC-SRGM more effective in ongoing projects than other models? **For most datasets, the proposed DC-SRGM outperforms the LSTM and Logistics models as it has a lower mean AE value. The improvements are significant in industrial datasets. Hence, DC-SRGM is more effective to describe the future number of bugs correctly for ongoing software development projects.**

6.2. RQ2: Impact of clustering factors on DC-SRGM

RQ2 examined the prediction accuracy of two different clustering factors on DC-SRGM. Two models were built. One used the project similarity score, which is a cross-correlation, and the other used the project type to identify important factors for modeling. Fig. 6c shows a boxplot for median AE from the predictions using the two different clustering factors. "Project Similarity" and "Project Type" in Table 9 report the details of the AE values, where bold denotes the better result. Blank cells are projects which cannot be determined in the selected experiment datasets. The project similarity-based DC-SRGM obtained better scores in most cases, and the median was below the threshold. On the other hand, the project type-based model was close to the threshold. Hence, project clustering by similarity scores affected the model's ability to obtain suitable projects data to learn the number of bugs. Although the domain was the same, clustering by project type did not affect the model performance. For some projects, it might include irrelevant projects with very different growth pattern for the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

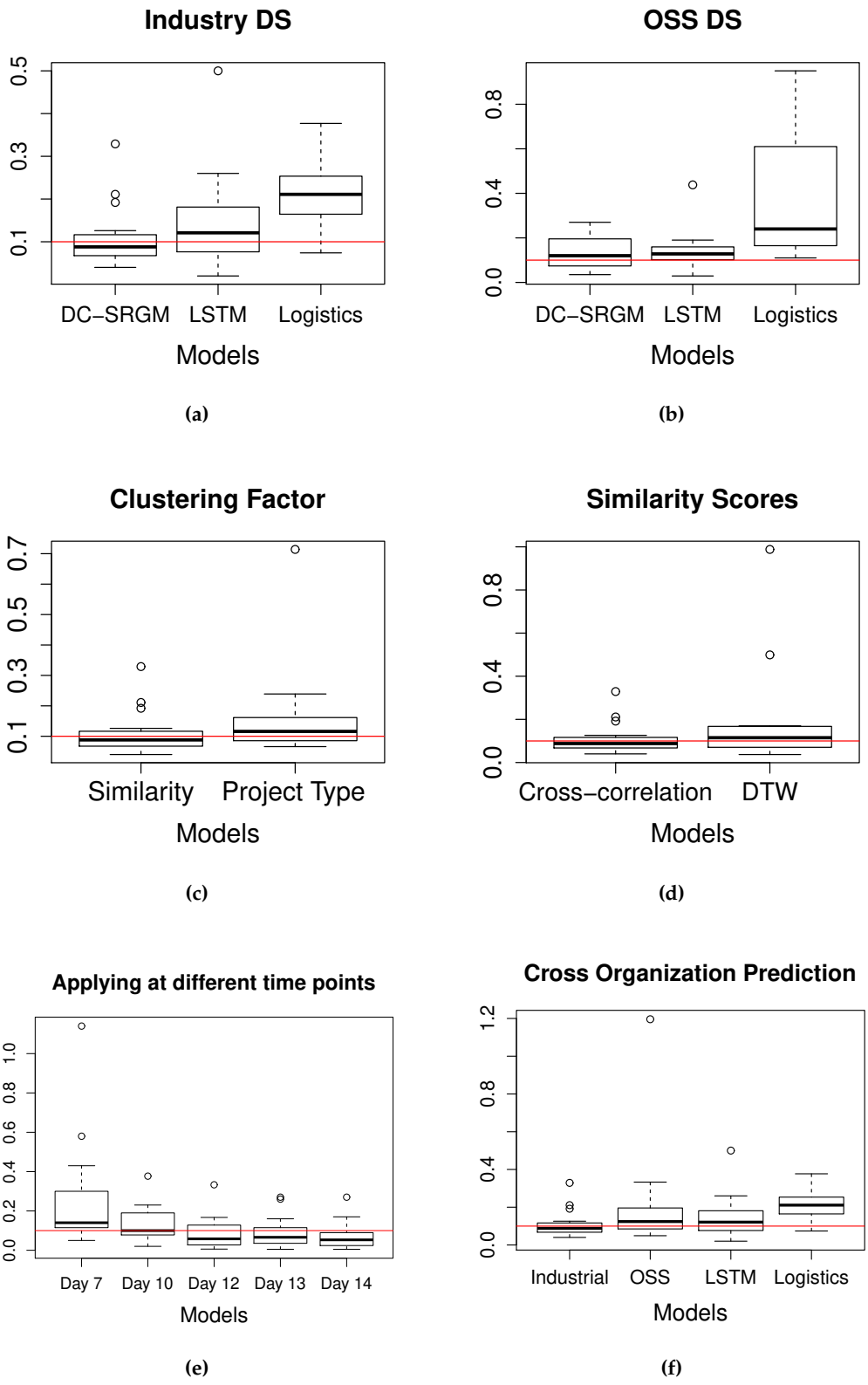


Figure 6. Comparison of the model prediction accuracy in terms of average error, AE. (a) Performance in Industrial datasets (DS), (b) Performance in OSS DS, (c) DC-SRGM based on project similarity and project type, (d) DC-SRGM based on cross-correlation and DTW, (e) DC-SRGM applied at different number of days, and (f) DC-SRGM across organizations.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

