Devising a Trend-break-detection Algorithm of stored Key Performance Indicators for Telecom Equipment

Utformning av trendbrytningsalgoritm av lagrade nyckelindikatorer för telekomutrustning

ALI SYMERI
LAZAR HEDEDAL KLINCOV
Devising a Trend-break-detection Algorithm of stored Key Performance Indicators for Telecom Equipment

Utformning av trendbrytningsalgoritm av lagrade nyckelindikatorer för telekomutrustning

Ali Symeri
Lazar Hededal Klincov

Degree Project in Computer Engineering
First cycle, 15 ECTS
Supervisor at KTH: Martin Jacobsson
Examiner: Ibrahim Orhan
TRITA-STH 2017:32

KTH
The School of Technology and Health
141 52 Flemingsberg, Sweden
Abstract

A problem that is prevalent for testers at Ericsson is that performance test results are continuously generated but not analyzed. The time between occurrence of problems and information about the occurrence is long and variable. This is due to the manual analysis of log files that is time consuming and tedious. The requested solution is automation with an algorithm that analyzes the performance and notifies when problems occur. A binary classifier algorithm, based on statistical methods, was developed and evaluated as a solution to the stated problem. The algorithm was evaluated with simulated data and produced an accuracy of 97.54 %, to detect trend breaks. Furthermore, correlation analysis was carried out between performance and hardware to gain insights in how hardware configurations affect test runs.

Keywords

Trend analysis, trend-break-detection, anomaly-detection algorithm, key performance indicator, regression.


**Sammanfattning**


**Nyckelord**

Trendanalys, trendbrytningshittande, avvikelse-hittande algoritm, nyckelindikator, regression.
Acknowledgements

This paper is a result of a thesis project within the field of computer engineering, at the Royal Institute of Technology, for Ericsson.

We want to thank everyone in the CU team for their help in providing information regarding the current system at Ericsson and how the testing is performed. We want to thank Lars Karlström, Behdad Aminian and everyone else at Ericsson that helped us with this thesis. We also want to thank our mentor Martin Jacobsson for his guidance, support, and suggestions during this project.
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic benchmark</td>
<td>A benchmark that is designed with simulated data to test the limits of software or hardware.</td>
</tr>
<tr>
<td>Test suite</td>
<td>A collection of acceptance tests.</td>
</tr>
<tr>
<td>Test case</td>
<td>An acceptance test that ensures a feature to be implemented properly</td>
</tr>
<tr>
<td>Test manager</td>
<td>A person responsible for monitoring test results in addition to other responsibilities.</td>
</tr>
<tr>
<td>Test track</td>
<td>To and from upgrade package versions of the software in an upgrade test, also called upgrade path.</td>
</tr>
</tbody>
</table>
# Table of contents

1 Introduction .............................................................................................................. 1  
   1.1 Problem ............................................................................................................ 1  
   1.2 Goals ................................................................................................................ 2  
   1.3 Delimitations ..................................................................................................... 3  

2 Theory and background ......................................................................................... 5  
   2.1 Current system at Ericsson ................................................................................. 5  
      2.1.1 Test system .................................................................................................. 6  
      2.1.2 Node ............................................................................................................. 6  
      2.1.3 Node configuration ....................................................................................... 7  
      2.1.4 Shared pool ................................................................................................. 7  
      2.1.5 Test execution ............................................................................................ 7  
      2.1.6 Test tagging ............................................................................................... 8  
      2.1.7 Test outcome ............................................................................................. 8  
   2.2 Trend-break-identifying algorithm .................................................................... 9  
      2.2.1 Definition of anomaly .................................................................................. 9  
      2.2.2 Desired features of the algorithm ............................................................... 10  
      2.2.3 Types of anomalies ..................................................................................... 10  
      2.2.4 Binary classification ................................................................................... 10  
   2.3 Statistical analysis ............................................................................................ 11  
      2.3.1 EWMA ........................................................................................................ 11  
   2.4 Regression .......................................................................................................... 11  
      2.4.1 Simple linear regression .............................................................................. 12  
      2.4.2 Regression suitability evaluation .................................................................. 12  
   2.5 Evaluation of trend-break-identifying algorithm ................................................ 13  
      2.5.1 Contingency matrix ..................................................................................... 13  
      2.5.2 Measuring accuracy .................................................................................... 14  
      2.5.3 Algorithm scoring ...................................................................................... 14  
      2.5.4 Cross validation ......................................................................................... 14
2.6 Suitable tools for statistical calculations ......................................... 14
  2.6.1 Java ....................................................................................... 15
  2.6.2 R ......................................................................................... 15
  2.6.3 Interaction between Java and R ............................................ 16
2.7 Previous work about trending within Ericsson ......................... 16
  2.7.1 ADAT .................................................................................. 16
2.8 Related work ............................................................................... 17
  2.8.1 Relevance of related work .................................................. 19
3 Methodology .................................................................................. 21
  3.1 Literary study and pre-study ...................................................... 22
  3.2 Tooling .................................................................................... 22
  3.3 General flow ............................................................................ 24
    3.3.1 Collect data ......................................................................... 24
    3.3.2 Structure data ...................................................................... 26
    3.3.3 Analyze data ....................................................................... 26
    3.3.4 Present results ..................................................................... 26
  3.4 Choice of approach .................................................................... 27
    3.4.1 Choice of statistical regression .......................................... 27
    3.4.2 Residual analysis of data .................................................... 28
  3.5 Two algorithm proposals ......................................................... 29
    3.5.1 EWMA based algorithm .................................................... 29
    3.5.2 Anomaly based algorithm .................................................. 30
    3.5.3 Test scenarios ....................................................................... 31
    3.5.4 Algorithm tuning, EWMA based algorithm .......................... 31
    3.5.5 Algorithm tuning, anomaly based algorithm ...................... 34
  3.6 Algorithm benchmarking ........................................................... 34
  3.7 Correlation analysis .................................................................. 35
    3.7.1 Analyze data ......................................................................... 36
  3.8 Test candidates .......................................................................... 37
4 Results .......................................................................................... 39
4.1 Model used in the prototype ....................................................... 39
  4.1.1 Structure data ........................................................................... 39
  4.1.2 Data analysis ............................................................................ 40
  4.1.3 Present results from prototype ................................................ 40
4.2 Algorithm evaluation ................................................................... 42
4.3 Correlation .................................................................................... 43
  4.3.1 Cell count .................................................................................. 43
  4.3.2 UP version ............................................................................... 44
  4.3.3 Cell count and UP version together ......................................... 45
4.4 Algorithm execution and correlation analysis ................................ 46
  4.4.1 Node KIENB1052 from track 16B to 17A .................................. 46
  4.4.2 Node KIENB1327 from track 16B to 17.Q1 .............................. 48
5 Analysis and discussion ................................................................. 51
  5.1 Interpreting results of prototype ................................................ 52
     5.1.1 Accuracy and reliability of evaluation .................................... 53
     5.1.2 Tuning methodology ............................................................... 53
     5.1.3 Amount of data for correlation ............................................. 53
  5.2 Interpreting results of correlation .............................................. 54
     5.2.1 Correlation between cell count and time ............................... 54
     5.2.2 Significance of influence from cell count ................................ 54
     5.2.3 Notes on cell count correlation ............................................. 54
     5.2.4 Correlation methodology ....................................................... 55
     5.2.5 Correlation between UP and time .......................................... 55
     5.2.6 Comparison of nodes with different cell counts .................... 55
     5.2.7 Correlation in conjunction with algorithm ............................ 56
  5.3 Goals and expectations ............................................................... 56
  5.4 Alternate methodology ............................................................... 56
     5.4.1 Machine learning .................................................................. 57
     5.4.2 Root cause analysis ................................................................. 57
     5.4.3 Type of contending algorithm ............................................... 57
1 Introduction

The telecom industry is continuously evolving; hardware becomes faster and smaller, and software more efficient. Companies have the aim of improving their hardware and software in their business plan as a motivation to become the best in the industry.

The most publicly known products of telecom companies, besides the smartphone, are the Radio Base Stations (RBS), which is the hardware enabling communication between cellular devices. This type of hardware is deployed around the world and require adaptability for any environment. For this to be true, the quality of the product must be high. For the product to result in high quality, a range of different tests have to pressure it to its limits to prove that it is stable and worthwhile. However, the amount of test results within, for example Ericsson, are significantly large and diverse, and as such it becomes a challenge to find relevant data to identify problems that occur.

1.1 Problem

The testing of hardware is becoming a time consuming, error-prone, and tedious task. When hardware becomes more advanced, the risk of different issues or bugs appearing increases. Testing and analyzing the system irregularly is insufficient to achieve high quality. Analyzing different performance metrics strives to determine if the system has met the non-functional requirements.
One company that strives to create high quality products is Ericsson. They upgrade the existing software in the RBS and the upgrade process generates seven performance metrics, referred to as Key Performance Indicators (KPI). They are measurements of different resources and time consumption of tests during the upgrade process [1]. The analysis of these metrics is performed manually. Manual analysis can become expensive over time, because it requires a test manager to process the test results to find issues. A KPI that exceeds its threshold of maximum allowed value affects the test outcome. However, the measured KPIs lack any kind of automatic data analysis. A possible solution to this problem could be to track performance changes over time [2].

By continuously tracking performance changes over time, it would become possible to detect unintended performance degradation introduced by new features [2]. This would also become a leaner approach as the issues would be caught as soon as they arise which would minimize processing [3]. It would also reduce the risk of blocking a software release due to KPIs that exceed the specification. Therefore, to be able to make use of the KPIs, Ericsson sees the need for an algorithm that automatically analyzes these values and notifies of trend breaks. This would enable test managers to focus time and effort on solving the cause of KPIs that change drastically, rather than manually collecting and analyzing data for trend breaks, a task which would consume a significant amount of time.

1.2 Goals
To realize the proposed solution of automatically analyzing KPI values, goals were defined based on requirements from Ericsson. The two main goals that were established for this thesis are:

- Design and implement a prototype algorithm which identifies trend breaks of performance metrics in acceptance test results.
- Determine if correlations exist between the test hardware configuration and detected trend breaks.
There are also sub-goals for the two main goals that must be considered, such as extracting and structuring KPIs as well as evaluating different hardware configurations. The following steps have been identified:

- Study and evaluate the prerequisites.
- Investigate how the test results are stored and accessed.
- Analyze if hardware configuration affects KPIs.
- Find the most relevant hardware configuration values.
- Implement a prototype with the trend-break-detection algorithms.

### 1.3 Delimitations

The following delimitations are set:

- The prototype will only support one hardware generation, generation one (G1).
- Out of seven KPIs in total for G1, only two of them will be used:
  - Upgrade duration
  - Downtime duration
- Resource efficiency and system performance of the implementation will not be optimized or presented.
- Database connection details of the system are not regarded in this paper.
- Visualization in our prototype, of test results, is not prioritized; rudimentary techniques will be employed.
- Software design patterns will not be used in the prototype.
2 Theory and background

This chapter presents the theoretical framework for the proposed solution and the background of the problem. Section 2.1 describes the testing pipeline, how tests are executed, scheduled, and how results are stored. Section 2.2 presents theory relevant to trends including the type of algorithm that is devised and a description of features of the desired algorithm. Section 2.3 presents statistical theory about the data preparation for the algorithm. Section 2.4 presents a regression analysis and evaluation of the correctness of a chosen regression method. Section 2.5 presents theory regarding methods for accuracy evaluation of an algorithm. Section 2.6 presents a selection of tools that are available for statistical calculations and implementation of the algorithm. This is followed by an analysis of previous work about trend analysis and trend detection within Ericsson in section 2.7. The chapter concludes with section 2.8, containing a presentation and evaluation of related work.

