RADig-X: a Tool for Regressions Analysis of User Digital Experience

Federico Di Menna
University of L'Aquila
L'Aquila, Italy
federico.dimenna@graduate.univaq.it

Vittorio Cortellessa
University of L'Aquila
L'Aquila, Italy
vittorio.cortellessa@univaq.it

Maurizio Lucianelli
MICRON
Avezzano, Italy
mlucianelli@micron.com

Luca Sardo
MICRON
Vimercate, Italy
lsardo@micron.com

Luca Traini
University of L'Aquila
L'Aquila, Italy
luca.traini@univaq.it

***Abstract*—The successful operation of a modern company relies on the dependability of its software infrastructure. However, ensuring a robust and dependable software infrastructure can be challenging, as software applications are subject to continuous updates that can introduce bugs and performance regressions. To mitigate this challenge, many companies use Application Performance Management (APM) tools to monitor their digital devices and identify potential issues that could affect business operability. However, the large volume and heterogeneity of the data collected by these tools can make it difficult to effectively analyze and exploit the rich source of information available. In this paper, we propose *RADig-X*, a tool designed to support the identification and analysis of digital experience issues. *RADig-X* leverages AI algorithms and a ranking heuristic to: (i) detect anomalies in runtime metrics collected by APM tools, (ii) assess the relevance of these anomalies based on their impact on the overall IT infrastructure, and (iii) rank problematic software updates that may be the root cause of relevant anomalies. We report on the adoption of *RADig-X* by a large company that monitors over 30,000 digital devices around the world. Our results demonstrate that *RADig-X* is able to improve the effectiveness of the identification process of digital experience issues, by enabling to identify and address potential anomalies that could impact business operations. *RADig-X* is currently used in production within the case company to support the diagnosis and problem resolution of digital experience issues.**

***Index Terms*—AIOps, Anomaly Detection, Application Performance Management**

I. INTRODUCTION

Modern enterprises heavily rely on digital devices, such as desktop computers, to carry out their business operations. Therefore, a dependable IT infrastructure is essential to ensure seamless business continuity. However, avoiding significant software problems that can hinder business activities presents an ongoing challenge for companies. Due to the current fast-to-market trend [1], modern software applications are continuously updated to meet new requirements and to experiment new features, and the time reserved for quality assurance activities during software development is often limited [1], [2]. In that, each software update poses a potential source of issues that can directly impact business digital devices. For instance, recent studies suggest that seemingly non-invasive software changes, such as maintenance activities, can unexpectedly compromise software quality [3], [4].

To deal with this context, companies often rely on Application Performance Monitoring (APM) tools [5] to monitor the execution behavior of software applications (*e.g.,* Dynatrace[1], Datadog[2], Aternity[3]). These tools continuously record metrics related to different aspects of software execution in the field (*e.g.,* fault rates, logging, execution times), and enable the adoption of quick remediation actions when relevant software failures occur [5]. Despite the utility of these tools, the effective exploitation of collected metrics for diagnosis and resolution of relevant software issues remains challenging. Software metrics are known to be subject to severe fluctuations (*e.g.,* due to varying workloads [6] and/or execution environments [7], [8]), and it is often difficult to determine to what extent these fluctuations are "normal" or symptoms of actual anomalies. Furthermore, anomalies are not uncommon in practice, and it can be challenging to assess the significance of an anomaly, thus to determine which ones should undergo root cause analysis.

In this paper, we present *RADig-X* (Regression Analysis of User Digital Experience), a tool to support the identification and analysis of digital experience issues, and we report on our experience in the adoption of this tool in a large real-world company, namely Micron, which monitors more than 30,000 digital devices around the world. *RADig-X* leverages a combination of AI algorithms and a ranking heuristic to: (i) determine actual anomalies in runtime metrics gathered from APM tools, (ii) prioritize the relevance of anomalies

This article has been accepted for publication in the *Industry Track* of the *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2024).*
This document is the postprint version of the article.
Citation information: DOI 10.1109/SANER60148.2024.00043

[1]Dynatrace. https://www.dynatrace.com
[2]Datadog. https://www.datadoghq.com
[3]Aternity. https://www.aternity.com

according to their impact on the overall IT infrastructure, and (iii) rank problematic software updates that are correlated with relevant anomalies. Through an empirical evaluation using real-world monitoring data, we demonstrate that *RADig-X* achieves a 94% improvement in F1-score for anomaly detection compared to the current practice employed within the company. Furthermore, our results suggest that *RADig-X* has potential in identifying software updates that exhibit a notable correlation with software application crashes. *RADig-X* is currently deployed within Micron to support the diagnosis and problem resolution of digital experience issues. To date, *RADig-X* has facilitated the detection of multiple regressions, including two specific ones that had substantial impacts on the digital user experience. Since the adoption of *RADig-X*, there has been a 14.66% enhancement in the index measuring the quality of digital experiences within the company.

The rest of the article is structured as follows. Section II presents the industry context of this work. Section III describes our formative study and the main challenges that we aim to tackle. Section IV outlines the main components of *RADig-X*. In Section V, we present our research questions along with experimental design and results. Section VI discusses the insights from the usage in practice of *RADig-X*. Section VII presents the threats to validity of our study, Section VIII presents the related work, and Section IX concludes this paper.

II. Context of this work

Micron is a leader company in memory solutions manufacturing. As of 2023, it is the 4th largest semiconductor company in the world, with a presence in 17 countries and a workforce of ∼49,000 employees.

Micron employees make an extensive use of digital devices (*e.g.,* desktop computers) to carry out their business activities. However, these devices are subject to frequent software updates that make them susceptible to unexpected performance degradations and/or software failures. These software issues can have a severe impact on employee productivity, such as causing a slowdown in business activities and potential economic effects for the company.

To mitigate the impact of these issues, Micron closely monitors more than 30.000 business devices using a popular APM tool[4]. A dedicated monitoring team is in charge of promptly identifying software issues that can have a significant impact on employee productivity. Each of the 30,000+ devices is equipped with a *APM tool* agent that records a variety of runtime metrics, such as response time of user actions, boot times, software crashes and resource usage, thus totaling to several gigabytes of data per day. These metrics can then be accessed either directly by the monitoring team through visual dashboards, or programmatically through REST APIs to observe the state of each device. Due to the vast volume and variety of metrics collected by APM tool, it is often difficult to gather a comprehensive picture of the health status of the company IT infrastructure. To this aim, Micron relies on the APM tool's *IDX* that summarizes, in a unique metric, the observed quality (the higher *IDX* the better quality) of the employee's digital experience. *IDX* provides two key benefits: (i) it offers a unique metric that acts as a proxy measure for the health status of the entire IT infrastructure, and (ii) it allows for comparison with *IDX* of other similar companies through a dedicated dashboard, thereby providing useful reference baselines. *IDX* can be configured to suit the specific business needs of a company, for example by assigning larger weight to the runtime metric of a particular software application. *IDX* calculation considers as input a subset of runtime metrics collected by the APM tool agent on the devices; based on these, a local *IDX* value (*IDX*-subscore) devoted to each precise runtime metric considered is calculated. These local *IDX* values are subsequently combined according to the configuration established by the company to arrive at a unique final *IDX* score (global *IDX* score). However, it can be challenging to understand how fluctuations in a specific runtime metric (*e.g.,* number of crashes in a particular software application) will ultimately affect *IDX*. Indeed, the process of deriving the *IDX* sub-scores starting from the raw metrics is often obscure for the company, since its computation also considers the metrics associated to other APM tool customers, which might not be available to the company. In addition, *IDX* has a fixed refresh rate, and therefore it is not directly accessible in real-time by the company. This can be a problem when, for example, a (buggy) software update is deployed on a significant number of devices, and the monitoring team needs to promptly detect any *IDX* drops to prevent negative effects on employee operability.

