PREDICTIVE ANALYSIS OF SOFTWARE PROCESS ANOMALIES – SPC BASED SOLUTION APPROACH

S. M. Didar Al Alam

REPORT NUMBER 110/2014

SOFTWARE ENGINEERING DECISION SUPPORT LABORATORY
UNIVERSITY OF CALGARY

January, 2014
# Table of Contents

Abstract ........................................................................................................................................5

1. Introduction ................................................................................................................................5

2. Literature Review ....................................................................................................................8

   3.1 Difficulties in Applying SPC in Software context ............................................................11
      3.1.1 Variation in Software Process .....................................................................................11
      3.1.2 Hard to Obtain Homogeneous Data .............................................................................11
      3.1.3 Allow Rework ...............................................................................................................11
      3.1.4 Re-calculated Control Limits .......................................................................................11
      3.1.5 Detecting Occurred Changes .......................................................................................11
      3.1.6 Interpreting Anomalies and Tuning Sensibility .........................................................11
      3.1.7 Re-active Approach ......................................................................................................11

4. Problem Statement ................................................................................................................12
   4.1 Challenge 1- Determine Process Control Limits ..............................................................12
   4.2 Challenge 2- Detect Anomalies .........................................................................................12
   4.3 Challenge 3- Investigate for Assignable Cause ...............................................................12
   4.4 Challenge 4- Adjusting Process Limits ...........................................................................12
   4.5 Proposed Challenges .........................................................................................................13
      4.5.1 Challenge 5- Predict Software Process Behaviour Over Time ..................................13
      4.5.2 Challenge 6- Anomalies Detection Using Predicted Behaviour ..................................13
      4.5.3 Challenge 7- Interpreting Detected Anomalies ..........................................................13
   4.6 Research Questions ...........................................................................................................13

5. Proposed Solution Approach ..............................................................................................14
   5.1 Process Characterization .................................................................................................17
      5.1.1 Identify Measurement Objective ................................................................................17
      5.1.2 Selection of Control Charts .......................................................................................17
      5.1.3 Identify the Reference Set ..........................................................................................18
      5.1.4 Determine Control Limits ..........................................................................................19
   5.2 Predictively Analyze Future Behaviour ..........................................................................20
      5.2.1 Identifying Uncertainty Factors and Response Factors ............................................20
      5.2.2 Define Uncertainty Ranges and Distributions ..........................................................21
      5.2.3 Conducting Process Simulation .................................................................................21
5.3 Anomalies Detection .................................................................................................................. 22
  5.3.1 Sigma Tests .......................................................................................................................... 23
  5.3.2 Limit Tests ........................................................................................................................... 23
  5.3.3 Trend Tests ........................................................................................................................... 23
5.4 Interpret Anomalies and Causes Investigation: ........................................................................ 24
  5.4.1 Sigma Test (RT1, RT2 & RT3) Failure .................................................................................. 24
  5.4.2 Run Above/Below CL Test (RT4) Failure ............................................................................ 24
  5.4.3 Mixing / Over control test (RT5) Failure ............................................................................. 24
  5.4.4 Stratification test (RT6) Failure ........................................................................................... 26
  5.4.5 Oscillatory Trend Test (RT7) Failure .................................................................................... 26
  5.4.6 Linear Trend Test (RT8) Failure ........................................................................................... 26
5.5 Tuning Control Limits .................................................................................................................. 26
  5.5.1 Occasional Changes ............................................................................................................. 26
  5.5.2 Occurred Changes ................................................................................................................. 27
  5.5.3 Ongoing Changes .................................................................................................................. 27