2.1 Current system at Ericsson

Ericsson performs quality assurance (QA) on their products and the responsibility of testing is divided into different teams, each with a domain of quality. The teams are internally referred to as QA teams. The domain of quality distinguishes the teams, i.e. QA team “Configure and Upgrade” is the team responsible for ensuring quality of software upgrades and configurations for the products. The QA team “Configure and Upgrade” is henceforth referenced to as the CU team.

The testing environment of the QA teams at Ericsson involves test suites with different forms of tests. This depends on the focus of the team but can be a combination of functional-, performance-, installation-, acceptance, and regression testing. The hardware of Ericsson RBS products is divided into two generations, G1 and G2. Each generation has its unique hardware configuration, where the number of radios or nodes in the RBS varies. The node itself contains different number of cells.
Currently at Ericsson, there are also tests that check different approaches regarding upgrading the software on the radio hardware existing in the RBS. There is limited temporal overview of the performance metrics from the test results because of inconsistent data and limited analysis. Due to this, test managers in every team, who monitor the test verdicts and results, have to extract the performance metrics manually.

The CU team executes automated test suites nightly. Each suite consists of multiple test cases. The nightly tests generate test results that are stored in a relational database for archiving and they contain a wide range of information such as, test verdict, performance metrics, timestamps, software version and more.

After the tests are executed, if there is an issue or bug present, the tester has to write a trouble report (TR). The TR is written in detail so that a fix can be proposed in a short amount of time. In this case if there is a trend break, the TR will contain any information regarding when and where the trend break occurred.

2.1.1 Test system

The current test system at Ericsson consist of several parts. Each part has its own contribution towards the final product, which is the test execution environment. The parts are node, node configuration, shared pool, test tagging.

2.1.2 Node

A node is a Signaling Transfer Point (STP). It is a dedicated hardware in an RBS, that is responsible of managing the communication in a mobile network.
2.1.3 Node configuration
The upgrade testing is executed nightly on RBS hardware in the laboratory of Ericsson. To be able to maximize the usage of the nodes inside the RBS, a configuration file for every node is created to keep track of what the hardware specifications look like. The configuration file holds information regarding the different nodes since there are multiple nodes in every RBS. These configurations vary in what the actual node is capable of, i.e. cell count, upgrade package (UP) version and network manager.

The cell count is the number of cells in a node and UP version is the software version installed on the node. Cell count influences the geographical coverage of the cell service. More cells provide higher coverage.

2.1.4 Shared pool
The availability of the different nodes that are present are shared between many QA teams. When the nightly tests are about to run, a node on which it will be executed is first selected and delegated to the test. The node is selected by a scheduler called Test Generic Framework (TGF) which is developed internally at Ericsson. All available nodes are placed in a shared pool where the accessibility is open for every QA team to use, this means that the scheduler will go to this pool, select a node, and delegate it to the test that is requesting a node. In the shared pool, all node configurations are created in a way so that a test from any QA team could run on any available node, which is the purpose of the shared pool. This means that the configuration is generic and every different test suite can run on any node in the pool.

2.1.5 Test execution
The executable test cases are mostly single threaded Java tests. Every test case that is executed on the node will run sequentially which means that if there are multiple test cases to run, it will wait for the first test case to finish to be able to advance to the next. However, the test suites run in parallel because multiple nodes are running tests simultaneously but the nodes have no correlation to each other.
2.1.6 Test tagging

The test suites vary for every QA depending on what the focus is for that team. Since the CU team is performing software upgrade, they decided to split it up to yearly and quarterly software test tracks. The different tracks, also known as software versions, are tagged with an id depending on the current quarter of the year. To simplify, the id is incorporated in the test suite as 16A or 16B. It is also a key importance with the track change since it is always the latest track that is delivered to the customer.

Upgrading the software from one track to another can affect the KPIs. Since the tracks are different, it means that the preceding track should always be less feature packed and require less time for KPIs. An upgrade to a newer track should mean that the KPIs will consume more time since there are more features. However, in the available data, it is impossible to know if there are new features or better optimization of the code, which could increase or decrease the KPIs time consumption.

Worth noting is that the KPI upgrade duration has a threshold of 10 minutes and 20 seconds, and downtime duration has a threshold of 90 seconds. Any measured time above the threshold leads to a failed test case.

2.1.7 Test outcome

When test suites finish execution and test results are generated, the suite finalizes the execution by storing the results in a relational database. The relational database used at Ericsson is Test Automation Result Database (TAR DB), which is built on top of MySQL. The test results stored in TAR DB are also mirrored to a database warehouse, TTS Dynamic Report (TDR), which stores data from many source systems with dates of creation.

Test suites have multiple test cases which are children to the test suite. The storing process for this setup is perfect for a relational database such as MySQL. The parent, which is the test suite, is stored in one table and the children which are the test cases are stored in another table with reference to their parent.
The test cases stored in TAR DB consist of the test outcome and KPI values. The relevant KPIs for QA configure and upgrade are “Upgrade duration” and “Downtime duration”. The first is the duration of the installation of a software upgrade package onto the hardware. The second KPI is the time it takes for the hardware to make a restart and initialize to a usable state. During downtime, the RBS will provide limited cellular network service. The data extraction will therefore be focused on those two KPIs and targeted for analysis.

2.2 Trend-break-identifying algorithm

Required features of a trend-break-identifying algorithm are: accuracy, flexibility, and adaptability.

2.2.1 Definition of anomaly

In a broader sense the algorithm is needed to detect anomalies. An anomaly is a deviation from the normal occurrence [4]. The definition of the magnitude of the deviation and the amount of deviations can vary. In the current context, an anomaly is defined as a trend break. A trend break is when additional KPI measurements change and are at a higher level than previous KPI measurements that have been on a similar level. The change involves several measurements to increase in magnitude yielding a new trend. The establishment of a new trend requires at least three measurements to be on a new level after the first deviation was detected. The smallest constituent part of a trend break is a trend deviation. A trend deviation consists of a noticeable change in the magnitude of a KPI measurement, noticeable is about 5-10%. A trend deviation on its own is an outlier, several contiguous trend deviations constitute a trend break.

A trend break is defined with regards to the testing environment as a change in magnitude of one KPI measurement. The reason for not regarding one deviating instance as an outlier is the way the tests are run. Because of tests running nightly, there is much time to analyze and potentially fix issues that arise during the day. This means that the proportion between the types of detections are skewed towards false detections. This is because an undetected error is more damaging than the contrasting situation. This makes it difficult to filter outliers. A more precise way to denote this type of anomaly is trend-deviation or change detection.
2.2.2 Desired features of the algorithm

An algorithm which identifies trend breaks must be adaptable to the gradual changes in the data. Newer measurements should have a greater impact on the detection outcome in contrast to older measurements. Furthermore, the algorithm should be generic to the type of KPI.

2.2.3 Types of anomalies

As shown by Peiris et al. [2], two types of software performance anomalies are relevant to identify. The first type is based on performance requirements and the second type is based on deviations from normal performance. The first type is, in this case, the maximum allowed KPI specification. The second type requires establishing a baseline that represents normal performance and defining a threshold to allow for minor variations [5]. Incoming data is compared to the established baseline to identify anomalies. One common cause of unexpected performance degradation, are new software releases [1,6]. In this case performance degradation is increased upgrade and restart time, which are exhibited through increased KPI measurements.

2.2.4 Binary classification

Schölkopf et al. [7] present binary classification at its most basic form as a method of assigning data to the corresponding set. More concretely, given data that belongs to one of two classes, how are new data points assigned to the correct class? The assignment is based on some property of the data. Furthermore, the same authors formalize the problem as follows:

\[
(x_1, y_2), \ldots, (x_m, y_m) \in X \times \{\pm 1\}
\]  

(1)

In (1) \(X\) is a non-empty set of arbitrary elements and the pairs of \((x_m, y_m)\) is just a representation of the data. In the representation \(x_i\) is the pattern (or input) and \(y_i\) is the label (or output). The labeling with +1 or -1 is the actual binary classification step.

This shows that the trend-break-identifying algorithm is a binary classifier. The classification occurs when a new KPI measurement is determined to be within the general trend or not.
2.3 Statistical analysis

An integral part of the trend-break-identifying algorithm is statistical analysis. It supplies methods for processing and manipulation of fluctuating data.

2.3.1 EWMA

Exponential Weighted Moving Average (EWMA) is a technique for describing historical data with emphasis on the most recent data. Roberts [8] shows in his research that EWMA is a version of moving averages with weights assigned to each entry. The weights are assigned in decreasing order from most recent to oldest observations. EWMA is defined recursively by the following formula:

\[
S_t = \alpha \cdot Y_t + (1 - \alpha) \cdot S_{t-1}, \text{ for } t > 1
\]

(2)

In (2) \(S_t\) denotes the calculated EWMA. \(Y_t\) is the new value to calculate and \(t\) is denoting the position in the sequence. \(S_{t-1}\) is the previously calculated EWMA. This formula requires an initial value, setting it to the first value \(S_1 = Y_1\) is the most basic way of achieving this. Furthermore, \(\alpha\) is the exponential smoothing factor, the range bounds it \(0 < \alpha \leq 1\). The exponential smoothing factor is sometimes referred to as ratio. It determines how influential the current term is. Thus, the closer the factor \(\alpha\) is to 1, the faster the discard-rate of older values as the second term approaches 0. There are several conventions for determining the value of \(\alpha\), Welles Wilder’s ratio of \(\frac{1}{n}\) is one [9].

2.4 Regression

Traditional regression as shown by Davino et al. [10] is a statistical method to analyze datasets. It focuses on a variable \(Y\) for a value of a set of variables \(X\) which is the regression mean function as follows:

\[
E(Y|X)
\]

(3)
The focus lies only on a set (3) of the $Y$ distribution while estimating the conditional expectation of the dependent variables $X$ in the set. This means that the average of the set is estimated. The typical use of regression is to first summarize observed data and then perform prediction and forecasting on the dataset using analytical methods as well as finding correlations between variables \([11]\).

**2.4.1 Simple linear regression**

Simple linear regression is a method to predict scores in datasets. When there are two variables $X$ and $Y$, then a score from variable $X$ is predictable from the scores of variable $Y$. Stated by Weisberg \([11]\), the predicted value $X$ is a criterion variable and the variable that it is based on, $Y$, is called the predictor variable. If there is only one predictor value, then the method called simple regression is being referred to. Since the simple regression method is a linear method considering at most one or two functions, a graph plotted with the predicted $Y$ values as a function of $X$ will be linear, a straight line. The best case for linear regression is to find the best fitted straight line through the plotted values, called the regression line \([11]\).

There also exist multiple other regression methods such as, nonlinear regression and quantile regression. They are complex and support nearly an infinite number of possible functions or they focus on the median regression \([12,13]\). They are also model fitting, so the line curves to fit the model better, which is determined to not be suitable for the dataset in this project \([10,12]\).

**2.4.2 Regression suitability evaluation**

Residual analysis is a method for evaluation of whether a linear- or nonlinear regression describes the data more accurately \([14]\). Residuals are the difference between the observed value, and the value predicted by a regression model \([15]\). The formal definition is as follows:

$$ e = y - \hat{y} $$

(4)
Residuals $e$ are calculated in (4) where $y$ is the dependent variable and $\hat{y}$ is the predicted value by the regression. By creating a “residual vs. fit” plot, the fit of the chosen regression can be evaluated. In a “residual vs. fit” the residuals are plotted on the y-axis and the predicted values are plotted on the x-axis. A linear regression is the best fit for the data if the plot is random, otherwise the data is more accurately described with a nonlinear regression model [14].