Table 6. Comparison of DC-SRGM with the LSTM and Logistics models by the AE values. Bold denotes the best AE values. W/L is the number of DS that each method is better and worse. # DS Threshold below 0.1 is the number of DS that each model's performance is lower than the threshold.

Project	DC-SRGM	LSTM	Logistics
F01	0.067	**0.040**	0.266
F02	**0.071**	0.080	0.146
F03	0.192	**0.130**	0.142
F04	**0.091**	0.260	0.377
F05	**0.075**	0.127	0.218
F06	**0.040**	0.090	0.211
F07	0.329	0.500	**0.146**
F08	**0.049**	0.104	0.187
F09	0.055	**0.048**	0.146
F10	**0.088**	0.121	0.214
F11	**0.068**	0.073	0.074
F12	**0.095**	0.161	0.359
F13	**0.211**	0.243	0.348
F14	0.107	**0.020**	0.183
F15	**0.126**	0.201	0.191
Average	**0.110**	0.146	0.220
Improved%	-	+24.6%	+50%
W/L	10/5	4/11	1/14
# DS Threshold below 0.1	10	6	1

Table 7. Prediction Accuracy of the models on OSS datasets by the AE values. Bold denotes the best AE Values. W/L is the number of DS that each method is better and worse. # DS Threshold below 0.1 is the number of DS that each model's performance is lower than the threshold.

Project	DC-SRGM	LSTM	Logistics
Camel	**0.081**	0.099	0.440
Ignite	0.067	**0.063**	0.110
Jclouds	0.190	**0.029**	0.260
Karaf	**0.035**	0.105	0.830
Lucene	**0.270**	0.438	0.950
Maven	**0.120**	0.122	0.240
Shiro	**0.100**	0.139	0.110
Spark	0.201	**0.139**	0.190
Syncope	0.240	**0.128**	0.220
Tez	**0.037**	0.180	0.780
Zookeeper	**0.133**	0.190	0.140
Average	**0.134**	0.148	0.388
Improved%	-	+9.45%	+65.4%
W/L	7/4	4/7	0/11
# DS Threshold below 0.1	5	3	1

Table 8. Statistic results with the Nemenyi test for the effectiveness of DC-SRGM. * and ** denote there were significant differences in the groups as the significance level were 0.1 and 0.01, respectively.