6. Case Study ................................................................................................................................. 27
  6.1 Data Source ............................................................................................................................. 27
  6.2 Available Data in GitHub Repository ....................................................................................... 28
    6.2.1 Commit Related Information .............................................................................................. 28
    6.2.2 Issue Related Information .................................................................................................. 28
    6.2.3 Pull Request Related Information ....................................................................................... 28
    6.2.4 Other Information .............................................................................................................. 28
  6.3 Details Description of Scenario 1 (Illustrative Example) .......................................................... 29
    6.3.1 Data for Scenario 1 .............................................................................................................. 29
    6.3.2 Data Collection Process ..................................................................................................... 30
    6.3.3 Data Collection Tools ....................................................................................................... 31
    6.3.4 Process Characterization .................................................................................................... 32
    6.3.5 Selection of Control Chart ................................................................................................. 32
    6.3.6 Identifying Reference Set and Control Limits .................................................................... 32
    6.3.7 Identifying Uncertainty Factors and Response Factors ..................................................... 33
    6.3.8 Defining Uncertainty Ranges and Distributions ................................................................. 33
    6.3.9 Conducting Process Simulation ......................................................................................... 34
    6.3.10 Anomalies Detection and Interpretation .......................................................................... 35
  6.4 Scenario 2 (Illustrative Example) ............................................................................................ 37
    6.4.1 Identifying Reference Set and Control Limits .................................................................... 37
    6.4.2 Determining Uncertainty Ranges and Distribution ............................................................. 38
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.3</td>
<td>Conducting Process Simulation</td>
<td>38</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Anomalies Detection and Interpretation</td>
<td>38</td>
</tr>
<tr>
<td>6.5</td>
<td>Scenario 3</td>
<td>40</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Result Analysis</td>
<td>41</td>
</tr>
<tr>
<td>6.6</td>
<td>Scenario 4</td>
<td>42</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Result Analysis</td>
<td>43</td>
</tr>
<tr>
<td>6.7</td>
<td>Scenario 5</td>
<td>44</td>
</tr>
<tr>
<td>6.7.1</td>
<td>Result Analysis</td>
<td>45</td>
</tr>
<tr>
<td>6.8</td>
<td>Scenario 6</td>
<td>46</td>
</tr>
<tr>
<td>6.8.1</td>
<td>Scenario Data</td>
<td>46</td>
</tr>
<tr>
<td>6.8.2</td>
<td>Identifying Reference Set and Control Limits</td>
<td>47</td>
</tr>
<tr>
<td>6.8.3</td>
<td>Defining Uncertainty Ranges, Distributions and Conducting Simulation</td>
<td>47</td>
</tr>
<tr>
<td>6.9</td>
<td>Scenario 7</td>
<td>49</td>
</tr>
<tr>
<td>6.9.1</td>
<td>Result Analysis</td>
<td>50</td>
</tr>
<tr>
<td>7</td>
<td>Threats to Validity</td>
<td>51</td>
</tr>
<tr>
<td>8</td>
<td>Discussion and Conclusion</td>
<td>52</td>
</tr>
<tr>
<td>8.1</td>
<td>Discussion</td>
<td>52</td>
</tr>
<tr>
<td>8.1.1</td>
<td>Discussion of Results in Respect of RQ1</td>
<td>53</td>
</tr>
<tr>
<td>8.1.2</td>
<td>Discussion of Results in Respect of RQ2</td>
<td>53</td>
</tr>
<tr>
<td>8.1.3</td>
<td>Discussion of Results in Respect of RQ3</td>
<td>53</td>
</tr>
<tr>
<td>8.2</td>
<td>Conclusion</td>
<td>54</td>
</tr>
<tr>
<td>9</td>
<td>References</td>
<td>55</td>
</tr>
</tbody>
</table>
ABSTRACT

Context: Software development organizations require quantitatively managed process to achieve higher levels in software quality standards. Statistical Process Control (SPC) can manage process quantitatively and ensure process predictability in software development. SPC can detect process anomalies reactively (i.e. after anomalies occurred in the process). Due to high variability and dynamic nature of software, reactive detection and solution of process anomalies can be costly and risky. This research attempts to enhance SPC capabilities by introducing predictive analysis of software process anomalies.

Objective: Objective of this research is to predictively detect future process anomalies and take corrective actions to prevent anomalies or minimize their influence on process performance. This research investigates three challenges to achieve this objective. These challenges include: i) Predicting behaviour over time for selected software process characteristics, ii) Predictive detection of anomalies using predicted process behaviour, iii) Interpreting detected anomalies using both observed and predicted process behaviour.

Method: A work scheduling simulation is applied to predict future behaviour of the selected process characteristic (i.e. task completion rate). The simulation approach combines work scheduling method (to model the software development process) and Monte Carlo Simulation (to model the uncertainty in the software development process). Systematically varied input values for selected uncertainty factors (i.e. task size, task arrival, and developer’s productivity) allows to predict behaviour of the process characteristic in varying process circumstances. Selected “Run Rules” are applied to detect process anomalies within observed and predicted behaviour of the process. A set of guidelines are proposed to interpret detected anomalies in consideration of both observed and predicted behaviour of the process. Proposed solution approach is presented as an extension of earlier approach “Monitoring problem-SPC Based Solution”. Predictive detection of process anomalies allows to initiate early investigation to prevent an anomaly or minimize its influence on process performance. Initial validation of the proposed approach is done by using two mid-size Open Source Software (OSS) development project (i.e. “octoki.rb” and “NuGetDocs”) available online in GitHub.