The monitoring team continuously monitors both raw metrics and *IDX* to spot anomalies through APM tool dashboards. Nonetheless, the frequency of software updates on employee devices poses a persistent threat that can prevent employee operability at any given moment. Software updates in Micron devices can be triggered directly by the employees or forced by the company. The latter are typically scheduled upfront and deployed on a large number of devices to maintain enterprise software up-to-date across all the company devices (*e.g.,* due to security reasons). In such cases, potentially problematic software updates can be deployed over a large amount of devices, with significant consequences on the entire IT infrastructure. To prevent updates from causing failures on systems, the mass release phase is usually preceded by the test deployment phase, where a subset of devices is randomly chosen to assess the potential impact of the updates. Despite these mitigation strategies, Micron often struggles to avoid side-effects due to software updates, especially due to the scale of the problem. About 70% of devices contain more than a thousand installed applications, and software change events per month are in the order of millions, including installations, upgrades and removals.

Figure 1 presents the digital experience monitoring process adopted in Micron. *APM tool* agents deployed on employee devices continuously record in a data storage software/system

[4]For privacy reasons, we name as *APM tool* the one used here, and as IDX the adopted digital experience quality index.

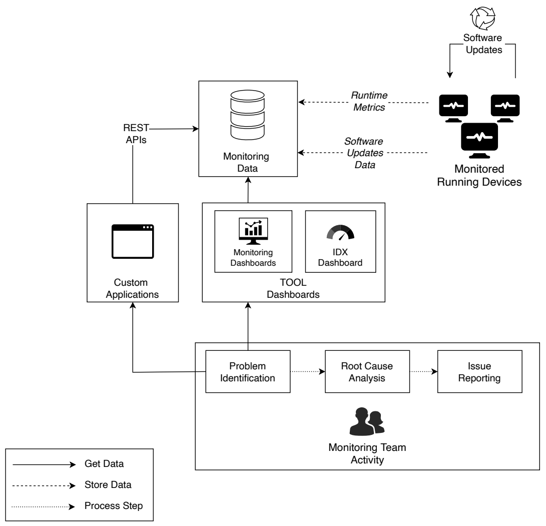

Fig. 1: Workflow of Digital Experience Monitoring Process

runtime metrics, and software changes (*i.e.,* software installations, updates and removals). Then, the monitoring team accesses the recorded data mainly through *APM tool* dashboards, which provide a collection of facilities to query and visualize software runtime metrics and software changes data. In addition, the monitoring team has developed a series of custom applications, on top of the APM tool APIs, to enable automated queries and analyses, such as automating metric checks to trigger an alert if the number of software crashes increases over a threshold.

The monitoring team is in charge of three main activities within Micron: (i) *problem identification* , (ii) *root cause analysis*, and (iii) *issue reporting*. *Problem identification* entails understanding if there are significant anomalies in the monitored runtime metrics that can negatively impact employee productivity. *Root cause analysis* involves extracting the potential causes of a manifested runtime metric anomaly (*e.g.,* buggy software updates or installations), while *issue reporting* consists in reporting the identified anomalies and the associated possible causes to the team in charge of managing the entire IT infrastructure.

III. Formative study

We conducted an initial formative study to understand the challenges faced by the monitoring team. The first author spent the first two months of the study on-site, together with the monitoring team, to directly experience the activities and challenges faced by the team. In the first two weeks, daily training meetings were held on the main context-related concerns, such as: team organization, used monitoring tools, IT infrastructure and support applications. Afterwards, the first author actively participated in the daily activities carried out by the monitoring team, such as problems identification of ongoing *IDX* regressions and root cause analysis, for a total training period of two months. At the end of this phase, a discussion with all the co-authors has been held in order to extract the main concerns.

We distilled three main challenges: (i) *anomaly detection*, (ii) *digital experience impact evaluation* and (iii) *root cause analysis*.

Anomaly detection. The monitoring activity of devices involves ongoing tracking of application quality metrics, *e.g.,* application crashes and wait times, which results in time series of metric values concerning applications that are spread across a wide range of devices. Digital experience regressions occur when the system behavior is changing unexpectedly. Hence, some of metric time series may present anomalies that differentiate the running state from the past usual patterns. In that, a major challenge consists in detecting anomalies within metric time series, by distinguishing between seasonal fluctuations in recorded metrics and true anomalies. Examples of seasonal fluctuations are due to device running hours. For instance, less application crashes can be due to many idle or powered-off devices that are not running any application (*e.g.,* for different working hours in different world time zones). These kinds of fluctuations do not impact employees user experience, so they should not be reported.

Current internal practices mostly rely on human-driven analysis, which can be expensive given the large number of metrics that affect *IDX*.

Anomaly impact evaluation. Industry processes lead to frequently evolving changes to infrastructure and software that can bring anomalous situations due to bugs and conflicts. However, even if every anomaly represents a potential impact to digital experience, it might not have a relevant effect on employees. The anomalies of interest are in fact those that affect a large number of devices, thus impacting a wide number of users, and those that persist over time, consequently having a continuous impact on *IDX*. In this perspective, it is necessary to quantify the digital experience impact as soon as anomalies are identified, and to prioritize the ones that most negatively affect the *IDX* score. In this way, anomalies can be classified according to their *IDX* impact.

Since *IDX* is a third-party index with a fixed refresh rate, it is not possible to immediately know the *IDX* score from raw metrics recorded values in a critical situation, so the impact assessment task of an anomaly is a major challenge. In addition, a related concern consists in estimating *IDX* impact of anomalies in near future, thus allowing for proactive corrective actions, before regressions can have devastating impacts.

Root cause analysis. A third challenge consists in analyzing the causes of the identified anomalies threatening the employees digital experience. Software changes (*i.e.,* updates, installations and removals) are millions per month. This makes the identification of changes responsible for an anomaly an expensive task that requires wide analysis. Indeed, each of the 30,000+ devices has its own configuration involving thousands of installed applications, so that each one triggers specific software updates and different potential conflicts can come out.

In addition, the instant when a software change is performed on a specific application may vary from device to device. In that, the monitoring team deals with a huge amount of data to analyze. Currently, internal root cause analysis practices rely on expensive and manual human-based heuristics and statistical analysis.

IV. Proposed solution

Figure 2 presents an overview of *RADig-X*, a tool for regression analysis of employees digital experience, which includes three major components, each supporting a targeted challenge: (i) *Anomaly Detection*, (ii) *Anomaly Impact Estimation* and (iii) *Root Cause Analysis*.

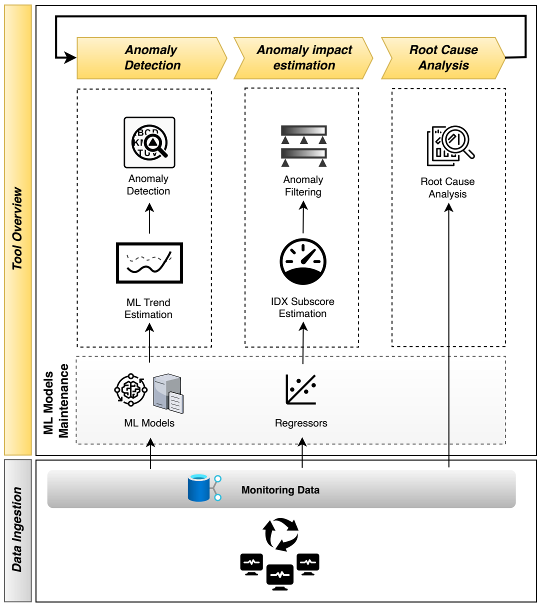

Fig. 2: Overview of *RADig-X*

The first component of *RADig-X* leverages machine learning models aiming to detect anomalies. Thereafter, linear regressors are used in the second component for estimating the digital experience impact of each identified anomaly of runtime metrics. The third component supports Root Cause Analysis (RCA) activity by analyzing, through heuristic ranking, the impact of software updates on the digital experience. Across the three components, the tool employs an underlying model maintenance layer that is responsible for updating employed models with incoming data.