	Models	p_Value
Industry	DC-SRGM & LSTM	0.0710 *
	DC-SRGM & Logistics	0.0045 **
OSS	DC-SRGM & LSTM	0.3657
	DC-SRGM & Logistics	0.0288 *

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

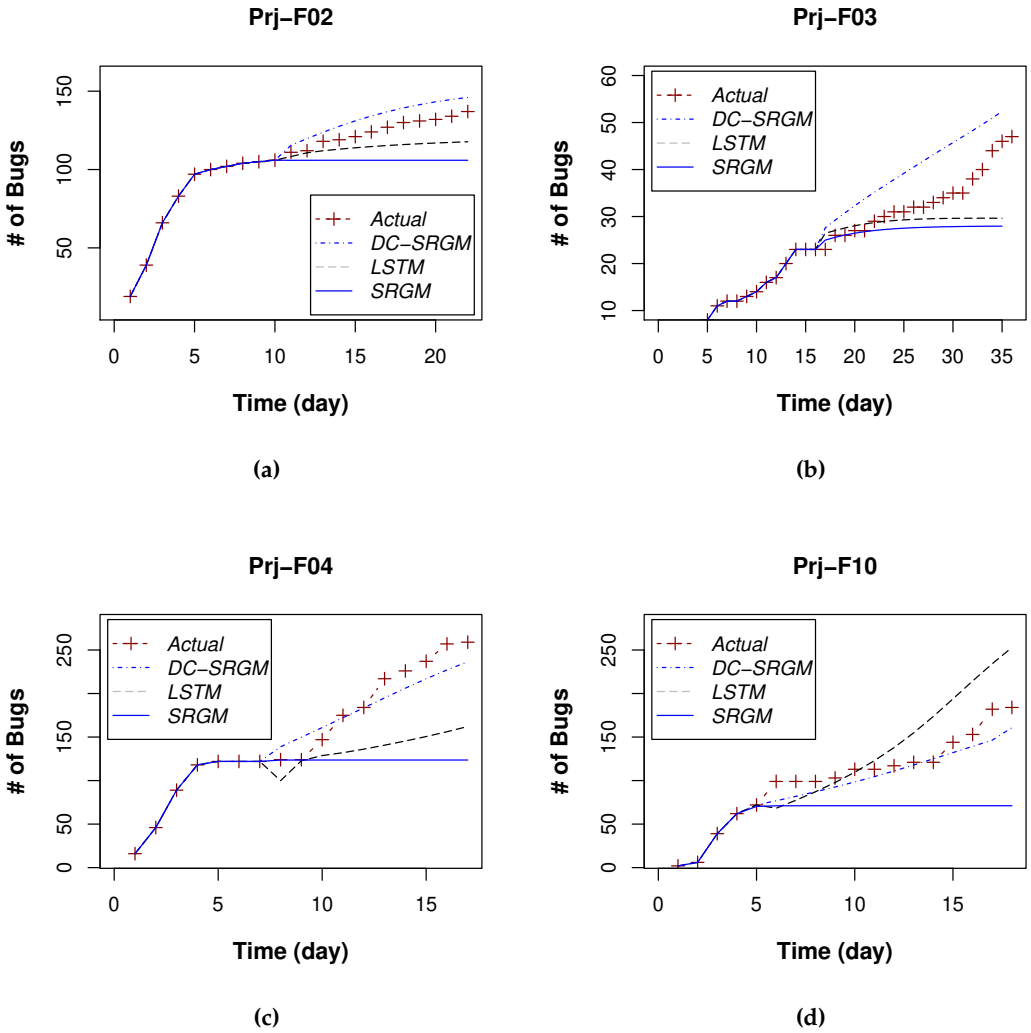


Figure 7. Predicted number bugs as function of the number of days. Actual, DC-SRGM, LSTM, SRGM represent the actual detected number of bugs, the prediction by DC-SRGM, LSTM model, and the Logistics SRGM model, respectively

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

Table 9. Comparison of the prediction accuracy of DC-SRGM using project similarity and project type as clustering factors. W/L is the number of DS that each method is better, and worse. Threshold below 0.1 is the number of DS that each method's performance is lower than the threshold.

Project	Project Similarity	Project Type
F01	**0.067**	0.074
F02	**0.071**	0.091
F03	0.192	**0.129**
F04	**0.091**	0.137
F05	0.075	–
F06	**0.040**	0.119
F07	**0.329**	0.714
F08	**0.049**	0.186
F09	**0.055**	0.113
F10	**0.088**	0.096
F11	0.068	**0.066**
F12	0.095	–
F13	**0.211**	0.239
F14	0.107	–
F15	0.126	**0.080**
Average	**0.110**	0.170
W/L	9/3	3/9
# DS Threshold below 0.1	10	7

number of bugs even though they are in the same domain. Therefore DC-SRGM modeling should be performed using the project similarity scores as the first priority rather than the project type.