Results: In lack of past knowledge and legacy data, future behaviour of software process characteristics can be predicted using the observed behaviour of the process. Run Rules are capable of detecting process anomalies in respect of both observed and predicted behaviour of the process. However, specific patterns in anomaly detection (i.e. limit and trend tests) are considered significant for interpretation. Interpretation of detected anomalies depends on both predicted and observed anomalies. This combined interpretation allows to detect anomalies or to understand their influence on software process early. This also facilitates initiating early corrective actions.

Conclusion: The added benefit of the presented approach is that process-monitoring techniques are able to predictively detect process anomalies and initiate early corrective actions. Initial validation supports key contributions of this research. They are: i) Predictive determination of future behaviour for selected process characteristic, ii) Predictive detection of future process anomalies using predicted behaviour of the software process, iii) Combined interpretation of detected anomalies based on observed and predicted anomalies in the software process, iv) Early corrective actions to prevent an anomaly or minimize its influence on future process performance.

1. INTRODUCTION

Context: Software development process develops software as a desired output from available inputs utilizing a combination of man, machine and systematic methods [1],[2]. Quality of the software product relies on the quality of the process that develops them. Therefore, in past decade software community highlighted process management and continual process improvement as a way to improve the quality of software products [3]. A culture of measuring software process and monitoring consistency, effectiveness and efficiency of the software process has developed. Measurement based approaches can be categorized in two major categories: i) focus on finding what improvements are required for the software process e.g. QIP/GQM [4], ii) focus on finding when process improvement is required. The second one is achieved through monitoring process performance and verifying effectiveness of induced improvements e.g. the time series analyses [5]. Statistical Process Control (SPC) [1],[6] is a well-established time series analysis technique, popular in manufacturing context. SPC recently gained interest in software community as well. However, due to process diversity [7] (i.e. variation among processes in different organizations, projects or project executions) application of SPC in software context impose several challenges [8].

Motivation: Internationally recognized software quality standards require an organization to have statistically managed process in order to achieve higher levels of quality. In Capability Maturity Model Integration (CMMI),
maturity level 4 or higher is only achievable if the software process is quantitatively managed and optimized [9]. Therefore, SPC gained significant importance in software development context as a technique to achieve quantitatively managed process and ensure certain level of process predictability. A growing number of organizations reported successful application of SPC in software context [3],[2],[10],[30],[11]. However, applying SPC in software organizations have proved difficult. SPC was developed for manufacturing context and does not consider specific characteristics of software processes. Only 7% of the software organizations in the world were assessed to achieve maturity levels 4 or higher. Experience of using SPC in software context is not yet mature. We need better understanding of SPC outcomes in software context. We also need to customize SPC in consideration of the differences between Manufacturing and Software development process.

Problem Statement: Due to unique characteristics of software development compared to manufacturing process, process-monitoring in software development impose unique challenges. Four key challenges of using SPC in software are well known and well studied in literature. They include: i) characterizing software process in terms of quantifiable process characteristics and process limits, ii) detecting anomalies in a software process, iii) investigating assignable causes responsible for detected anomalies, iv) adjusting process limits towards desired process behaviour changes. SPC can detect process anomalies reactively (i.e. detection after the anomaly occurred in the process). Software processes are dynamic and significantly vary over time. Reactive detection and solution of anomalies can be costly and risky for dynamic processes like software development.

In this research, our objective is to enhance the capability of process monitoring technique i.e. Statistical Process Control (SPC). We want SPC to be capable of predicting future anomalies and proactively initiating appropriate corrective actions to prevent the anomaly or minimize its influence on process performance. To accomplish this objective we enhanced our understanding of the key challenges for applying SPC in software context. We proposed three new challenges related to our research objective as listed below. Three research questions investigated in this report are in line with these proposed challenges as well.

Predicting Software Process Behaviour Over Time: To detect and analyze anomalies in process predictively, we need to predict future behaviour for selected process characteristics. We can accomplish this using appropriate prediction models. Due to process diversity, unavailability of legacy data and past knowledge predicting future behaviour of a process characteristic becomes a key challenge.

Anomalies Detection Using Predicted Process Behaviour: Due to uncertainties and high variation in software process, predicted behaviour has lower trustworthiness compared to observed behaviour of the process. It is difficult to rely on single observation value within predicted behaviour of the process. Therefore, specific patterns in observation values are of higher importance than single observations falling out of the control limit. We need to determine applicable rules that can detect anomalies using predicted behaviour of the process.