2.5 Evaluation of trend-break-identifying algorithm

During algorithm development, the algorithm must be evaluated to ensure maximum accuracy. The accuracy of an algorithm can be determined through benchmarking. The level of acceptable sufficiency should be defined in the evaluation process. One approach to determine accuracy is to measure the number of false positives in relation to true negatives. This creates an optimization problem where an interval of the lowest number of false positives and the highest number of true negatives is sought.

An important aspect of evaluating the accuracy is to define a realistic dataset. It should vary in a way that generates the edge cases for which the algorithm is likely to generate wrong results. Furthermore, the behavior of data must be known for it to act as a reference when determining if the results of the algorithm were correct. Another important aspect is determining the lowest level of sufficient accuracy. 100% accuracy is always the goal; the question is; what level of inaccuracy is acceptable? Furthermore, what type of inaccuracy is more manageable, a higher number of false positives or false negatives.

2.5.1 Contingency matrix

The outcome from an accuracy measurement can be represented in a $2 \times 2$ contingency matrix [16]. The values for the matrix are generated by comparing the results of the binary classification algorithm to a reference classification. There are four possible combinations of outcomes: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). True positive is when the prediction and actual outcome are both “yes”. False positive is when the prediction is “yes” but the actual outcome is “no”. True negative is when both the prediction and outcome are “no”. False negative is when the prediction is “no” but the outcome is “yes”.
2.5.2 Measuring accuracy

When evaluating the accuracy of an algorithm a rigorous definition of accuracy is needed. For classification problems accuracy is defined as in (5). This describes overall accuracy while precision and recall defines parts of the accuracy. Precision is defined in (6) and recall in (7) [17].

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)
\]

\[
\text{Precision} = \frac{TP}{TP + FP} \quad (6)
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \quad (7)
\]

These metrics are derived in conjunction with a contingency matrix.

2.5.3 Algorithm scoring

One approach is to benchmark the algorithm and assign scores based on the result of the benchmark. The scoring can be based on the contingency matrix principle presented in section 2.5.1. It is appropriate to introduce a second algorithm as a reference for comparison, the algorithm with the highest scores is then adopted.

2.5.4 Cross validation

When a statistical model is devised it is based on some sample data. However, it needs to be validated to ensure that new data is recognized in the desired way [18, 19, 20]. Cross validation is one approach to ensure that the model is not “overfitted” to the sample data. Overfitting occurs when the model is skewed towards the sample data. This results in a model that handles new data insufficiently. Overfitting is a particularly significant issue for data mining and machine learning algorithms [19].

2.6 Suitable tools for statistical calculations

To minimize lead time, suitable tools were evaluated. The tooling in this case are primarily targeted for the algorithm to acquire implementations of statistical methods.
2.6.1 Java

Java is a general-purpose object-oriented programming language developed by Sun Microsystems and maintained by Oracle corporation. Java runs on the Java Virtual Machine (JVM), which makes the programs portable between platforms [21]. The code is compiled to the intermediate bytecode which is interpreted by the JVM. Java has a default library that is exposed through the Java API [21]. Additional features can be introduced through external dependencies [22]. These are packaged as Java Archive (JAR) files; external dependencies can be managed by a build automation tool such as Maven [23].

Java is a versatile language supporting both standalone desktop applications and web applications. It is distributed in different editions for different purposes. Java Standard Edition (Java SE) is geared toward standalone desktop application development [21]. Java Enterprise Edition (Java EE) is a superset of Java SE, specialized for web application development [24].

2.6.2 R

R is a programming language for statistical analysis and work environment for interactively performing statistical calculations. The common statistical fields include: descriptive statistics, analysis of variance and linear regression methods [25]. Nonlinear least squares calculations are available as well [26]. Additional features are available through packages; these are distributed in an “as is” condition via the Comprehensive R Archive Network (CRAN) [25]. Most advanced features are available in these packages.

There also exists a similar language for statistical analysis, called MATLAB. It is a system for technical computations designed for scientific and engineering calculations [27]. It covers many topics including linear regression [28], Markov process and numerical methods for integration and first-order differential equations [27]. Data analysis features are also provided [28]. An Application Programming Interface (API) is provided for the Java programming language. It enables applications to interact with MATLAB to perform calculations [29].
2.6.3 Interaction between Java and R

As Java and R are two different languages some form of interaction needs to be established. There are different ways of achieving this. One way is to use Java R Interface (JRI) which enables the Java program to instantiate R in a thread within the program. JRI is distributed in the rJava package [30]. Another form of interaction is provided by Rserve, a socket server which is reachable over local sockets or the Transmission Control Protocol (TCP) [30]. The server can be started from within the R Graphical User Interface (GUI) or terminal [31]. With both approaches R exposes an API to Java, Rserve needs a client-side implementation, this makes it support languages other than Java [30].

2.7 Previous work about trending within Ericsson

The internal situation at Ericsson is limited when it comes to analyzing data from test results. Three tools are used internally to analyze data, but none of them are focused specifically on the test results or the KPIs in test result data. They are more general and analyze the overall situation of the testing environment and the network. The existing tools are, Automatic Data Analysis Tool (ADAT), EIS insight, and Insights Tool Kit (ITK).

The first tool, ADAT, is the most and only interesting out of the three because it mentions KPIs and uses a statistical analysis approach.

2.7.1 ADAT

This tool is trying to automate human activity as much as possible in network analysis. ADAT prepares information and finds patterns and anomalies with different data analysis techniques. Most of the reports from ADAT are written in the statistical computing language R.

To use ADAT, an administrator has to define the test objective and set reference and measurement periods. Once the test is started the reference will be compared to the measurement period to analyze and detect significant changes in behavior. The analysis is done partly at KPI level.

Furthermore, ADAT is also utilizing different approaches in analyzing data. It uses i.e. sliding window analysis and daily analysis. However, this thesis will not extend ADAT, so the methods will not be explained in detail.
Sliding window is used to continuously monitor the network KPIs before data of a whole day are available. The present measurement will be compared with the reference period and the information from the reference period is summarized as a confidence band which present values should not exceed. ADAT has chosen 99% confidence level to reduce the number of false positives. It expects that most of the measurements will fall inside the confidence band. To indicate a change in a KPI, then a considerable amount of measurements has to fall outside the confidence band. The primary analytical methods that sliding window analysis in ADAT uses is quantile regression and state space model.

The daily analysis is executed on daily levels and displays KPI changes across days. The method for analyzing data that it uses is the Student’s t-test that is based on the statistical hypothesis test.

2.8 Related work

In a study by Verbesselt et al. [32] the Breaks For Additive Seasonal and Trend (BFAST) method was used to identify trend breaks. The method is available as a package for the statistical programming language R. The goal was to identify trend breaks from seasonal changes over a nine-year period. This was achieved by analyzing vegetation with satellite imagery and identify seasons. It was shown that BFAST detected trend breaks based on the historical data when the seasonal changes exhibited temporal deviations.

In a study conducted by Zhou et al. [33] from 2016 anomalous natural changes caused by i.e. flooding or forest fire, were detected from satellite images. The method used was EWMA, it was compared to another method, proposed by the same authors in another paper, which was based on z-scores. Both methods were customized specifically for Season-Trend model Residuals (STR) data, which is data that consists of the decomposition of a seasonal times series by using a Season-Trend model. The resulting components from the model were trend, season, and residuals [34].

It was shown that EMWA provided a higher level of granularity for change-detection. Both major and minor changes were identified in contrast to the z-scores method which only detected major changes. The accuracy of the trend-change-detection was also significantly higher. The performance of the two methods was compared using a receiver operating characteristic (ROC) curve.
Gow et al. [35] created a platform agnostic method to identify performance anomalies in the form of application slowdown events. This was achieved by using regression curve fitting with Cumulative Distribution Function (CDF) from M/M/1 queue theory. The data was regarded as a Poisson distribution. The anomaly detection was based on comparison of application signatures of performance. The pre-computed signature was compared to the most recent one, computed within the current time window. The signatures were system wide. The actual detection used nonlinear regression least squares analysis. The need for calibration when conditions change was reduced. The method was only evaluated for corporate business web application.

Zhang et al. [6] evaluated a non-negative least-squares regression-based approach for identifying anomalous Central Processing Unit (CPU) usage of multi-tier applications. The identification was focused on CPU usage that arose from software updates and patches. It was concluded that the accuracy of the results from the regression analysis heavily rely on the quality of the data that was used. The data that was used in the anomaly detection step was retrieved from logs. The anomaly detection step was defined as comparing actual performance to the predicted performance of the system for a limited time. The change in performance is compared to a threshold, only threshold violations triggered alarms.

Xu [36] et al. analyzed correlations of problems that occurred in large scale systems (also referred to as root cause analysis). It was carried out by analyzing console logs with data mining and machine learning techniques. This approach enabled full automation and was divided into four steps: code analysis, log structuring, anomaly detection in console logs and visualizations of the findings. The source code that generated the log was defined as its schema.
Through code analysis all possible log messages could be acquired and represented with a data structure. The log messages were analyzed for variables and related messages were grouped together. State ratio and message count were two features that were tracked among the log messages. The first represented system behavior over time and the second detected problems of operations. The unsupervised learning algorithm Principal Component Analysis (PCA) was used to identify anomalies by labeling features as normal or abnormal. The results of the analysis were visualized in a decision tree. An accuracy of 99.8 % was achieved for two systems that were evaluated.

2.8.1 Relevance of related work

The common denominator of the evaluated related work is the focus on large scale systems or seasonal time series data. The set goal is geared towards a domain which have different requirements than the server environment, that appears in some of the related work. The amount of generated data is consequently considerably lower. This should be considered to achieve similar accuracy as in scenarios with huge amounts of data. This requires additional methods to be evaluated and adapted.
3 Methodology

This chapter presents the chosen methods and tools to fulfill the goals defined in section 1.2. The choices were based on an initial pre-study consisting of a literature study as well as researching and investigating the current system at Ericsson. Implementing and testing different algorithms were also included in the methodology to evaluate the chosen solution. Section 3.1 presents the methods that were used to conduct the pre-study. Section 3.2 presents the tooling that was used to implement the prototype. Section 3.3 presents the model that was devised as a basis for the prototype. The approach taken to meet the goals is presented in section 3.4. Section 3.5 presents two algorithms that will be evaluated. Section 3.6 presents the methods used to benchmark the algorithms. Section 3.7 presents the methods that were used to perform correlation analysis. Finally, the way data was filtered for correlation analysis is presented in section 3.8.

Berndtsson et al. [37] writes that many projects in computer science consist of introducing or developing new solutions to different problems. Even if the current solution is working, a new and possibly better solution could be created with more advantages than the previous one. The implementation-methodology is a necessity for solving a problem of this nature [37]. Since this thesis work includes researching different solutions for anomaly detection, the proposed implementation-methodology was chosen to develop two prototype algorithms.

To analyze and compare the different algorithms, the experimental approach has been chosen. The focus of the experimental approach lies in changing the experimental conditions to see if the changes affect different variables [37]. By running experimental tests with the different algorithms, in combination with changing the conditions and variables, a valid comparison could be performed.
3.1 Literary study and pre-study

A literary study was conducted during the first phase of the pre-study for several reasons. The primary reason was to identify suitable approaches when designing algorithms which identifies trend breaks. This was done through analyzing related work. Related work was also a component in evaluating in which way broad topics such as statistical analysis, machine learning and data mining could be applied to the task of devising a trend-break-identifying algorithm. This was the foundation for determining the most suitable approach for the implementation.