RQ2: What factors influence the performance of DC-SRGM? **In most cases, DC-SRGM clustered by project similarity scores outperforms the model clustered by project type on AE values, indicating that project similarity is an important factor in the clustering process for good predictions results.**

6.3. RQ3: Impact of similarity measurements on DC-SRGM

RQ3 compared the performances of DC-SRGM based on cross-correlation and DTW to assess the impact of the similarity measurement technique and to determine a better similarity measurement for DC-SRGM. Fig. 6d shows boxplots for the median AE values of both methods. DC-SRGM based on the cross-correlation had lower AE values with a median below the threshold. On the other hand, the DTW-based model, was close to the threshold, implying that cross-correlation shows a better performance. "Cross-correlation" and DTW in Table 10 represents details of the AE values, where bold denotes the better method. Across 15 datasets, although there is no obvious difference between two methods in the number of datasets with the lower AE value, the cross-correlation-based model outperformed the DTW-based model on average, and achieved a lower than threshold in more cases.

Clustering based on DTW could not always classify relevant datasets or eliminate the irrelevant datasets for the target project. One reason is that DTW function basically returned the scores based on the shape of the datasets sequence whereas cross-correlation returned the scores based on the value and pattern of datasets. Another reason is that the cross-correlation scores can describe the correlation level such as significant or non-significant. In DTW, it is difficult to identify the threshold in the variations of datasets. Therefore, changing the applied similarity measurement technique impacted

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

Table 10. Comparison of the prediction accuracy DC-SRGM using cross-correlation and DTW as similarity measures. W/L is the number of DS that each method is better, and worse. Threshold below 0.1 is the number of DS that each method's performance is lower than the threshold.

Project	**Cross-correlation**	**DTW**
F01	0.067	**0.037**
F02	0.071	**0.039**
F03	**0.192**	0.499
F04	0.091	**0.081**
F05	0.075	**0.048**
F06	**0.040**	0.170
F07	**0.329**	0.988
F08	**0.049**	0.166
F09	**0.055**	0.089
F10	**0.088**	0.115
F11	**0.068**	0.169
F12	**0.095**	0.115
F13	0.211	**0.165**
F14	0.107	**0.060**
F15	0.126	**0.089**
Average	**0.110**	0.188
W/L	8/7	7/8
# DS Threshold below 0.1	10	7

the model performance. To identify the similar project groups correctly, the cross-correlation technique is better suited for DC-SRGM.

RQ3: Do different similarity measurements affect the prediction quality of DC-SRGM? **Cross-correlation-based DC-SRGM achieves a better accuracy than DTW. To enhance source project selection, cross-correlation is a better technique for DC-SRGM from the SRGM modeling viewpoint.**

6.4. RQ4: Impact of applying DC-SRGM at different time points

To determine the impact of the amount of data from an ongoing project applied in DC-SRGM modeling, the experiment was conducted using the target datasets from industrial data at day 7, 10, 12, 13, and 14. The model's performances at different time points were compared to determine if there is a suitable time frame to apply DC-SRGM in ongoing development stages. Table 11 shows the AE values of the models at each time point. Fig. 6e compares the median accuracy values at each prediction time point. Accurate results were not obtained when applying DC-SRGM at Day 7 of ongoing projects, but a few projects had significant improvement upon applying at Day 10. Applying the model at Day 12 or later improved the AE values. Overall, the proposed method can identify the correct clusters and achieve stable results starting from Day 12. Therefore, DC-SRGM can be applied to ongoing software development projects beginning on Day 12.

RQ4: Can DC-SRGM precisely describe ongoing projects' status? **The model applied at Day 12 of the ongoing projects provides a more stable and improved accuracy than the other models. Hence, managers can start applying DC-SRGM at Day 12 to correctly describe the reliability of a projects.**

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

Table 11. Comparison of DC-SRGM for different numbers of days. DS Threshold below 0.1 is the number of DS that each model's performance is lower than the threshold.