Interpreting Detected Anomalies: Anomalies detected within predicted behaviour of a process may have different interpretation compared to anomalies detected in observed behaviour of the process. We need to create proper guidelines to interpret these anomalies. We also need to analyze how interpretation of predicted anomalies is influenced by observed anomalies and vice versa.

Proposed Solution Approach: Before we can address three proposed challenges and our research questions, we need a solution for four key challenges of applying SPC in software context. Different authors attempted to solve one or several of these challenges in different ways. Instead of investigating four key challenges again, we built our solution approach on an existing solution approach. Baldassarre et al. in [12], [13] addressed all four key challenges with “Monitoring problem-SPC based solution” approach (i.e. SPC based solution approach). We consider our proposed solution as an extension to this approach. Major steps of our proposed solution approach include-

Process Characterization: To monitor and control a software process over time, the process needs to be characterized first. We need to determine the measurement objective, related process characteristics and process measures. Based on monitoring requirements, we should select an appropriate control chart. We need to find a stable set of observation referred as Reference Set. Once the Reference Set is ready, we can calculate the control limits and start monitoring the process on control chart.

Predictively Analyze Future Behaviour: Predictively analyzing future behavior is a key step in our solution approach. We predict behavior of our selected process characteristic (i.e. task completion rate) using a work schedule simulation. This is a combination of work scheduling method and Monte-Carlo simulation. While work-scheduling method models the development process and schedules available tasks, Monte Carlo simulation models uncertainty of the process. Systematically varying values for uncertainty factors (i.e. task size, task arrival, and developer’s productivity) help to develop a probable behavior pattern (i.e. distribution) for the response factor (i.e. task
completion rate) in different circumstances. This behavior pattern represents future behavior of the process characteristic.

**Detecting Anomalies:** Predicted and observed data points (behaviour) are plotted on the same control chart. These data points are plotted against the control limits in order to find anomalies and instabilities in the process. We utilize run tests [14]–[16] for anomaly detection in respect of both predicted and observed data points. Due to variability in software process, systematic patterns in control chart (i.e. limit and trend tests) are considered significant compared to single point exceptions (i.e. sigma tests).

**Interpreting Anomalies and Causes Investigation:** Isolated interpretations for detected anomalies are no longer considered. A set of guidelines are proposed to interpret anomalies, considering the influence of observed anomalies on predicted anomalies and vice versa. These guidelines help to early detect anomalies or their pattern and influence on software process. They help to interpret detected anomalies and initiate action to prevent the anomaly or minimize its influence on the process.

**Tuning Control Limits:** Software process changes significantly. To accept desirable changes in a software process we need to tune the control limits. Tuning control limit is dependent on observed values as control limits are only tuned when process shift has already occurred in the process. Predicted values help to early detect a process shifts and initiate early investigation to prevent undesirable process shifts. Predicted values does not have any direct impact of tuning control limits.

**Contribution:** Our proposed solution approach is an extension to the earlier “Monitoring problem-SPC based solution” approach. We introduce predictive anomaly detection for software processes. Proposed approach is capable of predicting future behaviour of a process, detect and interpret future anomalies using it. Therefore, it can initiate early actions to prevent a future anomaly or minimize its influence on the process performance. Our contributions in this research are:

1. We predictively determined future behaviour of selected process characteristics using observed data of the process. Instead of creating the burden of collecting legacy project data or past knowledge, we use the observed data of the process.
2. We predicted future anomalies using the predicted behaviour of selected process characteristics.
3. We developed guidelines for combined interpretation of detected anomalies in respect of both observed and predicted behaviour of the process.
4. We facilitate early cause investigation of process anomalies to prevent a future anomaly or minimize its impact on process performance.

**Validation:** We attempted to validate our proposed approach using open source software (OSS) projects form Github repository [17]. Github is a web-based hosting service for OSS development projects that use Git as a revision control mechanism. Github is an open source program for tracking changes in files. We performed our case study on two midsize projects “octokit.rb” and “NuGetDocs”. Our selected process characteristic “contribution rate” (i.e. task completion rate) was monitored using proposed approach. Predictive work-scheduling simulation was performed in consideration of three uncertainty factors i.e. task arrival, task size, developers skill level. Distribution and ranges for uncertainty factors were calculated using Experience Database. Available probability functions were fit into available empirical data (i.e. observed value of the process) using EasyFitExcel [18] tool. We utilize varying values for uncertainty factors within determined ranges and distribution in the work scheduling simulation. These uncertainty values represent unforeseen future events. Corresponding output values for response factor (i.e. task completion rate) provide a probable behaviour pattern for the response factor. We utilize these predicted behaviours to detect anomalies using Run Rules.