The whole tool lays on a external *Data Ingestion* layer that contains all employees devices monitored data, which is continuously populated by a monitoring agent.

RADig-X offers a clear separation of concerns in terms of challenges addressed by individual components separately. Indeed, in the daily practice, *RADig-X* components can be used either sequentially, to automate the whole process, or individually to support a single challenge (*e.g.,* root cause analysis).

In the following we describe, in detail, how our solution addresses each of the detected challenges.

A. Anomaly detection

We employ time series ML forecasting algorithms (*e.g.,* LSTM) to model runtime metric fluctuations patterns based on the historical trends, as represented in Figure 2 (*ML Trend Estimation*). Specifically, one model has been trained for each time series of monitored runtime metric. In this way, we allow *RADig-X* to learn from the underlying seasonal patterns of each monitored runtime metric, thereby enabling the prediction of expected future values within a predefined forecasting horizon. In order to perform anomaly detection, predictions for expected runtime metric values are continuously generated by *RADig-X*, and the residuals between the expected and actual values are assessed to quantify the deviation from the expected time series behavior.

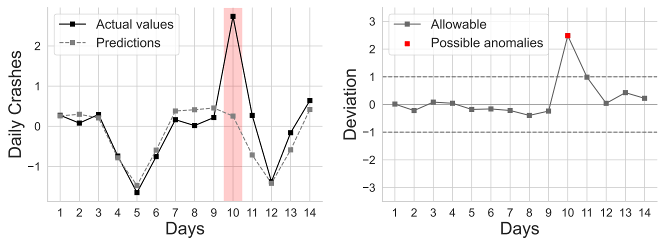

Fig. 3: Real example of plots included in *RADig-X* visual anomaly detection dashboard.

These deviations can be assessed in two different ways: (i) using a predefined threshold T, which enables automated anomaly detection, and (ii) using a dedicated dashboard that allows the monitoring team to visually compare the actual runtime metric trend with the expected one. Figure 3 shows an example of such visual analysis. On the left side, a solid line shows the real (standardized) number of crashes observed in two weeks, while a dashed line represents the predictions. The plot on the right side of the figure instead reports the deviation between the actual and predicted values. This example shows that in *day 10* there is a potentially impactful detour, and the monitoring team can decide whether to deem it as an anomaly based on their domain expertise.

B. Anomaly Impact Estimation

Our second challenge consists in pointing out anomalies with a relevant impact on digital experience. The capability to have a *IDX*-estimation strategy permits to assess the digital experience impact of runtime metrics anomalies without waiting for next *IDX* refresh, as motivated in Section III. In order to point out relevant anomalies, *RADig-X* aims to estimate the *IDX* subscore of each considered metric starting from runtime metric values. In that, the *IDX Subscore Estimation* component of Figure 2 relies on linear regressors fit with the past observed data using raw metric values as input and their associated *IDX* subscores as target values. This technique allows to provide an estimation of *IDX*, while ignoring the details behind its calculation.

As reported in Section II, global *IDX* score is based on a subset of runtime metrics, each one associated with a *IDX* subscore. Taking into account the fact that global *IDX* derives from several runtime metrics distinct in their characteristics, our solution estimates *IDX* subscore of each individual metric separately, and then performs the aggregation through a weighted average according to the *IDX* configuration established by the company. Indeed, each metric may have different units of measurement (*e.g.,* time units, percentage, etc.) and may be influenced by different factors (*e.g.,* device performance, user behavior, etc.). In that, *RADig-X* aims to estimate the impact of runtime metric anomalies on digital experience, by means of the global *IDX*. The *Anomaly Filtering* component, represented in Fig. 2, uses the impact estimation to filter out negligible anomalies (with a very low impact).

Furthermore, when used in tandem with the ML models presented in the previous section, this approach can also be used to predict future estimations of the global *IDX*, and estimate the future impact on digital experience of current runtime metric fluctuations.

C. Root Cause Analysis (RCA)

The huge amount of data collected from monitored devices makes the root cause analysis (currently conducted manually) a labour-intensive and time-consuming task. Our proposed solution implements a heuristic ranking, by automating a before-and-after analytical process on all devices. This step is represented in Figure 2 with the *Heuristic Ranking* component. This process is aimed at automatically narrowing down the many possible causes to a few changes, with respective quantitative and qualitative impact measures. A major challenge consists of managing devices with heterogeneous software configurations, which execute updates in different instants determined by the user.

Figure 4 shows a high-level description of *RADig-X* Root Cause Analysis (RCA). The analysis starts from the anomaly report generated by the previous tasks containing: (i) the time interval in which an anomaly has been observed, (ii) the software application that is crashing, and (iii) all the details related to the devices that are affected. *RADig-X* extracts all the data related to software changes and software crashes collected by the agent installed on the devices, basing on the time interval in the anomaly report. In particular, software changes data are extracted as a dataset, in which each row corresponds to a software change event and specifies: device ID, software name, software version and timestamp. Similarly, software crashes data are collected as a tabular dataset, where each row identifies a software crash recorded by the agent, and it specifies: application name, device ID and crash timestamp. We also include a filtering step to possibly remove non-relevant data (*e.g.,* changes concerning very few devices).

Upon completion, the tool executes a loop running through all the unique software changes contained in the software changes dataset. Within each iteration (*i.e.,* each unique software change), a nested loop is performed over all the devices that received that specific change. For each device, *RADig-X* splits the crashes dataset of that device in two parts, namely: before SC subset and after SC subset. Statistical metrics are then carried out, for each software change, by comparing before/after events. Finally, generated data are sorted out on a parameter that summarizes the global impact of the software changes on all the devices.

V. Empirical Evaluation

We independently evaluated the effectiveness of each *RADig-X* sub-component (*i.e., anomaly detection*, *anomaly impact estimation* and *RCA*), by aiming to address the following research questions:

- **RQ1**: How effective is *RADig-X* in detecting anomalies?
- **RQ2**: How effective is *RADig-X* in assessing the relevance of anomalies in terms of employees digital experience?
- **RQ3**: How effective is *RADig-X* in supporting root cause analysis?

The results of our empirical evaluation are reported in the following subsections, each one dedicated to a specific research question.

A. RQ1: How effective is RADig-X in detecting anomalies?

To address this research question, we first assess the feasibility of different machine learning models in predicting time series of runtime metrics. Then, we evaluate the effectiveness of these ML models for anomaly detection, through a comparison with the current internal practice.

To select *RADig-X* ML model for performing anomaly detection, we tested the effectiveness of two widely-used Recurrent Neural Networks (RNN) models, namely *bidirectional Long Short-Term Memory (LSTM)* [9], [10] and *Gated Recurrent Unit (GRU)* [11], employing *Random Forest Regressor* (RFR) [12] and *Linear Regression* (LR) as baseline models for comparison, since they have been widely employed for time series forecasting [13]–[17]. To construct the evaluation dataset the monitoring team has suggested eight metrics concerning application crashes with higher weights in *IDX* configuration, and so expectedly most impactful on *IDX* in case of regressions. We extracted more than one year of data with a sampling interval of one day for each metric, generating eight time series with *400* values. We then formulated time series forecasting task as a supervised learning regression problem. The dataset has been crafted creating consecutive shifted windows, each with a width of 28 time steps, for each time series. For each window, the first 14 time steps are used as input and the last 14 values as output. We adopted a multiple-output strategy for addressing the multi-step-ahead problem, thus directly predicting all the forecasting horizon [10], [18]. For each windowed dataset that has been generated, first 80% of windows are used for training (of which 10% for validation set) and the last 20% as testing set.