Project	Day 7	Day 10	Day 12	Day 13	Day 14
F01	0.070	0.078	0.072	0.060	0.060
F02	0.050	0.030	0.045	0.040	0.050
F03	0.580	0.377	0.167	0.160	0.170
F04	0.100	0.087	0.073	0.031	0.028
F05	0.130	0.070	0.029	0.024	0.020
F06	0.140	0.225	0.039	0.043	0.030
F07	1.140	0.780	0.333	0.270	0.140
F08	0.410	0.098	0.009	0.011	0.015
F09	0.160	0.111	0.007	0.005	0.005
F10	0.190	0.112	0.143	0.110	0.079
F11	0.140	0.020	0.006	0.007	0.007
F12	0.190	0.230	0.058	0.066	0.060
F13	0.430	0.190	0.025	0.260	0.270
F14	0.130	0.100	0.131	0.120	0.100
F15	0.080	0.190	0.125	0.087	0.050
Average	0.262	0.179	0.090	0.092	0.072
# of DS below Threshold	4/15	7/15	10/15	10/15	12/15

6.5. RQ5: predicting the performance by cross organization datasets

For RQ5, the experiment was designed to validate the effectiveness of the DC-SRGM model applied using cross organization OSS datasets for predictions of industrial projects. DC-SRGM trained by OSS datasets predicted the second half of the industrial datasets. The performance was compared with the results of models trained by industrial datasets.

Table 12 shows the AE values predicted utilizing industrial datasets and OSS datasets along with the performances of the LSTM model and Logistics model. Fig 6f shows the median of the AE values. Among the models, DC-SRGM based on industrial datasets achieved the best performance on average. However, the industry-based model and OSS-based model produced the same number of best cases. Therefore, OSS datasets can be applied to predict industrial projects in situations where source project data is unavailable.

RQ5: Can DC-SRGM trained with OSS datasets indicate the industrial project's situation? **DC-SRGM trained with OSS datasets obtains a better accuracy than LSTM models and Logistics model. However, its accuracy is not better than industrial projects-based model. Thus, OSS projects can be applied when previous source project data is unavailable.**

6.6. Use Cases

Practitioners from e-Seikatsu Co., Ltd wanted to focus on the situation of the ongoing software development projects because it helps with effective test planning and resource arrangements. Because the traditional reliability growth model could not describe the growth in number of bugs for a project, we attempted to model with an advanced methodology, deep learning based LSTM model. Due to the lack of training data, the model's performance required additional refinement. The company had a lot of data from previously developed and released projects. By applying data from previous projects, we

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

Table 12. Accuracies of DC-SRGM built with industrial datasets and cross-organization datasets (OSS) are compared with the LSTM model and Logistics model. W/L is the number of DS that each method is better, and worse. Threshold below 0.1 is the number of DS that each method's performance is lower than the threshold.

Project	DC-SRGM		LSTM	Logistics
	Industry DS	Cross-org DS		
F01	0.067	**0.051**	0.040	0.266
F02	**0.071**	0.104	0.080	0.146
F03	0.192	**0.107**	0.130	0.142
F04	**0.091**	0.124	0.260	0.377
F05	0.075	**0.049**	0.127	0.218
F06	**0.040**	0.136	0.090	0.211
F07	0.329	0.196	0.500	**0.146**
F08	**0.049**	0.333	0.104	0.187
F09	0.055	0.196	**0.048**	0.146
F10	**0.088**	0.120	0.121	0.214
F11	0.068	**0.066**	0.073	0.074
F12	0.095	**0.066**	0.161	0.359
F13	0.211	**0.205**	0.243	0.348
F14	0.107	0.172	**0.020**	0.183
F15	**0.126**	0.196	0.201	0.191
Average	**0.110**	0.141	0.146	0.220
W/L	6/9	6/9	2/13	1/14
# DS Threshold below 0.1	10	4	6	1

developed the DC-SRGM framework to apply in the middle or earlier stage of development projects. The proposed approach is applicable when the past data is unavailable in the initial stage of the current development projects. By implementing DC-SRGM in the ongoing projects of e-Seikatsu, the proposed model provided a more accurate prediction than the other models considered.