The other phase of the pre-study was performed by analyzing the current situation at Ericsson. This was done through researching the execution flow of tests and the different internal tools. An analysis was carried out to determine feasibility and relevance of further development of internal tools. Thus, ADAT was found. However, ADAT could not be used or contributed to because it was customized for one area, the network. On the other hand, a few of the implemented methodologies that ADAT used could be considered for this thesis, a few of those were the statistical language R and the sliding window approach.

3.2 Tooling

The basic philosophy with the tooling was to rely on using reviewed and widely used implementations where possible. This approach reduced the code base, test base, lead time, and the risk for erroneous implementations.

Java was chosen as the programming language because of its aforementioned level of flexibility, with regards to application-types, extensibility with external libraries, and interoperability between platforms. The authors prior experience with the language was also a contributing factor.
Based on the philosophy of relying on well tested implementations, R was chosen as the source of statistical implementations. Another benefit was flexibility, as different experimental features could be tried manually out in the R GUI. If the feature was relevant it could easily be implemented into the prototype using the same statements that were experimented with manually. A drawback to this R based approach is the more complex and more platform-dependent deployment. Another drawback is the mixed code base containing both Java and R code. But no requirements were posed on code maintainability or deployment, and the tradeoff was considered reasonable for the set goals. A production grade implementation could be done in some other language with or without the R dependency.

R packages were acquired from CRAN in addition to the default statistical functions that are bundled with R. The distribution of functionality over bundled and additional packages that were used are presented in table 3.1.

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (default installation)</td>
<td>Simple linear regression</td>
</tr>
<tr>
<td>Technical Trading Rules (TTR)</td>
<td>EWMA</td>
</tr>
<tr>
<td>Rserve</td>
<td>Server reachable over a TCP connection</td>
</tr>
</tbody>
</table>

Java required additional dependencies to access the Rserve instance over a TCP connection. The dependencies were acquired over the MVNRepository with “org.rosuda.REngine” as the groupId and “Rserve” as the artifactId. The version of the dependency used was 1.8.1. The complete set of Maven dependencies that were used are presented in listing A.1 in appendix A.

The alternative, MATLAB, could also have been used since it provided similar features as R. However, MATLAB had insufficient documentation for it to be used properly which would make the implementation more time consuming. R was also used in ADAT at Ericsson, which could possibly allow integration to the tool.
3.3 General flow

For every project of the implementation and experimental nature, where analysis must be performed on a dataset, a general flow of the system is commonly defined as a model. The experimental method often pairs up with an implementation of a model which can be simulated and tested [37].

For this thesis, it was decided that an obvious general pipeline of the system, as seen in figure 3.1, should be the model to be implemented. The pipeline consists of four phases, collecting data, structuring data, analyzing data and the optional phase of visualizing data.

![Figure 3.1: The general pipeline of the trend-break-identifying system.](image)

3.3.1 Collect data

The data collection step was concerned with querying the TDR database warehouse to extract the necessary data. For the extraction, MySQL queries were sent to the database to get the necessary data. For the CU team, the queries that were sent, focused on finding data regarding software upgrade, night runs and KPI. The information was not a single value that corresponded to the column name and datatype. It could be a combination of strings and values, with a structure like a log. Data from such entries was parsed using regular expressions in Java to isolate the required values.

The MySQL queries that were sent to the database are presented in listings 3.1 and 3.2 below. The first SQL query, listing 3.1, collected the parent data which were the test suites, from the database. To make sure the collected data was correct, the query specified that the parent must have children which were test cases and they must have at least one of the two KPIs. It also targeted the CU team, their night runs and the last 1500 entries. The second SQL query, listing 3.2, used the results from the first query, hence the “parent” parameter in this query. This query collected all the child entries for every parent from the database.
SELECT *
FROM tar_testsuite_runs
WHERE TgfActivityType='QA Configure and Upgrade LTE'
  AND TgfActivitySpec='nightRun'
  AND SeqId IN
    (SELECT TestsuiteRunsSeqId
     FROM tar_ctc_runs
     WHERE comment regexp 'duration|downtime')
ORDER BY SeqId DESC limit 1500;

Listing 3.1: SQL query that polls the test suite table for test suites that has children with KPIs, for the CU team.

SELECT *
FROM tar_ctc_runs
WHERE TestsuiteRunsSeqId=" + parent + ";

Listing 3.2: SQL query that polls the test case table for the KPIs for the specified parent.

The dataset that was collected consisted of multiple variables, also known as columns, presented in appendix B in table B.2 for the test suites and B.1 for the test cases. However, for the algorithm to function and correlation to be detected, the only required variables were in the test case table and they were “CtcProperties” and “Comment”. The CtcProperties field contained information regarding the hardware configuration, i.e. cell count, node name and UP version. The Comment field contained information regarding the measured and maximum allowed KPI.

The limit of 1500 entries was determined by inspection of the number of results for the queries which was about 1300. The limit had no effect on the query if there were less than 1500 entries. However, if entries accumulated to being more than 1500, it would limit the number of received data which would decrease the query time. This amount of relevant data was reasonably low to process in the Java application. The queries were considerably slow in how long they took to execute, at about 90 seconds; thus, the data processing was moved to the application level. The high latency of queries was due to table structure in the database and the total number of entries. The entries amounted to a total of 82 064 155 rows for the test cases and 11 278 792 rows for the test suites at the time of inspection.
3.3.2 Structure data

The purpose of a data structuring step was to enable the algorithm to be generic for the different types of KPIs. One approach to achieve this was to define a convention of structuring and tagging the data. This makes different types of data self-descriptive and unified. If the need for analysis of a new kind of KPI would arise, the changes would be carried out in this and the previous step, which would leave the algorithm untouched.

Another important purpose of this step was that the data representation was translated from a relational model to an object-oriented model. Furthermore, the data was structured for conformity with the requirements posed by the analysis step. The structuring was concerned with both the representation and the contents of the data. The representation was regarding the structuring and the contents were regarding the grouping of data which was relevant to analyze. The relevant data was determined through filtering which was based on hardware configuration and test track. Test tracks were relevant to monitor for trend breaks separately from each other, on a hardware configuration basis. This was an important requirement for algorithm accuracy as variations between hardware configurations were high.

3.3.3 Analyze data

The actual trend-break-detection algorithm implementation is at the analysis step. The implementation is presented in detail in sub-sections 3.5.1 and 3.5.2.

3.3.4 Present results

In this step, the results from the analysis are presented. The presentation can be composed as simple output, a table or in a more complex form, such as visualized in a graph. The resulting data from the analysis step is expected on a generic format which enables different ways of presentation.

This kind of pipeline as processing model encapsulates the concerns within each step, this isolation of steps provides modularity of the system. If some requirement changes, changes only need to be carried out within a limited number of steps. If the need for additional steps arise, integration of such steps would in most cases not affect adjacent steps. Modularity is enforced by coding to interfaces, a programming principle which decouples code [38].
3.4 Choice of approach

The focus on the choice of approach lies on the three major and relevant approaches in the trending area. The three approaches, to solve the problem stated in section 1.1, are:

- Statistical analysis
- Machine learning
- Data mining

The chosen approach for algorithm design was statistical analysis for this thesis. It was chosen because it could produce the most accurate results for almost every dataset, with high confidence and more than enough background in the statistical analysis area to make it reliable. Also, since the dataset that this project worked with was considered to have two parameters and not more, upgrade duration over time and downtime duration over time, the statistical approach was better suited than the alternatives.

The alternatives, machine learning and data mining, were also considered, however, the two approaches were not as relevant to the core of this thesis as statistical analysis was. Because when it came to those alternatives, a more complex and tagged dataset with multiple parameters was required which was not present in the targeted dataset [39]. The data mining approach was also often combined with machine learning to process big data but the data that was available from the executable tests was not big data [39]. The two alternatives would also be more time consuming to implement than statistical analysis.

3.4.1 Choice of statistical regression

Since regression is a byproduct of the statistical approach, different statistical analytical methods are involved to complement the analysis. A useful statistical method to use in projects where trending is the focus, was regression. Therefore, out of all the different regression methods that are available, three were mentioned for this thesis:

- Simple linear regression
- Nonlinear regression
- Quantile regression
The chosen regression method, out of the three, was Simple linear regression. A test manager would only look at the visual graph and not the actual presented numbers, hence a straight linear line.

The alternatives nonlinear- and quantile regression provided a more complex and advanced visual graph than what was required for trend prediction in this thesis.

Simple linear regression was used for the feature of providing a visual aid that summarized the general trend of a KPI over time. Implementations were identified in both the Apache Java library “org.apache.commons.math3” and the default R distribution. The implementation bundled with R was chosen for consistency with the approach of relying on R and R packages for implementations of statistical methods.

3.4.2 Residual analysis of data

To further argue for the selection of simple linear regression, a residual analysis was performed to prove its suitability. Described in section 2.4.2, the “residual vs. fit” plotted data should not have a pattern and be random. This can be seen in figure 3.2 where an existing node was tested on, as a sample. It showed that there was no pattern in the plotted residuals and they were randomly distributed. This meant that, linear regression was most suited for the data that was available from the nodes.
3.5 Two algorithm proposals

To find the trend-break-detection algorithm with the highest accuracy two algorithms were devised. One algorithm was focused on trend-break-detection and the other was customized towards anomaly detection. The use of two different types of algorithms was based on the hypothesis that it might provide additional insights to the problem that were not obvious at the initial stage. Each algorithm was tuned for maximal performance. The two algorithms were benchmarked and compared as a part of the accuracy evaluation process. A requirement for the algorithms was to only identify positive trend breaks. That is, measurements that are higher in magnitude than the current trend.

3.5.1 EWMA based algorithm

The EWMA based algorithm was defined to use statistical methods for maximal accuracy and rate of adaptability. The adaptability rate was determined to concern how easily the algorithm accepts changes. The definition was based on the research from the pre-study. EWMA was set to be the core of the algorithm as it provided filtering of outliers over time and consideration of values within a range. Consequently, EWMA held responsibility for achieving most of the desired features.
The trend-break-detection used (8) to calculate and compare the relative difference between measurements. \( Y_t \) is the current measurement and \( S_t \) is the calculated EWMA value for that same measurement. A threshold, denoted as \( \delta_{max} \), was incorporated in the comparison to discard fluctuations in the measurements.

\[
\delta_{max} < \frac{Y_t - S_t}{Y_t}
\]  

The exponential smoothing ratio \( \alpha \) and the threshold were two parameters that had to be determined for more optimal trend-break-detection. This was carried out through algorithm tuning, described in section 3.5.4.

### 3.5.2 Anomaly based algorithm

The contender for the EWMA based algorithm was defined in the most rudimentary way as in (9). It focused on detecting anomalies by comparing the change between measurements. The comparison involved the relative difference between the most recent value \( Y_t \) and the previous value \( Y_{t-1} \), to be above the threshold denoted as \( \delta_{max} \).

\[
\delta_{max} < \frac{Y_t - Y_{t-1}}{Y_t}
\]  

The threshold was determined experimentally through tuning, described in section 3.5.5. Only one factor determines whether a new trend is established and no filtering of outliers is performed, in contrast to the EWMA based algorithm.
3.5.3 Test scenarios

The scenarios were defined to cover all relevant cases of pattern variation in the data. The scenarios were shared between tuning and benchmarking. The test scenarios were defined to cover the following patterns:

- Measurements without variation or anomalies
- Measurements with isolated instances of anomalies
- Randomized measurements
- Measurements that mimics an actual test case with the most number of successful runs
- Measurements that create a plateau

The simulated data was generated based on average magnitude of actual measurements.

3.5.4 Algorithm tuning, EWMA based algorithm

Algorithm tuning was used to identify the maximum achievable accuracy. The EWMA based algorithm has two parameters affecting the trend-break-detection performance: threshold and exponential smoothing ratio. The threshold was determined in conjunction with the determination of exponential smoothing ratio for the algorithm. It was set up as an optimization problem, where suitable combinations were sought. This task was formalized with an implementation of test code.