In order to identify the most effective ML approach for our dataset, we leveraged two widely used forecasting accuracy measures: (i) Mean Absolute Error (*MAE*) and (ii) Root Mean Squared Error (*RMSE*) [15], [19], [20] over the testing set. We have implemented *LSTM* and *GRU* using *Keras*. Linear

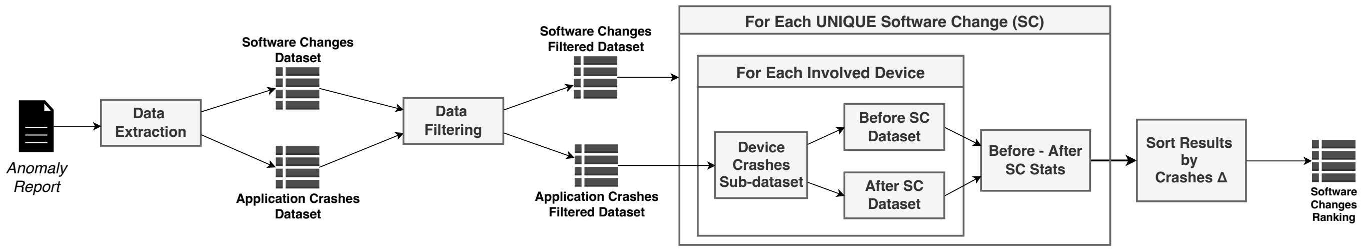
Anomaly Report
Data Extraction
Software Changes Dataset
Application Crashes Dataset
Data Filtering
Software Changes Filtered Dataset
Application Crashes Filtered Dataset
For Each UNIQUE Software Change (SC)
For Each Involved Device
Device Crashes Sub-dataset
Before SC Dataset
After SC Dataset
Before - After SC Stats
Sort Results by Crashes Δ
Software Changes Ranking

Fig. 4: *RADig-X* Root Cause Analysis (RCA) process diagram

Regression and Random Forest Regressor models have been implemented by using *scikit-learn*.

The effectiveness of *RADig-X* anomaly detection has been evaluated over the testing set of the same time series. For sake of such comparison, we have implemented an automated script to replicate the current anomaly detection practice used within the company. Specifically, current internal practice estimates future metric values computing the mean value based on the specific weekday past values. For instance, expected metric value for the next Monday is computed averaging values recorded on previous Mondays, considering a time window defined by taking into account retention policies on monitored data. Daily anomaly detection is then performed by comparing the expected values and the real observed ones with threshold-based criteria.

We computed *Precision*, *Recall* and *F1-score* using as ground-truth actual anomalies occurred on the selected time series, as reported by the monitoring team.

RADig-X and current internal practice anomaly detection are both performed by comparing the difference among expected and observed values against a threshold. To fairly compare them we used the same threshold, setting it equal to the standard deviation σ of the input sub-sequence: $T = \sigma$. We chose this conservative threshold based on the feedbacks of the monitoring team. Indeed, the monitoring team tends to consider false negatives more dangerous than false positives. In this context, an anomaly that goes unnoticed can be more damaging than a false positive, which instead might be double checked by the team and ignored afterwards.

Results. Figures 5 and 6 depict the RMSE and MAE distributions for each ML approach. Given the presence of outliers in error distributions and the variety of applications considered, we preferred to use the median values to compare the approaches. We noticed that RNN models exhibited favorable MAE distributions, with lower mean and median with respect to LR and RFR. Specifically, in Fig. 6 we observed a median of 0.071 for LSTM and 0.072 for GRU, while LR and RFR have shown median values of 0.083 and 0.093, respectively. By looking at Fig. 5, we again noticed that all RNN models exhibited a lower RMSE median compared to the baselines. Namely, RMSE median of 0.131 and 0.121 have been observed for LSTM and GRU, versus 0.137 of LR and 0.139 of RFR. In summary, both MAE and RMSE medians pertaining to GRU model are the lowest ones. In addition, the GRU error distribution is less spread out, thus suggesting a more reliable prediction. Based on these quantitative results, we have chosen LSTM and GRU models for the implementation of *RADig-X*, and therefore for the subsequent evaluation step.

In the second step of RQ1 evaluation, we compared *RADig-X* anomaly detection effectiveness against current internal practices. The evaluation has been performed over the same eight time series, by using anomalies manually labeled by the monitoring team as ground-truth. Results of anomaly detection evaluation are shown in Figure 7, where we present the distribution of *Recall*, *Precision* and *F1-score*, as grouped by approach. Specifically, we compared the following approaches: (i) current internal practice, (ii) LSTM-based *RADig-X* and (iii) GRU-based *RADig-X*. By examining the F1-score box-plots, we observed highest mean and median when using LSTM model, with a mean of 0.437 and a median of 0.408. Instead, GRU exhibited a median of 0.39 and mean of 0.30. With the current internal practice, we noticed a median F1-score value of 0.225 and a mean of 0.331. Thus, the median values show an advantage in using LSTM. Interestingly, even though the GRU model has shown a lower error during the previous model performance evaluation, the quality of *RADig-X* anomaly detection is better when using LSTM, which highlights a *median* F1-score improvement of **94%** compared to the current internal approach. By examining *precision* and *recall* separately, LSTM-based *RADig-X* has shown an improvement of **40%** in median *recall* compared with the current internal approach, suggesting that *RADig-X* detected considerably more true anomalies. In addition, a larger improvement in median *precision* (**100%** increasing) is also achieved, meaning that LSTM-based *RADig-X* significantly improved the number of actual anomalies among those identified. By considering the improvements presented above, we clearly observed that the proposed strategy improves the accuracy of the anomaly detection technique.

B. *RQ2: How effective is RADig-X in assessing the relevance of anomalies in terms of employees digital experience?*

Second RQ aims to address the effectiveness of *IDX* impact estimation of the identified anomalies.

Our strategy leverages linear regressors with a supervised learning approach, by using runtime metrics data as input and the corresponding *IDX* subscore as output, in order to estimate a global *IDX* value. We have considered all the runtime

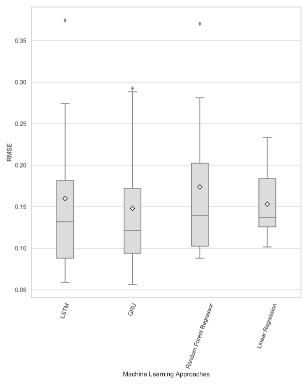

Fig. 5: RMSE of ML Models.

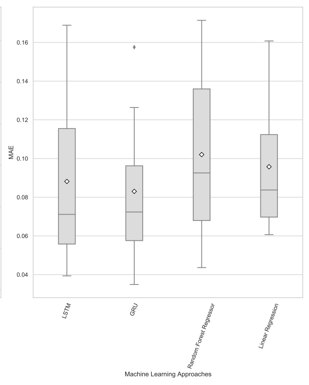

Fig. 6: MAE of ML Models.