We tested seven scenarios to validate our proposed approach. We analyzed our case study results using four analysis questions targeted to address the research questions. They are –

- Analysis 0: Can we predictively generate future behavior of a selected process characteristic and plot it in a XmR control chart along with observed behavior of the process characteristic?
- Analysis 1: Can we use run tests to detect and interpret predicted anomalies using predicted behavior of the process characteristic?
- Analysis 2: Can we interpret detected anomalies in respect of both predicted and observed anomalies?
- Analysis 3: Can we validate predicted anomalies against real world anomalies?
Primarily study results positively support all contributions of this study. Study results show that, we can predict future behaviour of selected software process characteristics using observed values. Even in lack of past knowledge or legacy data, observed values are enough to continue anomaly prediction. However, the quality of prediction is dependent on the prediction method used, and the quality of data. Trend and Limit test (specific patterns in control chart) are effective in detecting anomalies in both observed and predicted values. Due to high variability of software process and low trustworthiness of predicted data, sigma tests are not considered as significant and do not initiate any corrective action. Combined interpretation of detected anomalies helps to detect the anomalies or their pattern and influence on software process early. It provides more information that helps deciding corrective actions compared to isolated interpretations. However, due to data unavailability and limited scope this case study results suffers from assumptions and validity threats. Therefore, we consider a more comprehensive validation of the proposed approach utilizing a better predictive simulation is still outstanding.

This report is organized in eight sections. Literature review is presented in Section 2. Key challenges of utilizing SPC in software context are discussed in Section 3. Section 4 illustrates our problem statement. Section 5 demonstrates the solution approach. Section 6 presents our case study. Threats to validity of our study are listed in Section 7. Finally, Section 8 provides discussion of study results and conclusion of the project.

2. LITERATURE REVIEW

Software processes are human intensive process and characterized by human intensive factors like developer’s experience, productivity, knowledge etc. They are significantly diversified among organizations, projects, culture even different executions of same project. This phenomena is formally called “Process Diversity” [19], [20]. Due to this, software process monitoring, management and improvement is not a trivial task [12]. Concepts and methods related to process management and continual improvement received increasing importance as a way to improve software product quality [3]. This approach is known as “process thinking”. Due to increasing popularity of this approach, practitioners became motivated to measure software processes in respect of consistency, effectiveness, and efficiency of the process. Existing measurement based approaches focus on either of two basic questions, i) what process improvements are required e.g. QIP/GQM [4] and ii) when process improvement is required e.g. time series analyses. Statistical Process Control (SPC) [1], [6] is a well-established time series analysis technique popular in manufacturing context.

SPC is a statistical based monitoring approach. It was developed in Shewhart [1], [6] to monitor manufacturing process. It became successful and popular in manufacturing context. Recently it gained popularity in software process monitoring. SPC uses control charts along with established control limits (i.e. permissible limits for process performance variation) to determine stability of a software process. Software stability is dependent on the presence of common cause and assignable cause variations. Common cause variations are the result of normal interactions of people, machines, environment, techniques etc.Assignable cause variations are generated from events that are not part of the process and make the process unstable. If a process performance is affected by common causes only it is considered as “stable” or “under control”.

Software processes significantly differs from manufacturing processes [21]. Some of these differences are listed in [5] and summarized in Table 1 below. Predominance of human factor is the key reason for variations in software process performance. Human activities are asymmetric and non-deterministic. Thus each software process execution becomes creative and unique [22]. Software process are complex to predict, monitor and improve [19]. Additionally, other difficulties also exist regarding software process control using SPC. For example, collecting metrics, selection and reliability of process characteristics [23], violation of SPC assumptions [24] etc. Curtis et al. [25], [26] analyzed differences between software professionals. Surprisingly, they ranged from 10:1 up to 100:1 and 20:1 differences are common. This finding questioned the validity and business value of utilizing SPC in software context [27]–[29].

REPORT NUMBER 000/0000

PAGE 8 OF 56
SPC must be addressing three more challenges.

To achieve Capability Maturity Model Integration (CMMI) maturity level 4 or higher, software process should be quantitatively managed and optimized [9]. SPC is considered as a technique to achieve quantitatively managed process and ensure certain level of process predictability. Therefore, SPC recently gained significant importance in software development context. However, SPC must be tailored properly to consider unique characteristics of software process.