Tuning scenarios were defined with simulated data. The scenarios were defined to cover all relevant cases of pattern variation in the data. A correct result for each scenario was defined as reference, this enabled distinction of runs with correct results. Figure 3.3 presents a used tuning scenario. The remaining tuning scenarios that were used are presented in appendix C. KPI measurements are plotted against runs. Blue circles are representing normal circumstances and orange circles are representing anomalous measurements.
A full factorial experiment was used as method for tuning [40]. Repeated runs of the tuning scenarios were carried out with parameter variation to generate all possible combinations of parameter within the specified range. The tuning parameters were ranging from 10-50 % and were varied in steps of 1 %. This granularity was determined to be adequate by experiment. Runs that yielded correct results were collected into a set. This set contained all successful combinations of tuning parameters for that tuning scenario. Only true positives (TP) were considered as correct results.

Figure 3.4 presents a visualization of the tuning process when involving two scenarios. The scenarios are one and seven. The viable combinations are plotted as blue and orange circles for scenarios one and seven. The intersect of the two sets is plotted with red squares. This results in the threshold being 0.12 for an exponential smoothing ratio of 0.1 and a threshold ranging between 0.11 and 0.12 for an exponential smoothing ratio of 0.11.
Tuning scenarios were run consecutively. The intersection of the sets resulting from two runs were calculated for each consecutive run. The combinations of tuning parameters at the end of this process were the ones that were shared between all tuning scenarios. It was discovered that threshold and the exponential smoothing ratio impacted detectability linearly.

Figure 3.5 shows the linear dependency between threshold and the exponential smoothing ratio of one tuning scenario impacting detectability. The formula in (10) was derived from the full factorial output of tuning scenario 1. The threshold is represented as $\delta_{\text{max}}$ and exponential smoothing ratio is represented as $\alpha$ in the formula.

$$\delta_{\text{max}} = -0.1364\alpha + 0.1268$$  \hspace{1cm} (10)

It shows that for a change of threshold by 0.01 the exponential smoothing ratio changes by 0.073.
The complete tuning process included scenarios the four scenarios presented in appendix C. It resulted in the one viable combination of threshold and $\alpha$ which was $0.11$.

### 3.5.5 Algorithm tuning, anomaly based algorithm

The competing algorithm was tuned using the same method as the EWMA based algorithm, the only difference was that the threshold was the only parameter that required tuning. The lowest threshold that was viable for all scenarios was used.

### 3.6 Algorithm benchmarking

Actual trend break data was not available. There was no trivial way of acquiring abundant amounts of information about reported issues from test runs, to correlate variations in the measurements to instances of known problems. There was only one instance with data that was associated with a TR. This single instance was not used as a reference because not being statistically significant. Thus, a synthetic benchmark was devised, it creates the basis for choosing between the two proposed algorithms. The following criteria were established as quality metrics for the algorithm:

- What is the accuracy of the algorithm?
- Is the algorithm sufficiently stable and consistent regardless of the dataset size?
- How does the algorithm handle edge cases?
- How does missing data affect the accuracy of the algorithm?
- What is the minimal acceptable accuracy?
The devised benchmark consisted of the same scenarios as those used in the algorithm tuning process. The results from the benchmarking were represented in a contingency matrix. The implementation with the highest precision would indicate viability of the implementation. The actual benchmark was designed similarly to the tuning process, data for which the outcome was known was supplied.

The expected outcome from the algorithm upon analysis of a benchmark case, defined the reference of the case. The reference for each scenario enabled calculations of the contingency matrix values, which was performed programmatically. To make the benchmark cases complete, sequence ids were generated from 0 to n where n was the number of simulated measurements.

A set of actual data points from the actual runs was composed as a reference for the algorithm evaluation. The nodes with the most runs were selected for statistical significance. The data was limited to one upgrade path, presented in section 3.8. This set was determined to have higher variance than most nodes, which contributed to the complexity of the hardware configuration. It was also included in many more hardware components than the average hardware configuration.

3.7 Correlation analysis

After researching how the current system at the CU team in Ericsson was, based on section 2.1, two correlation factors became important to target for the KPIs. Hence, the correlation analysis was based on the following factors:

- Cell count
- UP version

These two correlation factors also had major significance in the size of the node and what to test during the years. Since the UP version showed what the upgrade path was thus upgrading the software from one track to another track, the importance of this factor was high and necessary to analyze. The third factor, network manager, that was proposed, was minor and had little to no relevance in the generated KPI results, consequently it was discarded.
To evaluate if these factors had any impact on the KPIs, a linearity approach was chosen. It was used to answer the correlation question: does the cell count or UP version affect the downtime duration KPI? If a correlation was detected, a linear regression model would be applied and some correlation value would be extracted. This value could be, i.e. either “the cell count correlates to the KPI downtime duration by 14 % (or X seconds) per cell” or “there is no correlation between cell count and the KPI downtime duration”. The same strategy was applied to the upgrade duration KPI.

The strategy that was used would result in a general overview of the two correlation factors, in how they affected the KPIs. The overview would consist of “no” or a percentage or time of how much of an increase or decrease the factor added to the measured KPI.

To implement and experiment with the strategy above, a similar flow as the one for the algorithm, as seen in figure 3.1, was implemented and used. The collecting-, structuring-, visualizing data steps were identical. However, the analyze data step was modified to be applicable for finding correlation instead. The major difference was that the steps were initiated manually as opposed to the implementation for the algorithm which was designed to be completely automated and self-contained.

### 3.7.1 Analyze data

From the structured data, the necessary information regarding the correlation factors were extracted separately for analysis. To analyze that data, a spreadsheet in Excel was sufficient for plotting and finding some increase or decrease over time in the results, hence a linear relation. This would determine if there was some correlation between the KPIs and the correlation factors individually. To find a stronger connection between the KPIs and the correlation factors, the same method was applied on the factors together, to see if there was any correlation between a certain cell count and a certain UP version against the KPIs.

The correlation analysis was also used as a tool for the test manager to understand why a trend break was detected. With a combination of the algorithms, based on section 3.5, and this correlation analysis, an even more accurate result could be achieved by determining if the trend-breaks were false positives or not.
3.8 Test candidates

The algorithms and correlation analysis were tested on the exact same nodes and tracks. This was due to minimize the amount of possible errors and mistakes in the analysis. Comparing the results from the algorithm on a certain node and track with an analysis on the same setup would give the most accurate conclusion for that node and track. For this reason, two candidates for the nodes and tracks were selected for testing, the candidates were:

- Node KIENB1052 from track 16B to 17A
- Node KIENB1327 from track 16B to 17.Q1

The two nodes were chosen because they had the most reported test runs in the database, as seen in appendix D tables D.1 and D.2. The two tracks were chosen with courtesy of the test manager and because they were the most recent and up-to-date. They were also the two tracks with the most runs for these nodes.

The correlation analysis was also tested independently to find if there was any general correlation in certain cell counts or tracks with the KPIs. This required a limited number of candidates to be able to conclude some correlation, so the selected candidates for this was divided into its correlation factors:

- Cell count
  - All nodes, grouped by number of cells
- UP version
  - Track 16B to 17.Q1, all nodes
  - Track 17A to 17.Q1, all nodes
- Cell count and UP version together
  - All nodes with three and twelve cells, from track 17A to 17.Q1
All nodes for cell count were chosen because, as seen in tables D.1 and D.2 in appendix D, every node had sufficient number of entries for the KPIs. The tracks here were also chosen with courtesy of the test manager and because they were the most recent, up-to-date, and available in the database. The two correlation factors together were delimited because the number of nodes and KPIs with those cell counts had the most tests executed. The number of nodes and track combinations were also too many to cover.
4 Results

This chapter presents the results. The results of the algorithm implementation and the trend-break-detection system are presented in section 4.1, as well as the resulting accuracy of the two algorithms that were devised. Section 4.2 presents the results of the precision evaluation of the algorithms. Section 4.3 presents the results from the correlation analysis. Finally, results from executions on actual KPI measurements are presented in section 4.4.

4.1 Model used in the prototype

The prototype was devised as the trend-break-detection system. It served as the basis for implementation and evaluation of the trend-break-detection algorithms. The implementation was based on the model presented in section 3.3. The data collection step was regarded as a method and thus are the data collection details presented in section 3.3.

4.1.1 Structure data

The collected data was structured in a KPI class. This class contained all information relevant to the algorithm. A UML representation of the class is presented in figure 4.1. The fromUpgradePackage and toUpgradePackage fields were remapped to track names instead of UP versions as part of the filtering step. Each measurement tracks whether it is a trend break or not. Furthermore, the calculated EWMA value is stored if applicable for the algorithm. The methods of the class were only accessor and mutator methods, method() in the figure is just a placeholder for these.
The KPI values were stored in a list, this list was filtered based on node and track. The resulting list of values was passed on to the data analysis step.

4.1.2 Data analysis
The EWMA based algorithm implemented the formula (8) presented in 3.5.1. The calculation of EWMA values was performed using the EMA() function in the TTR package of R. Implementation details of the Java code are available in appendix E. The following actual arguments were supplied: a vector of measurements, the number of periods to average over which was set to one as the data was not periodical, and finally the exponential smoothing ratio. The determination of exponential smoothing ratio was carried out in conjunction with the algorithm tuning in section 3.5.4. The result from the EWMA calculation was an array of the EWMA for each value that was supplied. The anomaly based algorithm was an implementation of the formula (9) presented in 3.5.2. It was not dependent upon R.

4.1.3 Present results from prototype
The presentation of the results was done in two ways. Both a rudimentary text output and a visualization in a graph. The text output consisted of the number of identified trend breaks and the test suite sequence ids of the affected runs. Visualization of data was achieved through the plotting functionality of R using the plot() function. The output was saved to a Portable Network Graphics (PNG), .png file with the png() function. The plotting used a sequence of functions starting and ending with the png() and dev.off() functions. Data was prepared in two steps, normal measurements and trend breaks were ordered into sub-arrays stably, in other words, with respect to the position in the global sequence. The sub-arrays were plotted individually using the plot() function with different parameters.
The sequence id of the test suites was plotted below data points representing trend breaks with the `text()` function. Sequence ids were only plotted with trend breaks because of the limited plotting functionality of R. Plotting sequence ids with each point would be impossible to interpret in a situation with many data points as the text would overlap. The purpose of plotting the sequence ids was to have a reference to the test suite that were run, to allows for retrieval using internal tools. Once retrieved, review of the logs associated with the abnormal test suite could be inspected. A legend was generated with the `legend()` function, it was positioned in the lower right hand corner.

The visualization generated by the system is presented in figure 4.2. It consists of a run of test scenario seven. The x-axis represents the runs and the y-axis represents the measured KPI. Detected trend breaks were plotted as red equilateral triangles, the other data points were plotted as blue circles. A simple linear regression line was plotted as a visual aid for summarizing the general trend over the plotted period of runs.

![Upgrade duration](image)

*Figure 4.2: Example of output from visualization of benchmark scenario seven generated by the EWMA based algorithm.*
To calculate the simple linear regression with R, a frame was generated to hold the data in a structured way, as this was a prerequisite. The function data.frame() was used, its formal parameters were two vectors of points in time and the values at the corresponding times. The calculation was done with the lm() function which takes the vectors and the frame as actual arguments. The results were acquired by evaluating variables of R and using the parsing strategy provided by Rserve which involves casting. A basic implementation is available in appendix E.