7. Threats to Validity

In this research, we treated the number of bugs growing as time-dependent variable for model construction. However, there may be other related factors. For example, the number of detected bugs may depend upon testing efforts. In addition, the experiment was conducted with one LSTM architecture, although the LSTM network architecture may impact to its prediction performance. Moreover, when collecting data from open sources, data validity in reporting defect data may be an issue. One study indicated that open software often contains information which is not described as bugs [28]. These are threats to the internal validity.

We tested DC-SRGM only with two datasets from two organizations. This is insufficient to make generalizations. In the future, testing of more datasets from many organizations needs to be performed. Additionally, when comparing models, only the Logistics model was considered as a traditional method. However, the literature reports many traditional SRGMs. These are threats to external validity.

One threat to the construct validity is that we supposed that identifying of correct clusters means the group of projects with the same attributes such as the project scale and growth pattern of number

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

of bugs rather than the projects domains. Therefore, the project domains may differ within the same cluster in actual cases.

8. Conclusions and Future work

Herein we propose a new software reliability growth modeling method using a combination of a LSTM model and a cluster-based project selection method based on similar characteristics of projects via a similarity scoring process. This proposed method alleviates issues regarding insufficient previous data, and is an improvement compared to traditional methods for reliability growth modeling. We conducted experiments using both industrial and OSS data to evaluate DC-SRGM with a statistical significant test. The case studies showed that DC-SRGM is superior to all other evaluated models. It achieves the highest accuracy in industrial datasets, indicating that the project similarity is more important than project type when clustering projects. Moreover, cross-correlation performs better than DTW in specifying project similarity from a defect prediction viewpoint. The experiment involving different time points indicated that DC-SRGM can be used for a project with 12 days of defect data to stably and accurately predict the number of bugs that might be encountered in subsequent days. Finally, DC-SRGM in ongoing projects can assist managers in decision-making for testing activities by understanding reliability growth in ongoing projects.

Future works include implementing other metrics such as software product metrics [29,30] in the clustering factors and model construction as well as comparing to other prediction models reported in the literature. To improve the quality and continuous monitoring, this method should be extended to provide more reliability metrics beyond the prediction of the number of bugs.

Author Contributions: Conceptualization and methodology, Kyawt Kyawt San.; literature review and analysis, all authors. All authors have read and agreed to the published version of the manuscript.

References

1. Wang, J.; Zhang, C. Software reliability prediction using a deep learning model based on the RNN encoder-decoder. *Reliab. Eng. Syst. Saf.* **2018**, *170*, 73–82. doi:10.1016/j.ress.2017.10.019.
2. Washizaki, H.; Honda, K.; Fukazawa, Y. Predicting Release Time for Open Source Software Based on the Generalized Software Reliability Model. 2015 Agile Conference, AGILE 2015, National Harbor, MD, USA, August 3-7, 2015. IEEE Computer Society, 2015, pp. 76–81. doi:10.1109/Agile.2015.19.
3. Xu, Z.; Pang, S.; Zhang, T.; Luo, X.; Liu, J.; Tang, Y.; Yu, X.; Xue, L. Cross Project Defect Prediction via Balanced Distribution Adaptation Based Transfer Learning. *J. Comput. Sci. Technol.* **2019**, *34*, 1039–1062. doi:10.1007/s11390-019-1959-z.
4. Okumoto, K.; Asthana, A.; Mijumbi, R. BRACE: Cloud-Based Software Reliability Assurance. 2017 IEEE International Symposium on Software Reliability Engineering Workshops, ISSRE Workshops, Toulouse, France, October 23-26, 2017. IEEE Computer Society, 2017, pp. 57–60. doi:10.1109/ISSREW.2017.48.
5. Honda, K.; Nakamura, N.; Washizaki, H.; Fukazawa, Y. Case Study: Project Management Using Cross Project Software Reliability Growth Model Considering System Scale. 2016 IEEE International Symposium on Software Reliability Engineering Workshops, ISSRE Workshops 2016, Ottawa, ON, Canada, October 23-27, 2016. IEEE Computer Society, 2016, pp. 41–44. doi:10.1109/ISSREW.2016.45.
6. Honda, K.; Washizaki, H.; Fukazawa, Y.; Taga, M.; Matsuzaki, A.; Suzuki, T. Empirical Study on Tendencies for Unstable Situations in Application Results of Software Reliability Growth Model. 2018 IEEE International Symposium on Software Reliability Engineering Workshops, ISSRE Workshops, Memphis, TN, USA, October 15-18, 2018; Ghosh, S.; Natella, R.; Cukic, B.; Poston, R.S.; Laranjeiro, N., Eds. IEEE Computer Society, 2018, pp. 89–94. doi:10.1109/ISSREW.2018.00-25.
7. San, K.K.; Washizaki, H.; Fukazawa, Y.; Honda, K.; Taga, M.; Matsuzaki, A. DC-SRGM: Deep Cross-Project Software Reliability Growth Model. IEEE International Symposium on Software Reliability Engineering Workshops, ISSRE Workshops 2019, Berlin, Germany, October 27-30, 2019; Wolter, K.; Schieferdecker, I.; Gallina, B.; Cukier, M.; Natella, R.; Ivaki, N.R.; Laranjeiro, N., Eds. IEEE, 2019, pp. 61–66. doi:10.1109/ISSREW.2019.00044.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