Authors extended the challenges of using SPC in software context. For example, study process performance, find anomalies and react before it gets too late are some of the benefits of SPC. While some of the limitations are, it is difficult to work with small datasets, it is hard to achieve legacy data or past knowledge for using SPC in software context.

Recent publications focused on the applicability of SPC in software context and started customizing SPC. [22] proposed a continuous software process improvement mechanism using SPC. Later this idea was extended to more sophisticated approaches presented in [12]. Authors extended the challenges of using SPC in software context. They consider four key challenges: i) Baseline definition, ii) Anomalies Detection iii) Cause investigation and iv) Tuning sensibility. How SPC can be used as a decision support tool in software context is presented in [5]. Authors proposed a “Monitoring problem-SPC based solution” approach as solution for the challenges. They validated the proposed approach using industrial data. In our proposal, we extend these four challenges by including three more challenges. Proposed challenges are related to predictive detection of anomalies in software process using SPC. Our approach is an extension to the “Monitoring problem-SPC based solution” approach. It includes predictive anomaly detection capabilities into the earlier approach. More details regarding this model and its usage are presented in later sections.

SPC can quickly highlight shift in process performance. Based on this property [8] considered SPC is capable of overcoming dynamic calibration weakness. Authors proposed a dynamic calibration method for estimation models in. In [8] they introduced SPC in this model to detect when calibration is required. They also developed a tool support for this proposed method named SPEED (Software Project Effort Estimator using Dynamic calibration) [41].

Some major drawbacks of using univariate SPC models are presented in [42]. Typically, software processes are monitored in respect of individual variables. Only a few variables can be monitored at a time. The co-relation between these variables is not considered. This might lead to inappropriate corrective actions [43]. This report proposes the use of SPC for detecting assignable causes using multivariate SPC. The main advantage of this approach was correlated variable can be properly monitored as well.

Table 1 Software process vs Manufacturing process

<table>
<thead>
<tr>
<th>Software Process</th>
<th>Manufacturing process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human intensive process, Dependent on cognitive activity</td>
<td>Machine intensive process</td>
</tr>
<tr>
<td>Each process execution have different input and output</td>
<td>Each process execution have a same set of input and output</td>
</tr>
<tr>
<td>High variation in process performance</td>
<td>Low variation in process performance</td>
</tr>
<tr>
<td>Risk present in all phases</td>
<td>Risk is concentrated in the design phase</td>
</tr>
<tr>
<td>Requirement specifications are difficult to obtain.</td>
<td>Requirement specifications are comparatively easier to obtain.</td>
</tr>
</tbody>
</table>

Internationally recognized software quality standards allow organizations to achieve higher levels of quality if they have statistically managed process. For example, to achieve Capability Maturity Model Integration (CMMI) maturity level 4 or higher, software process should be quantitatively managed and optimized [9]. SPC is considered as a technique to achieve quantitatively managed process and ensure certain level of process predictability. Therefore, SPC recently gained significant importance in software development context. However, SPC must be tailored properly to consider unique characteristics of software process.
3. STATISTICAL PROCESS CONTROL IN SOFTWARE CONTEXT

Statistical Process Control (SPC) is a technique for time series analysis, first developed in 1920 by Shewhart. SPC uses control charts to monitor and evaluate process performance over time. Control charts establish operational limits for acceptable process variation. Two major types of variation considered are i) common cause variations, and ii) assignable cause variations. Common cause variations are process performance variation created due to common causes like normal interactions of people, machines, environment, techniques used etc. In presence of this variation, the process is stable and demonstrates a predictable behaviour within a certain error range. On the other hand, assignable cause variations are those arise from events external to the process. These types of variations make the process unstable and behave unpredictably.

Major process performance indicators are Central tendency (CL), Upper Control Limit (UCL = CL+3σ) and Lower Control Limit (LCL = CL-3σ). SPC can determine central tendency, upper and lower control limits for process performance variation using few data points. Statisticians calculate Sigma (σ) using a set of factors. Details regarding sigma (σ) calculation are available in [44]. Control chart calculates the process performance indicators and continuously monitor process performance over time. One/more values outside the control limits or non-random behaviours indicate presence of an assignable cause. Further cause investigation is required to detect the assignable cause and take preventive actions. We can summarize use of control chart in four key steps. They are: i) Collect samples from process, ii) calculate statistics (i.e. UCL, LCL and CL), iii) plot samples in presence of calculated statistics, iv) interpret results in respect of the control limits (i.e. if one/more data points are beyond control limits or other unusual pattern indicate assignable cause of variation.)