4.2 Algorithm evaluation

The achieved performance of the algorithms is presented in tables 4.1 and 4.2. These form the basis for evaluation with the formulas (5), (6), and (7) presented in section 2.5.2.

The accuracy of the EWMA based algorithm was 97.54 % for the simulated data, the precision was 82.35 % and finally the recall was 100 %.

**Table 4.1: Contingency matrix for the EWMA based algorithm.**

<table>
<thead>
<tr>
<th></th>
<th>Predicted no</th>
<th>Predicted yes</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actually no</td>
<td>105 (TN)</td>
<td>3 (FP)</td>
<td>108</td>
</tr>
<tr>
<td>Actually yes</td>
<td>0 (FN)</td>
<td>14 (TP)</td>
<td>14</td>
</tr>
<tr>
<td>Total</td>
<td>105</td>
<td>17</td>
<td>122</td>
</tr>
</tbody>
</table>

For the simulated data, the accuracy of the anomaly based algorithm was 94.26 %, the precision was 76.92 % and the recall was 71.42 %.

**Table 4.2: Contingency matrix for the anomaly based algorithm.**

<table>
<thead>
<tr>
<th></th>
<th>Predicted no</th>
<th>Predicted yes</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actually no</td>
<td>105 (TN)</td>
<td>3 (FP)</td>
<td>108</td>
</tr>
<tr>
<td>Actually yes</td>
<td>4 (FN)</td>
<td>10 (TP)</td>
<td>14</td>
</tr>
<tr>
<td>Total</td>
<td>109</td>
<td>13</td>
<td>122</td>
</tr>
</tbody>
</table>
4.3 Correlation

The achieved results are based on sections 3.7 and 3.8, by analyzing the different correlation factors.

4.3.1 Cell count

For the cell count analysis, all existing measured KPIs are extracted and grouped by their cell counts. With grouping the measurements by cell count, a median is calculated for each cell count. Figure 4.3 shows the median of all the measured KPIs, from all nodes, for upgrade duration. The KPI is measured in seconds. The plotted linear line estimates that for every added cell, the KPI increases with 59.8 seconds, which is the slope of the line.

Figure 4.3: Median of every measured upgrade duration, grouped by cell count.

Figure 4.4 shows the median of all the measured KPIs, from all nodes, for downtime duration. The KPI is measured in seconds. The plotted linear line estimates that for every added cell, the KPI increases with 3.9 seconds, which is the slope of the line.
4.3.2 UP version

For the UP version analysis, all existing measured KPIs for the specified UP version is extracted. Figure 4.5 shows the measured KPI, downtime duration, for UP versions 17A to 17.Q1. The KPI is measured in seconds and it is plotted over time. The result show that there is high fluctuation for this setup in the time span of 5.5 months. This means that for the UP version alone, there is no correlation between the UP version and the KPIs, since there is no clear change over time.
Figure 4.5: The measured KPI, downtime duration, for every node from track 17A to 17.Q1. The measurements are in seconds over a time span of 5.5 months.

However, this is one out of four analyzed tracks and the remaining ones show the same type of fluctuation in the data. Due to the similarity in fluctuations in all four figures, only one is presented here and the remaining ones are available in appendix F, figures F.3, F.4 and F.5.

4.3.3 Cell count and UP version together

The results for the two correlation factors, cell count and UP version, together showed that there is no correlation with them when combined. Figure 4.6 shows that there is high fluctuation in the twelve cell nodes and the three cell nodes are stable, on one track for upgrade duration. The twelve cell nodes have many high measured values as well as multiple low values, whereas the three cell nodes only have low measured values overall.
4.4 Algorithm execution and correlation analysis

The exponential smoothing ratio and threshold were both at 0.11 for all runs when running the two algorithms on actual stored measurement data.

4.4.1 Node KIENB1052 from track 16B to 17A

The following results were generated from the first candidate, node KIENB1052, which had three cells, from track 16B to 17A. When running the EWMA based algorithm, the results showed that it found four trend breaks. This can be seen in figure 4.7. For the anomaly based algorithm, with the same setup, the results showed that the algorithm detected one trend break, as in figure 4.8. For both runs, it also showed that the trend is going down, based on the plotted linear regression line. From the previous result on cell count correlation above, the expected result for the algorithm was to have low measured values for the KPIs and the outliers be trend breaks. This was true for these cases, since the majority were low measured KPI values. The highest value was around 560 seconds. However, the UP version correlation was shown to have no correlation and this case could not determine if that was true or not.
Figure 4.7: All measured KPIs for upgrade duration from node KIENB1052, from track 16B to 17A using the EWMA based algorithm.

The algorithm executions on this setup for downtime duration can be seen in appendix F, figures F.7 and F.8. They both detected the same one trend break.
4.4.2 Node KIENB1327 from track 16B to 17.Q1

The following results were generated from the second candidate, node KIENB1327 from track 16B to 17.Q1, which had twelve cells.

Upgrade duration on this setup for the EWMA based algorithm resulted in zero trend breaks. However, the anomaly based algorithm detected one trend break and it can be seen in figure 4.9. Since the node had twelve cells, the expected upgrade duration, based on section 4.3.3, was to have a fluctuating measured KPI with multiple trend breaks. Since this was not the case, it was less likely to have any correlation with the cell count and UP version together. The measured KPIs were not fluctuating but they were high in magnitude, compared to the three-cell node. The lowest value was approximately 1000 seconds.

![Image](image-url)

Figure 4.9: All measured KPIs for upgrade duration from node KIENB1327, from track 16B to 17.Q1 using the anomaly based algorithm.

For downtime duration on the same node and change of track, the EWMA based algorithm detected two trend breaks, presented in figure 4.10. Whereas the anomaly based algorithm detected seven trend breaks, presented in figure 4.11.
Figure 4.10: All measured KPIs for downtime duration from node KIENB1327, from track 16B to 17.Q1 using the EWMA based algorithm.

Figure 4.11: All measured KPIs for downtime duration from node KIENB1327, from track 16B to 17.Q1 using the anomaly based algorithm.
5 Analysis and discussion

This chapter analyzes and discusses the results presented in chapter 4, the implemented prototype and correlation. The analysis is also written in relation to the goals from section 1.2 and methodology in chapter 3. Section 5.1 focuses on the first main goal, the prototype, and the analysis for the second main goal is presented in 5.2. An overview of the expectations is presented in 5.3. Sections 5.4 targets alternate methods.

For this thesis, there are no environmental impacts identified. However, one can argue for environmental impacts since the project is within the field of upgrading software on hardware. The hardware can waste or save energy based on the quality of the software and upgrade process. But, since the core of this thesis is to use KPIs that are measured during the upgrade, and not the upgrade itself, an environmental impact is not relevant. There is also no ethical connection identified in this work. However, there is an economical aspect. Usage of a tool that employs the devised trend-break-detection algorithm, might potentially yield financial savings, as the time required for manual analysis is reduced. The manual analysis is only concerned with validation, whether detected trend breaks are problems that need to be addressed. Furthermore, when trend breaks are reported close to the occurrence it might potentially be less time consuming to identify the root cause. Lastly, the social aspect identified is that the test managers at Ericsson lose part of their tasks. This means that the human resource minimizes and less work is available. However, since there is no validation of the trend-breaks in the form of TRs, the social aspect is less relevant.
5.1 Interpreting results of prototype

Figures 4.7 through 4.11 constitutes the basis for comparison of the behavior of the algorithms. In figure 4.7 the EWMA based algorithm marks four measurements as trend breaks while the anomaly based algorithm marks one for the same data. This indicates the trend break identifying properties of the EWMA based algorithm in comparison to the change identifying properties of the anomaly based algorithm. This distinction is prevalent in all presented cases. When comparing figures 4.10 and 4.11 the EWMA based algorithm does not identify any trend breaks, which is more accurate regarding trend-break-detection. Whereas the anomaly based algorithm identifies one trend break, which is more relevant to the way of working at Ericsson. The outlier filtering characteristics of the EWMA based algorithm becomes apparent in this case. While for the anomaly based algorithm the consistency of instantaneous change detection appears in contrast.

The behavior of the EWMA based algorithm appears inconsistent with regards to trends when comparing figure 4.10 with figure 4.11. This is because outliers appear as trend breaks. It is arguable that a significant outlier needs investigation because something unusual might have occurred. The anomaly based algorithm detects many contiguous trend breaks due to the variability in the measurements which might be interpreted as an indication of a threshold that is not suitable or a desired feature. Because the situation of fluctuating measurements might indicate recurring problems or an overarching problem that need further investigation. Unrealistic worst cases were identified during benchmarking these significantly impacted the precision of the algorithms negatively. The scenarios involving them were discarded. Analysis of these scenarios might have provided some additional insight in the general behavior of the two algorithms.
The disparity of the overall accuracy is low at 3.28 % between the algorithms. This indicates equivalent overall performance. However, inspection of the precision and recall components of the accuracy provides additional insights. The EWMA based algorithm has an advantage in precision at 82.35 % versus the anomaly based algorithm which is at 76.92 %. This indicates that the EWMA based algorithm finds more measurements that are relevant for the defined scenarios. The EWMA based algorithm appears overfitted with regards to recall which is at 100 % in contrast to the anomaly based algorithm which is at 71.42 %. This indicates that it does not miss any relevant measurements defined in the scenarios.

5.1.1 Accuracy and reliability of evaluation

The reliability and the measured accuracy of the algorithms might be influenced by the fixed test scenarios. There is no variation within the scenario between runs in the way test scenarios are designed. Furthermore, the test scenario frequency is constant. The scenarios that are more realistic appear the same number of times as the scenarios that are not. Not having variation within scenarios makes the comparison between the algorithms more reliable as the algorithm is the only variable. Otherwise, with multiple runs, some type of average would need to be employed to make the results comparable. The uniform distribution of frequencies is a drawback for the reliability of the benchmark. It might potentially be an insufficient foundation for determining accuracy, as the scenarios are skewed towards being unrealistic.

5.1.2 Tuning methodology

The magnitude of tuning steps of 1 % and the upper limit of 50 % for the tuning parameters were not evaluated exhaustively. The tuning was based on inspection of a limited number of cases. Further evaluation might have introduced additional usable ranges for the parameters. However, regarding the set goals, the parameters identified with the set ranges yielded sufficient results. Using different data for tuning and benchmarking would have yielded more reliable results as it would have employed methods for preventing overfitting.

5.1.3 Amount of data for correlation

The number of measurements available for correlation were in some cases insufficient for statistical significance. This was accounted for during the selection of data.
5.2 Interpreting results of correlation

The analysis and discussion of the correlation is divided into the correlation factors alone and combined with the algorithm.

5.2.1 Correlation between cell count and time

The cell count for all nodes, which is proven to influence the measured KPIs, resulted in an interesting increase. Since the increase is 59.8 seconds for upgrade duration and 3.9 seconds for downtime duration, the obvious conclusion is that they both correlate to the KPIs. It is also a significant difference in increase between the two KPIs. Upgrade duration is 55.9 seconds higher than downtime duration, which means that there is a stronger correlation between the cell count and upgrade duration than cell count and downtime duration. This could be due to how the KPIs are measured and how many smaller inner tasks the two KPIs measure.

5.2.2 Significance of influence from cell count

Based on the difference between the two KPIs being so significant and that they are measured correctly, the number of tasks that upgrade duration measures has to be higher than downtime duration. There is also the possibility of incorrect measurements of KPIs in the tests. However, to determine if that is the case, the measurement method is assumed to be correct, as there is no way to get validated by Ericsson within the time frame of this project. Furthermore, there is no information about whether cell count is the only contributing factor to the increased KPI measurements. It is important to note that there is no information to draw conclusions whether change in cell count is the causality to change in KPI measurements.