8. Ullah, N.; Morisio, M. An Empirical Study of Reliability Growth of Open versus Closed Source Software through Software Reliability Growth Models. 19th Asia-Pacific Software Engineering Conference, APSEC 2012, Hong Kong, China, December 4-7, 2012; Leung, K.R.P.H.; Muenchaisri, P., Eds. IEEE, 2012, pp. 356–361. doi:10.1109/APSEC.2012.80.
9. Porto, F.R.; Minku, L.L.; Mendes, E.; Simão, A. A Systematic Study of Cross-Project Defect Prediction With Meta-Learning. *CoRR* **2018**, *abs/1802.06025*, [1802.06025].
10. Kitchenham, B.A.; Mendes, E.; Travassos, G.H. Cross versus Within-Company Cost Estimation Studies: A Systematic Review. *IEEE Trans. Software Eng.* **2007**, *33*, 316–329. doi:10.1109/TSE.2007.1001.
11. Lokan, C.; Mendes, E. Investigating the Use of Chronological Splitting to Compare Software Cross-company and Single-company Effort Predictions. 12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008, University of Bari, Italy, 26-27 June 2008; Visaggio, G.; Baldassarre, M.T.; Linkman, S.G.; Turner, M., Eds. BCS, 2008, Workshops in Computing.
12. Liu, C.; Yang, D.; Xia, X.; Yan, M.; Zhang, X. Cross-Project Change-Proneness Prediction. 2018 IEEE 42nd Annual Computer Software and Applications Conference, COMPSAC 2018, Tokyo, Japan, 23-27 July 2018, Volume 1; Reisman, S.; Ahamed, S.I.; Demartini, C.; Conte, T.M.; Liu, L.; Claycomb, W.R.; Nakamura, M.; Tovar, E.; Cimato, S.; Lung, C.; Takakura, H.; Yang, J.; Akiyama, T.; Zhang, Z.; Hasan, K., Eds. IEEE Computer Society, 2018, pp. 64–73. doi:10.1109/COMPSAC.2018.00017.
13. Chidamber, S.R.; Kemerer, C.F. A Metrics Suite for Object Oriented Design. *IEEE Trans. Software Eng.* **1994**, *20*, 476–493. doi:10.1109/32.295895.
14. Bin, Y.; Zhou, K.; Lu, H.; Zhou, Y.; Xu, B. Training Data Selection for Cross-Project Defection Prediction: Which Approach Is Better? 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, ON, Canada, November 9-10, 2017; Bener, A.; Turhan, B.; Biffl, S., Eds. IEEE Computer Society, 2017, pp. 354–363. doi:10.1109/ESEM.2017.49.
15. Jureczko, M.; Madeyski, L. Towards identifying software project clusters with regard to defect prediction. Proceedings of the 6th International Conference on Predictive Models in Software Engineering, PROMISE 2010, Timisoara, Romania, September 12-13, 2010; Menzies, T.; Koru, G., Eds. ACM, 2010, p. 9. doi:10.1145/1868328.1868342.
16. Goel, A.L. Software Reliability Models: Assumptions, Limitations, and Applicability. *IEEE Trans. Software Eng.* **1985**, *11*, 1411–1423. doi:10.1109/TSE.1985.232177.
17. Rana, R.; Staron, M.; Berger, C.; Hansson, J.; Nilsson, M.; Törner, F. Evaluating long-term predictive power of standard reliability growth models on automotive systems. IEEE 24th International Symposium on Software Reliability Engineering, ISSRE 2013, Pasadena, CA, USA, November 4-7, 2013. IEEE Computer Society, 2013, pp. 228–237. doi:10.1109/ISSRE.2013.6698922.
18. Honda, K.; Washizaki, H.; Fukazawa, Y. Generalized Software Reliability Model Considering Uncertainty and Dynamics: Model and Applications. *Int. J. Softw. Eng. Knowl. Eng.* **2017**, *27*, 967. doi:10.1142/S021819401750036X.
19. Salehinejad, H.; Baarbe, J.; Sankar, S.; Barfett, J.; Colak, E.; Valaee, S. Recent Advances in Recurrent Neural Networks. *CoRR* **2018**, *abs/1801.01078*, [1801.01078].
20. Bengio, Y.; Simard, P.Y.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks* **1994**, *5*, 157–166. doi:10.1109/72.279181.
21. Mikolov, T.; Joulin, A.; Chopra, S.; Mathieu, M.; Ranzato, M. Learning Longer Memory in Recurrent Neural Networks. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings; Bengio, Y.; LeCun, Y., Eds., 2015.
22. Zhang, X.; Ben, K.; Zeng, J. Cross-Entropy: A New Metric for Software Defect Prediction. 2018 IEEE International Conference on Software Quality, Reliability and Security, QRS 2018, Lisbon, Portugal, July 16-20, 2018. IEEE, 2018, pp. 111–122. doi:10.1109/QRS.2018.00025.
23. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. doi:10.1162/neco.1997.9.8.1735.
24. Zhu, W.; Lan, C.; Xing, J.; Zeng, W.; Li, Y.; Shen, L.; Xie, X. Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks. Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA; Schuurmans, D.; Wellman, M.P., Eds. AAAI Press, 2016, pp. 3697–3704.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 October 2021 doi:10.20944/preprints202110.0237.v1