Figure 1 presents a normal distribution (i.e. the Bell curve) along with percentage of observations fall in each corresponding area. In this distribution, μ is the theoretical mean and σ is the theoretical standard deviation. Upper Control Limit (UCL) and Lower Control Limit (LCL) are calculated correspondingly as (UCL=μ+3 σ), and (LCL=μ-3 σ). 99.73% of total observations fall within this calculated limit. In a stable process only 0.27% observations are allowed to fall outside this control limit. However, normal distribution is a good theoretical model but not an exact representation of different distributions encountered in real world scenarios. Three rule of thumb that works independent of data distribution are:

- 60% – 75% observations fall within ±1 sigma
- 90% – 98% observations fall within ±2 sigma
- 99% – 100% observations fall within ±3 sigma (i.e. UCL and LCL)

![Normal Distribution, the Bell Curve](image)
3.1 DIFFICULTIES IN APPLYING SPC IN SOFTWARE CONTEXT

Initially, SPC was created to statistically monitor and control manufacturing process. Applying SPC in software development process is different from applying SPC in manufacturing process. In literature authors reported successful attempts and outcomes of utilizing SPC in software process. However, a clear understanding of SPC, control charts and its application in context of software is still lacking [5]. Authors in [45], [46] presented differences between manufacturing and software processes. These differences influence the application of SPC in software context. Some major challenges of applying SPC in software context are briefly discussed below.

### 3.1.1 VARIATION IN SOFTWARE PROCESS

Software process demonstrates significant variation in process measurement compared to manufacturing process. While manufacturing is a machine intensive process, software process is a blend of Man, Machine and Systematic methods. Software process has predominance of cognitive activities. Human performance influences software process performance. Input and output of a software process varies between each execution of the process. Moreover, introducing an innovation introduce destabilization in software process. These are some key reason for extreme variations observed in software process. Thus to monitor stability of a software process we require different indicators and interpretations compared to manufacturing process.

### 3.1.2 HARD TO OBTAIN HOMOGENEOUS DATA

In manufacturing process, similar products are produced daily in mass number. It is easy to collect large set of homogeneous data that can perform as past knowledge for SPC. However, software processes usually result in a relatively small set of work products [11]. Different execution of a software process may have different input and outputs. Each process execution is a creative and unique activity and thus changes the output products as well. These effects are not typically visible in manufacturing process. Thus, it is hard to obtain large set of homogeneous data in SPC compared to manufacturing process.

### 3.1.3 ALLOW REWORK

If an anomaly is detected in manufacturing process, the assignable cause is removed to bring the process back on track. Such actions of manufacturing product control is evident in [31],[47]. Software processes allow to rework a product, if it presents an anomaly. Thus, the demand from software project managers is different. They prefer to receive potential warning signals (even if it is false) from control chart rather than missing them. Thus, SPC need to be customized to address the needs of software project managers.

### 3.1.4 RE-CALculated CONTROL LIMITS

When an anomaly is detected, software processes can rework on the product to fix the anomaly. It does not require explicit shutdown and startup activities like manufacturing process. In software context re-calculating control limits is cheap and easier compared to the manufacturing process. Thus SPC, control chart, indicators and tuning process need to be adjusted to reflect these characteristics of software process while working in software context.

### 3.1.5 DETECTING OCCURRED CHANGES

Software projects show high variability due to Human performance influences on software process (Card, 1994)(fill gap). Thus, detecting occurred changes are important compared to detecting occasional variations. Manufacturing processes focus on individual events and removes the assignable causes behind any detected event. However, in software context Control charts should detect process trends instead of individual nonconforming events. SPC indicators should be capable of indicating assignable variation and difference between occasional changes and occurred variations. A different set of indicators and interpretations are required to apply SPC in software context.

### 3.1.6 INTERPRETING ANOMALIES AND TUNING SENSIBILITY

SPC only detect anomalies present in a process. It does not provide indication towards cause investigation of the anomaly, guideline for interpretation of the anomaly or any corrective action. In software context specific patterns of observation values are important compared to single observation falling out of the control limits. Thus to apply SPC in software context different indicators and appropriate interpretation of those new indicators are required.

### 3.1.7 RE-ACTIVE APPROACH

SPC statistically monitors a process and detects the presence of unusual patterns and observations beyond the admissible process variation limits. SPC is a reactive approach that can only detect anomalies after the anomaly...
occurred in the process. SPC does not provide any predictive analysis facility that can early detect possible future anomalies and take corrective actions to prevent the anomaly or minimize its influence on the process. Software process is dynamic and presents a high level of variation over time. Early detection of anomalies and guidelines for corrective actions are important in context of software process.