5.2.3 Notes on cell count correlation

An important aspect to point out is that the measurements for cell count of one, four, and nine are outliers as they are based on two to three measurements. However, there are few nodes with these configurations, therefore, few runs were performed on those for the selected track. Regarding these runs as outliers would have generated more reliable results. However, if the outliers are filtered the number of measurements per cell count are significantly reduced resulting in insufficient data for correlation analysis. With the chosen approach, there is at least an indication, even though not completely reliable, of the general cell count behavior.
5.2.4 Correlation methodology
There was no knowledge about the type of probability distribution that governed the cell count. It was not known whether the data was skewed within the distribution. Median was chosen as the measure of central tendency because the lack of information about the skewness. It relies on the assumption that the measurements were distributed normally.

5.2.5 Correlation between UP and time
The UP versions 16B to 17.Q1 and 17A to 17.Q1 that were chosen shows high fluctuations in the KPI measurements. Furthermore, there is no distinct correlation between the number of runs and UP version over a 5.5-month period. In other words, there was no significant general trend in the measurements.

The correlation factors were not analyzed previously at Ericsson, so a known risk of the two chosen factors is that there could be no correlation between them and the KPIs. Since this becomes apparent for UP version, it is hard to determine if it has any impact on the KPIs at all. Ultimately, since it is unknown whether UP version has any impact or not, it is concluded that it alone has no correlation with the KPIs.

5.2.6 Comparison of nodes with different cell counts
When comparing upgrade duration for a node with three cells against a node with twelve cells, correlation is not apparent between the KPI measurements and time. However, patterns indicating other features arise as a side effect. The node with twelve cells have significantly higher variation and fail-rate than the node with three cells over a 5.25-month period. However, this approach might not be the most efficient way of determining the information that appears as side effects. Whereas some comparison of statistical dispersion in conjunction with central tendency could be more effective in providing the same information about side effects.
The correlation results are also subjects to inaccuracy because of the unknown knowledge in how complex the correlation factors might be. Thus, the correlation only shows if there is an increase or decrease at best. However, the cell count did present an increase of 59.8 seconds per added cell by using the median of all the measured KPIs, but by looking at figure F.1 in appendix F, a significant fluctuation is present for most of the cell counts. Based on the figure and that UP version has no correlation impact, it means that there must be other correlation factors that weigh in on the correlation verdict, due to the complexity of the system. The other factors are unknown to the authors of this thesis and even if they are to be used, there is no stored data regarding those in the database.

5.2.7 Correlation in conjunction with algorithm

It appears that using the knowledge from the identified correlation in conjunction with the results of the algorithms is no indicator of the cause of the trend break. It could possibly be the case that the amount of correlation knowledge is insufficient. However, it is likely that the method itself is lacking. Briefly mentioned in related works, methods that use logs as the basis for root cause analysis are proven to work with varying accuracy [36].

5.3 Goals and expectations

Since the first main goal is to detect trend breaks, one method for detecting trend breaks is devised and another method for detecting changes is devised as a contender. The second main goal is to determine whether some correlation is present between hardware configuration and trend breaks that are detected by the algorithm. No such correlation is apparent. Both goals are determined to be fulfilled based on the chosen methodologies. The only aspect that might have been useful to improve upon are tuning and benchmarking scenarios that are tailored towards the two KPI types.

5.4 Alternate methodology

The alternate methodologies, presented in chapter 3, resulted in additional details that requires analysis and discussion.
5.4.1 **Machine learning**

If each anomalous data point would have been associated with a TR, supervised machine learning could potentially have been a method which generated more accurate results. Manually associating TRs with runs was too time intensive for the scope of this project, as mentioned in section 2.1.

A significant advantage of using machine learning methods is that libraries are available for devising the algorithm. An important feature of the libraries is that validation is bundled.

5.4.2 **Root cause analysis**

The two main methods for root cause analysis are analyzing changes in logs exclusively and analyzing changes in logs with source code as the reference. The latter has an advantage of higher accuracy [36].

5.4.3 **Type of contending algorithm**

Two algorithms of the same type might have been devised for the comparison, but it would most likely not have given insights to different characteristics. However, a comparison of two different algorithms has contributed with these insights. A comparison of two algorithms of the same type would have provided insights on which approach yielded higher accuracy. However, it was not given that trend break detection was most suitable for the problem and available data.

5.4.4 **Type of benchmark**

By using two benchmarks, one designed for trend-break-detection and another for change-detection might have been more insightful in determining accuracy. This was however regarded as out of scope with regards to the posed time constraints. The used benchmark was designed to be general instead.

5.4.5 **Method for determining change**

Using percentage change for comparing delta was only observed to have a negative impact on the EWMA based algorithm. This was in a simulated situation when the measurements were increasing linearly and rapidly. However, such behavior is extremely unlikely for the KPI measurements and no other drawbacks were identified with this approach.
5.4.6 Integration with internal tools

The internal tool, ADAT, that Ericsson uses for their data analysis is one of the alternatives that can be applicable to the data that is used in this project. Since ADAT uses sliding window with quantile regression and daily analysis, it is a good contender to the developed algorithms. The similarities are that the algorithms also have sliding window and daily analysis. However, ADAT has a dynamic threshold, called confidence band, which the developed algorithms do not have and the regression type is different.
6 Conclusions

The trend-break-detecting prototypes that are implemented have met the desired goals and they have shown that there are trend breaks in the test result data. The detected trend breaks compared with the two correlation factors show that there is no correlation between them.

The results for the algorithm prototypes showed that the EWMA based algorithm had an accuracy of 97.54% on simulated data and the anomaly based algorithm had 94.26%. The simulated data was very similar to real data from test results.

The correlation analysis show that there only is correlation between cell count and KPIs. UP version has no correlation with the KPIs. The results from the correlation analysis can give an indication of how different hardware configurations affect measured values. It is also shown that for every added cell, upgrade duration increases with 59.8 seconds and downtime duration increases with 3.9 seconds, while UP version has no effect.

6.1 Future works

During the automatic trend-break-detection analysis, a flaw of the threshold was detected which indicates that another threshold selection might be better. It might therefore be more suitable to investigate the usability of a dynamic threshold and estimate its effect on the algorithm. To evolve the algorithm further, and make it more suitable for Ericsson, further research, and accessibility to ADAT is worth considering. Customizing the tool can allow the algorithm to extend ADAT and take advantage of its alternate methods, to gain a higher accuracy in trend-break-detection.

Furthermore, since this thesis is focused on trend-break-detection, an alternate approach can be to focus on change-detection. For data that is generated nightly, a change-detection approach can be more applicable and worth investigating.
Lastly, other solutions to solve the problem of this thesis is to utilize machine learning or data mining. The two approaches can result in a more complex understanding of the data and contribute to an improved correlation. However, to use them, the data that was used in this project has to be structured and tagged properly.
7 References


[28] Li X. Encapsulating Matlab linear regression function as web service.  
In: 2010 International Conference on Networking and Digital Society.  
Wenzhou, China. 30-31 May 2010. p. 76-79.


[33] Zhou ZG, Tang P. Improving time series anomaly detection based on exponentially weighted moving average (EWMA) of season-trend model


### Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAT</td>
<td>Automatic Data Analysis Tool, internal tool at Ericsson.</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface.</td>
</tr>
<tr>
<td>BFAST</td>
<td>Breaks For Additive Seasonal and Trend.</td>
</tr>
<tr>
<td>CDF</td>
<td>Cumulative Distribution Function.</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit.</td>
</tr>
<tr>
<td>CRAN</td>
<td>Comprehensive R Archive Network.</td>
</tr>
<tr>
<td>CU team</td>
<td>Configure and Upgrade team, one of several Quality Assurance teams within Ericsson.</td>
</tr>
<tr>
<td>EWMA</td>
<td>Exponential Weighted Moving Average, a type of weighted average.</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative.</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive.</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface.</td>
</tr>
<tr>
<td>G1</td>
<td>The first generation of a hardware component in a Signaling Transfer Point (STP).</td>
</tr>
<tr>
<td>ITK</td>
<td>Insights Tool Kit, internal tool at Ericsson.</td>
</tr>
<tr>
<td>JRI</td>
<td>Java R Interface.</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine.</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Indicator, a metric for determining the performance of some hardware component in a test case.</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis.</td>
</tr>
<tr>
<td>QA</td>
<td>Quality Assurance, verification that product requirements are fulfilled.</td>
</tr>
<tr>
<td><strong>ACRONYMS AND ABBREVIATIONS</strong></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>RBS</strong></td>
<td>Radio Base Station, a hub in a cellular network.</td>
</tr>
<tr>
<td><strong>ROC</strong></td>
<td>Receiver Operating Characteristic.</td>
</tr>
<tr>
<td><strong>STP</strong></td>
<td>Signaling Transfer Point.</td>
</tr>
<tr>
<td><strong>STR</strong></td>
<td>Season-Trend model Residuals.</td>
</tr>
<tr>
<td><strong>TAR DB</strong></td>
<td>Test Automation Result Database, a database containing all test results.</td>
</tr>
<tr>
<td><strong>TCP</strong></td>
<td>Transmission Control Protocol.</td>
</tr>
<tr>
<td><strong>TDR</strong></td>
<td>TTS Dynamic Report, a database warehouse that mirrors TAR DB and several other databases. TTS is a collection of three tools that were used previously within Ericsson, each tool is denoted in the acronym.</td>
</tr>
<tr>
<td><strong>TGF</strong></td>
<td>Test Generic Framework, an internal booking system, scheduler and test executor at Ericsson.</td>
</tr>
<tr>
<td><strong>TN</strong></td>
<td>True negative.</td>
</tr>
<tr>
<td><strong>TP</strong></td>
<td>True positive.</td>
</tr>
<tr>
<td><strong>TR</strong></td>
<td>Trouble Report, a detailed bug report.</td>
</tr>
<tr>
<td><strong>TTR</strong></td>
<td>Technical Trading Rules.</td>
</tr>
<tr>
<td><strong>UP</strong></td>
<td>Upgrade package.</td>
</tr>
</tbody>
</table>
Appendix A - Maven dependencies