25. Turhan, B.; Menzies, T.; Bener, A.B.; Stefano, J.S.D. On the relative value of cross-company and within-company data for defect prediction. *Empir. Softw. Eng.* **2009**, *14*, 540–578. doi:10.1007/s10664-008-9103-7.
26. Egri, A.; Horváth, I.; Kovács, F.; Molontay, R.; Varga, K. Cross-correlation based clustering and dimension reduction of multivariate time series. 2017 IEEE 21st International Conference on Intelligent Engineering Systems (INES), 2017, pp. 000241–000246. doi:10.1109/INES.2017.8118563.
27. Izakian, H.; Pedrycz, W.; Jamal, I. Fuzzy clustering of time series data using dynamic time warping distance. *Eng. Appl. Artif. Intell.* **2015**, *39*, 235–244. doi:10.1016/j.engappai.2014.12.015.
28. Herzig, K.; Just, S.; Zeller, A. It's not a bug, it's a feature: how misclassification impacts bug prediction. 35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013; Notkin, D.; Cheng, B.H.C.; Pohl, K., Eds. IEEE Computer Society, 2013, pp. 392–401. doi:10.1109/ICSE.2013.6606585.
29. Tsuda, N.; Washizaki, H.; Honda, K.; Nakai, H.; Fukazawa, Y.; Azuma, M.; Komiyama, T.; Nakano, T.; Suzuki, H.; Morita, S.; Kojima, K.; Hando, A. WSQF: comprehensive software quality evaluation framework and benchmark based on SQuaRE. Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2019, Montreal, QC, Canada, May 25-31, 2019; Sharp, H.; Whalen, M., Eds. IEEE / ACM, 2019, pp. 312–321. doi:10.1109/ICSE-SEIP.2019.00045.
30. He, P.; Li, B.; Liu, X.; Chen, J.; Ma, Y. An empirical study on software defect prediction with a simplified metric set. *Inf. Softw. Technol.* **2015**, *59*, 170–190. doi:10.1016/j.infsof.2014.11.006.