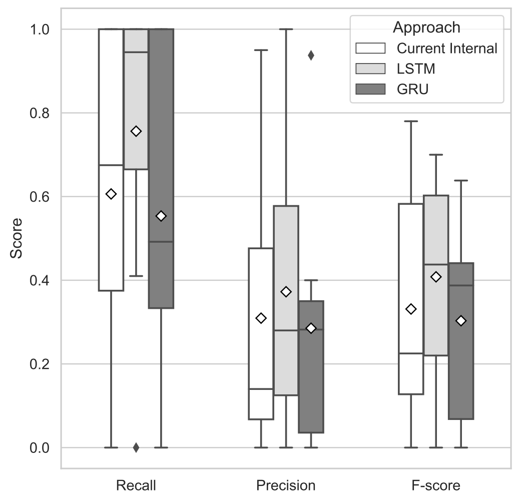

Fig. 7: Comparison of *recall*, *precision* and *F1-score* scores distribution for all anomaly detection approaches.

metrics accounted for *IDX* calculation (that are more than 60). Specifically, we built a dedicated linear regression model for each runtime metric, and performed a separate training for each one. We implemented linear regressors using *scikit-learn* in *python*. Each regressor has been trained using time series of runtime metric weekly values as input, and the corresponding *IDX* subscores time series as output. Linear regressors aim to predict *IDX* subscore of a specific runtime metric, starting from the metric weekly value. For each metric, the regressor has been fit using the first 80% of time series as input and the last 20% for testing. We used the *Root Mean Squared Error (RMSE)*, rescaled on a 0-100 scale using the minimum and maximum values of *IDX*, to evaluate the estimation quality of each runtime metric. In order to test the effectiveness of the global estimation, we have then aggregated all the metric *IDX* subscores to obtain the global *IDX* estimation. This outcome has been compared against the real global *IDX* score observed in the testing set. Predicted *IDX* subscores are aggregated according to the weights that the company assigned to each runtime metric in *IDX* configuration. Based on company requirements, a global *IDX* estimation percentage error less than 5% is acceptable.

Summing up, RQ2 evaluation has been conducted in two steps: (i) measuring linear regressors error to assess that the error distribution is acceptable across all the metrics, and (ii) evaluating the effectiveness of the aggregation technique of estimated scores by comparing the aggregated predicted score with the real global *IDX* score.

Results. Figure 8 shows the distribution of all the regressors absolute errors. We highlight that Q1 is very close to *0* and Q3 is less than *5*, *i.e.,* 75% of regressors have an error lower than *5*. Hence, the majority of regressors have a very small error. However, the plot also shows several outliers, thus suggesting that some components are too complicated to be modeled with a simple linear regressor.

Table I shows the global *IDX* estimation percentage error in the last *7* weeks. Global estimation error is always less than *2%*, thus meaning a suitable estimation technique based on company requirements.

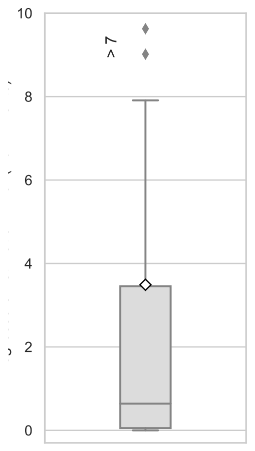

Fig. 8: *RMSE* distribution in *IDX* subscore estimation over all considered runtime metrics using linear regressors.

Testing Week	*IDX* Percentage Error
Week #1	0.33 %
Week #2	0.08 %
Week #3	0.54 %
Week #4	1.36 %
Week #5	0.39 %
Week #6	0.12 %
Week #7	0.79 %

TABLE I: Percentage error measured on weekly global *IDX* estimation.

C. RQ3: How effective is RADig-X in supporting root cause analysis?

The RCA strategy has been empirically evaluated by applying it in real-world scenarios, specifically in four cases where significant application crash anomalies occurred. The data we used for the evaluation consist, for each scenario, in application crashes and software change events, occurred in each device, which have been collected by APM tool agent. For each analysis, we have generated a ranking of software changes in a decreasing order of impact. Software modules and application names have been anonymized for privacy reasons. The naming convention is the following: software updates are indicated with the prefix $SU\_$, while applications that have experienced a crash regression are denoted by the prefix $CR\_$. Software update modules that pertain to the same application share the first letter after the prefix (*e.g.,* SU_A.1 and SU_A.2 in Tab. II are both related to the application A).

As for ranking metrics, we considered for each software change: (i) *Crashes Difference*, representing the relative difference between the total amount of crashes observed before and after the software change occurred on all the affected devices; (ii) *Devices*, which indicates the percentage of devices affected by the change; (iii) *Crashes Delta Avg*, calculated by averaging the difference between the crashes observed before and after the change of each affected device. The sorting of software changes ranking is performed in descending order based on *Crashes Delta Avg* value.

Results. In the first scenario, an application that we name as *CR_X* has experienced an increase of +450% in the number of crashes. In Table II, we reported the head of the ranking generated by *RADig-X* RCA. The top-ranked updates clearly depict major crashes difference increases (by more than +1,000%), all related to the same application *SU_A*.

Other two scenarios concern a common application that we name as *CR_Y*. One of them has manifested a notable crash regression in *CR_Y*, whereas in the other one there was a subsequent decrease in the number of crashes for the same application. The results are shown in Tables III (crash increasing phase) and IV (crash decreasing phase), respectively. By looking at *Crashes Difference* and *Crashes Delta Avg* column of Table III, we can easily infer that *SU_G.1* and *SU_G.2* updates result in a much stronger impact than other ones. Similarly, Table IV shows that the first row has a more significant crash delta average value (-22.18%) than the following ones. By looking at the software updates names, we find that these updates actually concern different modules of the same application update (*SU_G*), thus suggesting that *SU_G.1* and *SU_G.2* updates of Tab. III have caused regressions that might have been fixed by *SU_G.6* update in Tab. IV.

In the fourth scenario, which involved another crash regression, the heuristic ranking did not report any meaningful results.

In summary, in three out of the four scenarios the tool has highlighted a subset of software changes that may have caused the crashes regression. However, after consulting updates support team, we discovered that only in the first scenario (Table II) the software change that had really caused the regression has been highlighted. In contrast, the regressions in the other cases were attributed to particular workload distributions in the industrial setting and not to specific software changes. Hence, the approach has proved its usefulness only in the first case. One critical issue that has been found is the distortion of results on some software changes that occur simultaneously with the actually responsible one. Since the approach counts events before and after updates, if a group of changes is made at the same time then the consequences of one change belonging to the group could bias the results of the whole group. On the basis of the above consideration, we can conclude that the approach succeeds in readily filtering out the subset of software changes that impacted the devices most significantly, but the identified changes may not be the real culprits since crashes regression might be mainly due to some other reasons.

Software Update	Crashes Difference	Devices	Crashes Delta Avg
SU_A.1	+ 1205.27 %	58.59 %	5.49 %
SU_A.2	+ 1205.27 %	58.59 %	5.49 %
SU_A.3	+ 1209.29 %	58.62 %	5.49 %
SU_A.4	+ 1205.27 %	58.59 %	5.49 %
SU_A.5	+ 1206.36 %	58.59 %	5.49 %
SU_A.6	+ 1201.99 %	58.57 %	5.48 %
SU_A.7	+ 1206.78 %	58.43 %	5.46 %
SU_D.1	+ 345.99 %	80.20 %	4.03 %
SU_E.1	+ 266.42 %	100.00 %	3.45 %
SU_E.2	+ 233.08 %	89.00 %	3.37 %
SU_E.3	+ 146.55 %	61.28 %	3.02 %
SU_F.1	+ 30.28 %	95.79 %	1.01 %
...			

TABLE II: *RADig-X* root cause analysis evaluation: ranking for CR_X crash increase.