The dependencies that were used are presented in listing A.1.

```xml
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.41</version>
  </dependency>
  <dependency>
    <groupId>com.github.lucarosellini.rJava</groupId>
    <artifactId>JRI</artifactId>
    <version>0.9-7</version>
  </dependency>
  <dependency>
    <groupId>org.rosuda.REngine</groupId>
    <artifactId>Rserve</artifactId>
    <version>1.8.1</version>
  </dependency>
</dependencies>

Listing A.1: Maven dependencies from the pom.xml that were used
Appendix B - Database schemas

The MySQL database schemas of the database that was holding the relevant data are presented in this section. Table B.1 holds the schema for test cases and Table B.2 holds the schema for test suites.

### Table B.1: Test case table in the database TDR.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeqId</td>
<td>int(1) unsigned</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td>auto_increment</td>
</tr>
<tr>
<td>TmId</td>
<td>varchar(80)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TmHeadline</td>
<td>varchar(250)</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TestArea</td>
<td>varchar(50)</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TestObject</td>
<td>varchar(250)</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TestName</td>
<td>varchar(250)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>ThcTcJavaName</td>
<td>varchar(250)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Reported2TM</td>
<td>varchar(32)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TestsuiteRunsSeqId</td>
<td>int(1) unsigned</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Verdict</td>
<td>varchar(30)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Cause</td>
<td>varchar(32)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>StartDateTime</td>
<td>datetime</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>EndDateTime</td>
<td>datetime</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TargetHw</td>
<td>text</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TrId</td>
<td>varchar(64)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>CtcProperties</td>
<td>text</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Comment</td>
<td>mediumtext</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>AdditionalKPI</td>
<td>mediumtext</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>timestamp</td>
<td>timestamp</td>
<td>YES</td>
<td>MUL</td>
<td>CURRENT_TIMESTAMP</td>
<td>on update CURRENT_TIMESTAMP</td>
</tr>
<tr>
<td>testdata</td>
<td>text</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>qualityarea</td>
<td>varchar(25)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>focusarea</td>
<td>varchar(10)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>cause_of_error</td>
<td>varchar(200)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Checked</td>
<td>tinyint(1)</td>
<td>YES</td>
<td>MUL</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>indicator</td>
<td>varchar(16)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>parentId</td>
<td>varchar(45)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>misc_duplex</td>
<td>varchar(10)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>misc_suiteName</td>
<td>varchar(300)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>misc_du</td>
<td>varchar(10)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>testCriteria</td>
<td>varchar(45)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>releaseCriteria</td>
<td>varchar(45)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>keywords</td>
<td>varchar(50)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Null</td>
<td>Key</td>
<td>Default</td>
<td>Extra</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------</td>
<td>------</td>
<td>------</td>
<td>---------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>SeqId</td>
<td>int(1) unsigned</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td>auto_increment</td>
</tr>
<tr>
<td>TmId</td>
<td>varchar(128)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TmHeadline</td>
<td>varchar(250)</td>
<td>NO</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Build</td>
<td>varchar(50)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>BuildRecord</td>
<td>varchar(19)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Iteration</td>
<td>varchar(250)</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>UpSeqId</td>
<td>int(1)</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>ThcTcDeliveryTblSeqId</td>
<td>int(1)</td>
<td>NO</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>RunUser</td>
<td>varchar(30)</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TmUser</td>
<td>varchar(30)</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TheServer</td>
<td>varchar(60)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Regression</td>
<td>tinyint(1)</td>
<td>NO</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>RunType</td>
<td>varchar(6)</td>
<td>NO</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>varchar(16)</td>
<td>NO</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Repetition</td>
<td>varchar(30)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>AutoReport</td>
<td>tinyint(1)</td>
<td>NO</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>StartDateTime</td>
<td>datetime</td>
<td>NO</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>EndDateTime</td>
<td>datetime</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>LogFilePath</td>
<td>varchar(256)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>NoOfCtcsInSuite</td>
<td>int(1)</td>
<td>NO</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Comment</td>
<td>text</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TmAssetRegistry</td>
<td>varchar(64)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TmSite</td>
<td>varchar(30)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>SuiteSequence</td>
<td>varchar(30)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>MailOn</td>
<td>varchar(30)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>MailTo</td>
<td>varchar(60)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Capabilities</td>
<td>varchar(2048)</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>JobId</td>
<td>int(10) unsigned</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TgfRerunOf</td>
<td>int(10) unsigned</td>
<td>NO</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>timestamp</td>
<td>timestamp</td>
<td>YES</td>
<td>CURRENT_TIMESTAMP</td>
<td>on update CURRENT_TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>TgfActivityId</td>
<td>int(10) unsigned</td>
<td>YES</td>
<td>NULL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TgfActivitySpec</td>
<td>varchar(70)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>TgfActivityType</td>
<td>varchar(32)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C - Tuning scenarios

This section presents the five scenarios that were used for tuning. Figure C.1 presents tuning scenario one with twenty runs and one anomaly. Figure C.2 presents tuning scenario three, a scenario twenty runs but without anomalies. Figure C.3 presents tuning scenario four with twenty runs and one anomaly. Finally figure C.4 presents tuning scenario five with fifty runs and three anomalies.

Figure C.1: Tuning scenario one with one anomaly at the tenth run.

Figure C.2: Tuning scenario three, without anomalies.
Figure C.3: Tuning scenario four with one anomaly at the third run.

Figure C.4: Tuning scenario five with four anomalies at runs: 7, 8, 9, 27, 28, 29, 44, 45, and 46.
Appendix D - Specification for all nodes

This section presents the raw information about number of test runs and cell count for each node. The information is divided per KPI, upgrade duration is presented in table D.1 and downtime duration is presented in table D.2.

<table>
<thead>
<tr>
<th>Node</th>
<th>Pass &amp; Fail</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>KIENB4023</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>KIENB1092</td>
<td>28</td>
<td>2</td>
</tr>
<tr>
<td>KIENB2074</td>
<td>27</td>
<td>2</td>
</tr>
<tr>
<td>KIENB3004</td>
<td>35</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3038</td>
<td>52</td>
<td>3</td>
</tr>
<tr>
<td>KIENB1043</td>
<td>42</td>
<td>3</td>
</tr>
<tr>
<td>KIENB2089</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3098</td>
<td>46</td>
<td>3</td>
</tr>
<tr>
<td>KIENB2042</td>
<td>53</td>
<td>3</td>
</tr>
<tr>
<td>KIENB1052</td>
<td>61</td>
<td>3</td>
</tr>
<tr>
<td>KIENB1048</td>
<td>51</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3078</td>
<td>28</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3021</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>KIENB1045</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>KI20ENB2537</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3023</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3012</td>
<td>24</td>
<td>3</td>
</tr>
<tr>
<td>KIENB4033</td>
<td>48</td>
<td>4</td>
</tr>
<tr>
<td>KIENB3044</td>
<td>N/A</td>
<td>5</td>
</tr>
<tr>
<td>KIENB3099</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>KIENB1046</td>
<td>23</td>
<td>6</td>
</tr>
<tr>
<td>KIENB4034</td>
<td>60</td>
<td>6</td>
</tr>
<tr>
<td>KIENB1044</td>
<td>36</td>
<td>9</td>
</tr>
<tr>
<td>KIENB4044</td>
<td>29</td>
<td>9</td>
</tr>
<tr>
<td>KIENB1047</td>
<td>50</td>
<td>12</td>
</tr>
<tr>
<td>KIENB1327</td>
<td>134</td>
<td>12</td>
</tr>
<tr>
<td>KIENB3066</td>
<td>17</td>
<td>N/A</td>
</tr>
<tr>
<td>KIENB3007</td>
<td>39</td>
<td>N/A</td>
</tr>
<tr>
<td>KIENB4016</td>
<td>15</td>
<td>N/A</td>
</tr>
<tr>
<td>KIENB9211</td>
<td>21</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Table D.2: All existing nodes in the database with number passed and failed KPIs and cell count, for downtime duration.

<table>
<thead>
<tr>
<th>Node</th>
<th>Pass &amp; Fail</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>KIENB4023</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>KIENB1092</td>
<td>27</td>
<td>2</td>
</tr>
<tr>
<td>KIENB2074</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>KIENB3038</td>
<td>51</td>
<td>3</td>
</tr>
<tr>
<td>KIENB2089</td>
<td>32</td>
<td>3</td>
</tr>
<tr>
<td>KIENB1052</td>
<td>61</td>
<td>3</td>
</tr>
<tr>
<td>KIENB1045</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3021</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>KIENB2042</td>
<td>52</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3078</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>KI20ENB2537</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3098</td>
<td>46</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3012</td>
<td>24</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3004</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>KIENB1043</td>
<td>41</td>
<td>3</td>
</tr>
<tr>
<td>KIENB1048</td>
<td>44</td>
<td>3</td>
</tr>
<tr>
<td>KIENB3023</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>KIENB4033</td>
<td>48</td>
<td>4</td>
</tr>
<tr>
<td>KIENB3044</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>KIENB3099</td>
<td>25</td>
<td>6</td>
</tr>
<tr>
<td>KIENB1046</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>KIENB4034</td>
<td>58</td>
<td>6</td>
</tr>
<tr>
<td>KIENB1044</td>
<td>35</td>
<td>9</td>
</tr>
<tr>
<td>KIENB4044</td>
<td>30</td>
<td>9</td>
</tr>
<tr>
<td>KIENB1327</td>
<td>133</td>
<td>12</td>
</tr>
<tr>
<td>KIENB1047</td>
<td>47</td>
<td>12</td>
</tr>
<tr>
<td>KIENB4016</td>
<td>15</td>
<td>N/A</td>
</tr>
<tr>
<td>KIENB3066</td>
<td>17</td>
<td>N/A</td>
</tr>
<tr>
<td>KIENB3007</td>
<td>38</td>
<td>N/A</td>
</tr>
<tr>
<td>KIENB9211</td>
<td>21</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Appendix E - Java code snippets

This section presents the Java methods that were used in the prototype. Listing E.1 presents the Java method for calculating simple linear regression using a connection to an Rserve instance. Listing E.2 presents the implemented Java method for calculating EWMA through Rserve.

```java
public static void calculateSimpleLinearRegressionCoefficients(List<Integer> values) {
    RConnection connection = null;
    double[] coefficients = null;

    try {
        connection = new RConnection(IP, PORT);
        List<Integer> runs = new ArrayList<>();
        for (int i = 0; i < values.size(); i++) {
            runs.add(i);
        }

        connection.eval("v=data.frame(t=" +
                        vectorize(runs) + ", k=" + vectorize(values) + ")");
        connection.eval("Coefficients=lm(formula=k ~ t,
                         data=v)");

        REXP c = connection.eval("Coefficients");
        Object o =
        connection.eval("Coefficients").asList().get(0);
        coefficients = ((REXP)o).asDoubles();
    } catch (RserveException e) {
        e.printStackTrace();
    } catch (REXPMismatchException e) {
        e.printStackTrace();
    } finally {
        connection.close();
    }
    return coefficients;
}
```

**Listing E.1:** simple linear regression calculation in R via Rserve using Java.
public static double[] calculateEWMA(List<Integer> values, double alpha) {
    RConnection connection = null;
    double[] ewma = null;

    try {
        connection = new RConnection(IP, PORT);

        String v = values.toString();
        v = v.substring(1, v.length() - 1);

        String rVector = "c(" + v + ");
        connection.eval("ewma=TTR::EMA(" + rVector + ",
        n=1, ratio=" + alpha + ")");
        ewma = connection.eval("ewma").asDoubles();
    } catch (RserveException e) {
        e.printStackTrace();
    } catch (REXPMismatchException e) {
        e.printStackTrace();
    } finally {
        connection.close();
    }

    return ewma;
}

Listing E.2: EWMA calculation in R via Rserve using Java.
Appendix F - Correlation analysis graphs

Additional graphs from correlation analysis are presented here. Figure F.1 and F.2 present the plotted raw upgrade duration and downtime duration measurements per cell count respectively. An simple linear regression applied to establish the correlation. Figure F.3 and F.4 present the plotted upgrade duration and downtime duration measurements over time for the track 16B-17.Q1. Figure F.5 is the same type of plot but for track 17A-17.Q1. Figure F.6 presents a plot of all runs, for track 17A-17.Q1, on nodes with three and twelve runs respectively. Finally figures F.7 and F.8 presents the algorithm executions on node KIENB1052 and track 16B to 17A, for downtime duration.

![Graph](image)

**Figure F.1:** The measured upgrade duration for every node, grouped by cell count.
Figure F.2: The measured downtime duration for every node, grouped by cell count.

Figure F.3: The measured upgrade duration for every node, from track 16B to 17.Q1.
Figure F.4: The measured downtime duration for every node, from track 16B to 17.Q1.

Figure F.5: The measured upgrade duration for every node, from track 17A to 17.Q1.
Figure F.6: The measured downtime duration for every node with three and twelve cells, from track 17A to 17.Q1.

Figure F.7: All measured KPIs for downtime duration from node KIENB1052, from track 16B to 17A using the EWMA based algorithm.
Figure F.8: All measured KPIs for downtime duration from node KIENB1052, from track 16B to 17A using the anomaly based algorithm.