Software Update	Crashes Difference	Devices	Crashes Delta Avg
SU_G.1	+ 189.60 %	74.87 %	52.10 %
SU_G.2	+ 199.23 %	98.04 %	39.11 %
SU_G.3	+ 40.81 %	69.41 %	8.34 %
SU_G.4	+ 27.23 %	72.19 %	4.97 %
SU_G.5	+ 27.11 %	72.61 %	4.96 %
...			

TABLE III: *RADig-X* root cause analysis evaluation: software updates ranking scenario 2 (CR_Y crash increase).

Software Update	Crashes Difference	Devices	Crashes Delta Avg
SU_G.6	- 51.24 %	92.43 %	- 22.18 %
SU_G.7	- 20.92 %	76.80 %	- 5.78 %
SU_G.8	- 17.28 %	95.71 %	- 4.63 %
...			

TABLE IV: *RADig-X* root cause analysis evaluation: software updates ranking scenario 3 (CR_Y crash decrease).

VI. Insights from practice

RADig-X has been tested and deployed in Micron, used daily by monitoring team analysts. We report two success cases collected thus far from the adoption of the tool within the company, focusing on how *RADig-X* assisted the monitoring team in managing runtime metrics anomalies that have affected *IDX*.

In either cases, a first anomalous increasing phase in application daily crashes occurred followed by a later decreasing phase. Thanks to the visual dashboard provided by *RADig-X* (that includes plots like the ones shown in Figure 3), the monitoring team can double check and analyze the anomaly, observing more details about the deviation between expected and observed values to perform in-depth analyses. Both cases start with an anomaly reported by *RADig-X*, which is immediately analyzed by the monitoring team via the visual dashboard. Right away, by leveraging *RADig-X*, the real-time impact of the anomaly on *IDX* is estimated (without waiting for the next *IDX* refresh), thus allowing the monitoring team to immediately know the impact of the anomaly on the digital experience. Then, the extraction of crashes and software changes data collected during the anomaly-affected week is performed to generate the heuristic ranking with *RADig-X*, thus filtering the most impactful software changes.

We summarized quantitative data of two cases in Tables V and VI, showing the observed deviation calculated as the absolute percentage error between the collected actual values and the predicted ones in the metric time series. The error is computed on both the sum and the mean of the metric daily values. In addition, we report the anomaly percentage impact on the *IDX* metric subscore that incurred the anomaly.

Case 1: Results concerning the first case are presented in table V. *RADig-X* anomaly detection dashboard showed a difference of more than 60% between the expected and observed crashes of a running application *APP1*[5]. Such immediate recognition of the anomaly allowed for preliminary investigations to figure out the underlying concern. By comparing the total amount of *APP1* crash events occurred in all the devices in the week before the anomaly with respect to the week including the anomaly, the monitoring team noticed a 400% increase. By more closely looking at the devices that are plagued by the anomaly, it turned out that the problem was distributed on almost all the devices of the company, thus impacting the digital experience of many employees. Indeed, *RADig-X* estimated an impact to *APP1 IDX* subscore of 25.23%. Thereafter, *RADig-X* RCA ranking highlighted an impactful software change *SC1*. By summing all the application crashes observed in each individual device after this software change, it has been found that the number of *APP1* crashes after *SC1* was 16 times the number of crashes before the change (considering a one week length window before and after the change). The *APP1* anomaly and the likely cause *SC1* have been immediately reported to support groups to fix the issue. After three months, the *RADig-X* signaled another *APP1* anomaly concerning a decreasing trend in the amount of daily crashes of the same application. The gap between predicted and observed was about 35%. Even though this situation was supposed to increase *IDX*, by exploiting *RADig-X* the team noticed that the increase was not enough to raise the *APP1 IDX* subscore. *RADig-X* RCA ranking has shown a potentially critical software change *SC2* (which was a module of *APP1* itself). By comparing the sum of daily crashes occurred in all the devices before and after *SC2* (considering a one week period), the number of *APP1* crashes decreased by 30.2%.

[5]For privacy reasons, we cannot disclose the name of the actual applications and software changes.

	Observed deviation (%)		
	Sum	Mean	*IDX* Subscore
Regression Phase	64.51 %	64.50 %	- 25.23 %
Improvement Phase	35.99 %	36.03 %	0 %

TABLE V: *RADig-X* practical usage case 1 results.

Case 2: The second situation is reported in Table VI. By using the tool, the monitoring team identified an anomaly in *APP2* daily application crashes owing to an unusual growth pattern for observed crashes. The average gap between daily expected and observed values was over 200%. Then, *RADig-X* estimated a *IDX* subscore impact of almost 50%, thus allowing the monitoring team to be aware in real-time of the impact of the anomaly on the employees digital experience. *RADig-X* RCA, in this case, didn't highlight any specifically related impacting software update. After two months, *RADig-X* detected a decreasing in daily *APP2* crashes, thus signaling the anomaly to the monitoring team. On average, the percentage error among expected and observed daily crashes decreased by 11.11%. By using *RADig-X*, the monitoring team immediately estimated an improvement in *APP2 IDX* subscore of 69.37%.

In these two situations we observed the real utility of the tool in a practical context. Indeed, the monitoring team confirmed that these two situations consistently impacted the employees digital experience. Thanks to *RADig-X*, it was possible to: (i) immediately identify runtime metrics anomalies, (ii) estimate in real-time the digital experience impact without waiting for *IDX* refresh, and (iii) identify updates that exhibit a stronger correlation with the metric anomaly. In addition, we observed the effectiveness of *RADig-X* in detecting anomalies not only in the case of *IDX* regressions but also in the case of improvements. As a matter of fact, albeit they do not have a negative impact, they are still of interest to the team.

The context of the tool concerns third-party applications analysis (in both anomaly detection and software changes analysis), therefore it is not possible to investigate in detail which types of software modifications (refactoring, integration or others) led to crashes and which code fragments have been affected.

	Observed deviation (%)		
	Sum	Mean	*IDX* Subscore
Regression Phase	236.25 %	236.34 %	- 48.9 %
Improvement Phase	58.46 %	11.11 %	+ 69.37 %

TABLE VI: *RADig-X* practical usage case 2 results.

We also measured the global *IDX* since the adoption of *RADig-X*, resulting in a 14.66% improvement. In addition, the monitoring team remarked that the use of the system automatically implies more focus on the monitored data, leading to a greater sensitivity to any variation.

VII. Threats to Validity

a) Construct validity: We evaluate the effectiveness of each component of *RADig-X* independently from the other ones. The interactions among the different *RADig-X* components might affect the approach effectiveness. Unfortunately, due to APM tool retention policies, we were unable to conduct a comprehensive end-to-end evaluation of the approach. Nonetheless, in Section VI we report some of the benefits of employing *RADig-X* in practical scenarios. We have replicated the current practice used within the company through an automated script. Although this script may not precisely represent the monitoring team behavior in practice, it serves as a valuable baseline for evaluating *RADig-X*.

The anomaly detection component of *RADig-X* has been evaluated on eight distinct time series of runtime metrics. The effectiveness of *RADig-X* may vary when applied to other runtime metrics. Unfortunately, the selection of the time series used in this evaluation has been constrained by the retention policies of APM tool and the necessity of a ground-truth, *i.e.,* anomalies manually labeled by domain experts.

b) Internal validity: We evaluate *RADig-X* using well-established metrics, such as MAE and RMSE. Using different evaluation metrics could impact the interpretations of the results. *RADig-X* determines the presence of an anomaly based on a specific threshold T. The outcomes of *RADig-X* may vary if different threshold values are applied. To ensure a fair comparison, we have based our comparison between *RADig-X* and the baseline on the same threshold.

c) External validity: We use *RADig-X* within the context of Micron in combination with a specific APM tool. However, we designed *RADig-X* with a focus on generalizability, as we aim to facilitate its application to other contexts. Indeed, if the semantics of the data is the same as the one used in the Micron context, then the effort to adapt this approach would be related only to the data formatting. Of course, its effectiveness may be affected by the context in which it is used.

VIII. Related work

Software Anomaly Detection. In recent years, researchers have increasingly utilized machine learning techniques alongside traditional statistical approaches (*e.g.,* ARIMA models [21], [22]) for software anomaly detection [23]–[28]. Among the most commonly employed models are Recurrent Neural Networks [25], [29]–[31], such as LSTM and GRU. For instance, LSTM have been employed to predict failures in cloud service systems [32], or spacecraft anomalies [24]. LSTM have been also combined with other techniques such as Variational Auto-Encoder (VAE) [33] and multimodal learning [34] to perform this task. Other methods utilize other unsupervised machine learning approaches to detect anomalies [35]–[37]. Chen et al. [38] use pattern sketching to detect anomalies in online service system, a technique that provides better result interpretability when compared black box approaches. Most of the prior work apply anomaly detection to individual (in-house) software applications. Our work, instead, involves dealing with a large IT infrastructure that encompasses a variety of third-party software applications spanned across more 30,000 devices.

Software Changes. Software quality degradation may be induced by software changes (such as bug fixes, refactoring, functionality extension). For this reason, over the last decade, researchers have been committed in devising techniques to mitigate the impact of software changes. These techniques can be preventive, *i.e.,* before the changes are deployed into production to evaluate potential risks [39]–[42], or post-deployment [25], [43]–[45]. Our approach fits into this last category. Existing approaches of this category are based on examining variations on quality metrics before and after deployment. For instance, Lumos [46] adopts an A/B testing approach (similarly to our heuristic) while FUNNEL [45] leverages singular spectrum transform (SST) algorithm. Other approaches such as Gandalf [44] and SCWarn [25] have been realized to take into account data monitored from multiple sources, such logs and a variety of KPIs. Much of these approaches are focused on analyzing software changes related to a specific service or application. Our study, instead, proposes an analysis on software changes related to third party applications, considering the software change as a black box event.

Root Cause Analysis. Root cause analysis (RCA) of software systems has been extensively researched in a variety of contexts. For instance, DeLag [47] automatically detects deviations in the execution time of Remote Procedure Calls (RPC) that are correlated with performance degradation, in the context of service-based systems. In the same domain, automated pattern detection has been widely investigated for root cause analysis of performance issues and/or failures [48]–[50]. In StackMine [51], pattern detection has been employed to detect callstack patterns associated with performance issues. Chen et al. [52] proposed a failure cause diagnosis approach using decision trees trained over request traces in which there are failures. *RADig-X* seeks to identify software changes that are associated to significant digital experience regressions.

IX. Conclusion

In this work, we have introduced *RADig-X*, a novel approach to enhance the diagnosis and root cause analysis of digital experience issues. We have conducted an empirical evaluation on real-world monitoring data gathered from a large company, involving over 30,000 devices. The results have shown that *RADig-X* can be effectively used to (i) detect anomalies in runtime metrics, (ii) assess the relevance of anomalies with respect to their impact on the global *IDX*, and (iii) identify software updates that are likely culprits of anomalies. Since its deployment, *RADig-X* has helped to identify problematic software updates that have shown relevant impact on the digital experience of Micron employees. As a future work, we intend to broaden the assessment of *RADig-X* to incorporate other types of runtime metrics, such as those related to software performance. Furthermore, we plan to evolve *RADig-X* beyond its current capabilities. In particular, we plan to shift towards a multi-feature approach in the anomaly detection task, and to include the capability of proactively identifying potential problems with software updates before they are widely deployed across numerous digital devices.

References

[1] J. Rubin and M. Rinard, "The challenges of staying together while moving fast: An exploratory study," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 982–993. [Online]. Available: https://doi.org/10.1145/2884781.2884871

[2] L. Traini, "Exploring performance assurance practices and challenges in agile software development: An ethnographic study," *Empirical Software Engineering*, vol. 27, no. 3, p. 74, 2022. [Online]. Available: https://doi.org/10.1007/s10664-021-10069-3

[3] M. Di Penta, G. Bavota, and F. Zampetti, "On the relationship between refactoring actions and bugs: A differentiated replication," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 556–567. [Online]. Available: https://doi.org/10.1145/3368089.3409695

[4] L. Traini, D. Di Pompeo, M. Tucci, B. Lin, S. Scalabrino, G. Bavota, M. Lanza, R. Oliveto, and V. Cortellessa, "How software refactoring impacts execution time," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 2, dec 2021. [Online]. Available: https://doi.org/10.1145/3485136

[5] T. M. Ahmed, C.-P. Bezemer, T.-H. Chen, A. E. Hassan, and W. Shang, "Studying the effectiveness of application performance management (apm) tools for detecting performance regressions for web applications: An experience report," in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, 2016, pp. 1–12.

[6] D. Ardelean, A. Diwan, and C. Erdman, "Performance analysis of cloud applications," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 405–417. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/ardelean

[7] A. Maricq, D. Duplyakin, I. Jimenez, C. Maltzahn, R. Stutsman, and R. Ricci, "Taming performance variability," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA: USENIX Association, Oct. 2018, pp. 409–425. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/maricq

[8] L. Traini, V. Cortellessa, D. Di Pompeo, and M. Tucci, "Towards effective assessment of steady state performance in java software: are we there yet?" *Empirical Software Engineering*, vol. 28, no. 1, p. 13, 2022. [Online]. Available: https://doi.org/10.1007/s10664-022-10247-x

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[10] R. Chandra, S. Goyal, and R. Gupta, "Evaluation of deep learning models for multi-step ahead time series prediction," *IEEE Access*, vol. 9, pp. 83 105–83 123, 2021.

[11] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://aclanthology.org/D14-1179

[12] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 10 2001.

[13] Y. G. Cinar, H. Mirisaee, P. Goswami, E. Gaussier, A. Aït-Bachir, and V. Strijov, *Position-Based Content Attention for Time Series Forecasting with Sequence-to-Sequence RNNs*. Springer International Publishing, 2017, p. 533–544. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-70139-4_54

[14] A. Kusiak, A. Verma, and X. Wei, "A data-mining approach to predict influent quality," *Environmental Monitoring and Assessment*, vol. 185, no. 3, p. 2197–2210, Jun. 2012. [Online]. Available: http://dx.doi.org/10.1007/s10661-012-2701-2

[15] O. Barkan, J. Benchimol, I. Caspi, E. Cohen, A. Hammer, and N. Koenigstein, "Forecasting cpi inflation components with hierarchical recurrent neural networks," *International Journal of Forecasting*, vol. 39, no. 3, pp. 1145–1162, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207022000607

[16] K. Lin, Q. Lin, C. Zhou, and J. Yao, "Time series prediction based on linear regression and svr," in *Third International Conference on Natural Computation (ICNC 2007)*, vol. 1, 2007, pp. 688–691.

[17] W. Xu, H. Peng, X. Zeng, F. Zhou, X. Tian, and X. Peng, "A hybrid modelling method for time series forecasting based on a linear regression model and deep learning," *Applied Intelligence*, vol. 49, no. 8, p. 3002–3015, Feb. 2019. [Online]. Available: http://dx.doi.org/10.1007/s10489-019-01426-3

[18] S. Ben Taieb, A. Sorjamaa, and G. Bontempi, "Multiple-output modeling for multi-step-ahead time series forecasting," *Neurocomputing*, vol. 73, no. 10, pp. 1950–1957, 2010, subspace Learning / Selected papers from the European Symposium on Time Series Prediction. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231210001013

[19] G. White, A. Palade, and S. Clarke, "Forecasting qos attributes using lstm networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[20] D. Koutsandreas, E. Spiliotis, F. Petropoulos, and V. Assimakopoulos, "On the selection of forecasting accuracy measures," *Journal of the Operational Research Society*, vol. 73, no. 5, p. 937–954, Apr. 2021. [Online]. Available: http://dx.doi.org/10.1080/01605682.2021.1892464

[21] E. H. M. Pena, M. V. O. de Assis, and M. L. Proença, "Anomaly detection using forecasting methods arima and hwds," in *2013 32nd International Conference of the Chilean Computer Science Society (SCCC)*, 2013, pp. 63–66.

[22] J. Qiu, Q. Du, W. Wang, K. Yin, and L. Chen, "Short-term performance metrics forecasting for virtual machine to support anomaly detection using hybrid arima-wnn model," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, 2019, pp. 330–335.

[23] S. He, B. Yang, and Q. Qiao, "Overview of key performance indicator anomaly detection," in *2021 IEEE Region 10 Symposium (TENSYMP)*, 2021, pp. 1–6.

[24] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 387–395. [Online]. Available: https://doi.org/10.1145/3219819.3219845

[25] G. Zhao, S. Hassan, Y. Zou, D. Truong, and T. Corbin, "Predicting performance anomalies in software systems at run-time," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 3, apr 2021. [Online]. Available: https://doi.org/10.1145/3440757

[26] J. Zhou, Y. Qian, Q. Zou, P. Liu, and J. Xiang, "Deepsyslog: Deep anomaly detection on syslog using sentence embedding and metadata," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3051–3061, 2022.

[27] C. Lee, T. Yang, Z. Chen, Y. Su, Y. Yang, and M. R. Lyu, "Heterogeneous anomaly detection for software systems via semi-supervised cross-modal attention," *ArXiv*, vol. abs/2302.06914, 2023.

[28] N. Zhao, H. Wang, Z. Li, X. Peng, G. Wang, Z. Pan, Y. Wu, Z. Feng, X. Wen, W. Zhang, K. Sui, and D. Pei, "An empirical investigation of practical log anomaly detection for online service systems," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 1404–1415. [Online]. Available: https://doi.org/10.1145/3468264.3473933

[29] M. S. Islam, W. Pourmajidi, L. Zhang, J. Steinbacher, T. Erwin, and A. Miranskyy, "Anomaly detection in a large-scale cloud platform," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2021, pp. 150–159.

[30] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, J. Chen, X. He, R. Yao, J.-G. Lou, M. Chintalapati, F. Shen, and D. Zhang, "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 807–817. [Online]. Available: https://doi.org/10.1145/3338906.3338931

[31] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019,

p. 3009–3017. [Online]. Available: https://doi.org/10.1145/3292500.3330680

[32] Q. Lin, K. Hsieh, Y. Dang, H. Zhang, K. Sui, Y. Xu, J.-G. Lou, C. Li, Y. Wu, R. Yao, M. Chintalapati, and D. Zhang, "Predicting node failure in cloud service systems," ser. ESEC/FSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 480–490. [Online]. Available: https://doi.org/10.1145/3236024.3236060

[33] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.

[34] N. Zhao, J. Chen, Z. Yu, H. Wang, J. Li, B. Qiu, H. Xu, W. Zhang, K. Sui, and D. Pei, "Identifying bad software changes via multimodal anomaly detection for online service systems," ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 527–539. [Online]. Available: https://doi.org/10.1145/3468264.3468543

[35] G. Pang, K. M. Ting, and D. Albrecht, "Lesinn: Detecting anomalies by identifying least similar nearest neighbours," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, 2015, pp. 623–630.

[36] S. He, Q. Lin, J.-G. Lou, H. Zhang, M. R. Lyu, and D. Zhang, "Identifying impactful service system problems via log analysis," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 60–70. [Online]. Available: https://doi.org/10.1145/3236024.3236083

[37] Z. Ding and L. Xing, "Improved software defect prediction using pruned histogram-based isolation forest," *Reliability Engineering & System Safety*, vol. 204, p. 107170, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0951832020306712

[38] Z. Chen, J. Liu, Y. Su, H. Zhang, X. Ling, Y. Yang, and M. R. Lyu, "Adaptive performance anomaly detection for online service systems via pattern sketching," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 61–72. [Online]. Available: https://doi.org/10.1145/3510003.3510085

[39] E. Zhai, R. Piskac, R. Gu, X. Lao, and X. Wang, "An auditing language for preventing correlated failures in the cloud," *Proc. ACM Program. Lang.*, vol. 1, no. OOPSLA, oct 2017. [Online]. Available: https://doi.org/10.1145/3133921

[40] E. Zhai, A. Chen, R. Piskac, M. Balakrishnan, B. Tian, B. Song, and H. Zhang, "Check before you change: Preventing correlated failures in service updates," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 575–589. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/zhai

[41] S. Mehta, R. Bhagwan, R. Kumar, C. Bansal, C. Maddila, B. Ashok, S. Asthana, C. Bird, and A. Kumar, "Rex: Preventing bugs and misconfiguration in large services using correlated change analysis," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 435–448. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/mehta

[42] Y. Zhao, K. Damevski, and H. Chen, "A systematic survey of just-in-time software defect prediction," *ACM Comput. Surv.*, vol. 55, no. 10, feb 2023. [Online]. Available: https://doi.org/10.1145/3567550

[43] A. Gartziandia, "Microservice-based performance problem detection in cyber-physical system software updates," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2021, pp. 147–149.

[44] Z. Li, Q. Cheng, K. Hsieh, Y. Dang, P. Huang, P. Singh, X. Yang, Q. Lin, Y. Wu, S. Levy, and M. Chintalapati, "Gandalf: An intelligent, End-To-End analytics service for safe deployment in Large-Scale cloud infrastructure," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 389–402. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/li

[45] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, and Z. Zang, "Rapid and robust impact assessment of software changes in large internet-based services," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2716281.2836087

[46] J. Pool, E. Beyrami, V. Gopal, A. Aazami, J. Gupchup, J. Rowland, B. Li, P. Kanani, R. Cutler, and J. Gehrke, "Lumos: A library for diagnosing metric regressions in web-scale applications," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2562–2570. [Online]. Available: https://doi.org/10.1145/3394486.3403306

[47] L. Traini and V. Cortellessa, "Delag: Using multi-objective optimization to enhance the detection of latency degradation patterns in service-based systems," *IEEE Transactions on Software Engineering*, vol. 49, no. 6, pp. 3554–3580, 2023.

[48] C. Bansal, S. Renganathan, A. Asudani, O. Midy, and M. Janakiraman, "Decaf: Diagnosing and triaging performance issues in large-scale cloud services," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 201–210. [Online]. Available: https://doi.org/10.1145/3377813.3381353

[49] V. Cortellessa and L. Traini, "Detecting latency degradation patterns in service-based systems," in *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 161–172. [Online]. Available: https://doi.org/10.1145/3358960.3379126

[50] D. Krushevskaja and M. Sandler, "Understanding latency variations of black box services," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 703–714. [Online]. Available: https://doi.org/10.1145/2488388.2488450

[51] S. Han, Y. Dang, S. Ge, D. Zhang, and T. Xie, "Performance debugging in the large via mining millions of stack traces," in *2012 34th International Conference on Software Engineering (ICSE)*, 2012, pp. 145–155.

[52] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *International Conference on Autonomic Computing, 2004. Proceedings.*, 2004, pp. 36–43.