4. PROBLEM STATEMENT

In software development, process-monitoring techniques monitor the software development process over time to detect unusual pattern or observations beyond the admissible limits of variation. Process monitoring typically performs a series of actions. These actions are:

- Determine control limits for admissible process performance variation,
- Compare observed process performance variation with admissible limits of process variations
- Detect anomalies in the process under monitoring
- Investigate detected anomalies to find the assignable cause
- Perform reactive actions to prevent the anomaly or minimize its impact
- Adjust control limits dynamically to reflect changes in process behaviour over time.

As stated earlier, software development presents unique characteristics compared to manufacturing process. Due to these characteristics, process-monitoring in software development impose unique challenges to perform the above listed actions. Four key challenges are as follows:

4.1 CHALLENGE 1- DETERMINE PROCESS CONTROL LIMITS

We need to characterize the process in terms of quantifiable process characteristics before monitoring it. We need to determine control limits (i.e. threshold values) for admissible process performance variation of selected process characteristics. Legacy data or past knowledge is required to determine these control limits. Process monitoring technique measures selected process characteristics over time and compares with admissible control limits for process variation. Software development projects are unique, innovative and significantly different from past projects. Thus, required legacy data or past knowledge on the process under monitoring are rarely achievable. Unavailability of past knowledge of the process imposes major challenge while determining the process limits.

4.2 CHALLENGE 2-DETECT ANOMALIES

Process-monitoring techniques compare observed variations in process performance with admissible limits of process variations. Anomalies are detected if observations fall beyond these admissible limits. Unlike manufacturing process, single observation falling outside the limits are of less importance in software development. Instead, systematic pattern of observations are helpful to detect anomalies or process shifts over time. Thus, we need to determine new rules and indicators to detect such anomalies in context of software development.

4.3 CHALLENGE 3- INVESTIGATE FOR ASSIGNABLE CAUSE

Process-monitoring techniques detect anomalies in a software process. However, they are not responsible to provide guidelines that facilitate investigation of the assignable cause, responsible for the anomaly. Proper guidelines for investigation of the assignable cause, corrective actions to prevent the assignable cause or minimize its impact on the process are required. Next key challenge is to provide guidelines that facilitate investigation of the assignable cause.

4.4 CHALLENGE 4-ADJUSTING PROCESS LIMITS

Software process performance varies over time dynamically due to organization culture, type of project or experience level of employee etc. This variation is called Process Diversity [7]. We need to consider Process diversity in our process-monitoring technique and adjust the control limits accordingly. Otherwise, process limits will become too broad or too narrow and will impose a risk of false alarms or missing alarms. We should adjust process limits meaningfully towards desired process behaviour changes. Performing such adjustments dynamically over time impose a challenge for the monitoring process.
4.5 PROPOSED CHALLENGES

Above-mentioned four key challenges are well known and well studied in literature. Different authors attempted to solve one or several of these challenges in different ways. Baldassarre et al. in recent papers [5], [12], [13] addressed all four key challenges with “Monitoring problem-SPC based solution” approach. SPC is a process-monitoring technique capable of detecting anomalies in software process. However, SPC can reactively detect anomalies in a software process i.e. detection of an anomaly after it occurred in the process and influenced the process performance. It can initiate investigation to remove the assignable cause behind this anomaly. Software processes are dynamic and significantly vary over time. Reactively detect and solve anomalies in a highly dynamic process like software development can be costly and risky.

In this research, our objective is to enhance the capability of a process monitoring technique (e.g. SPC) by introducing predictive analysis of process anomaly. We want to predictively detect software process anomalies using SPC and initiate appropriate action to prevent the anomaly or minimize its influence on the process performance. To accomplish this objective, we enhanced our understanding of the key challenges for applying SPC in software context. We proposed three new challenges related to our objective.

4.5.1 CHALLENGE 5-PREDICT SOFTWARE PROCESS BEHAVIOUR OVER TIME

To detect and analyze software process anomalies predictively, we need to predict future behaviour for selected process characteristics. We can accomplish this by using appropriate prediction models. We need to identify uncertainty factors responsible for changes in the response factor (i.e. selected process characteristics). By systematically varying uncertainty factor values within given range and distribution, we can represent unforeseen events that might occur in the future process. These will help us to predict future behaviour of the response factor in different circumstances. Due to process diversity, unavailability of legacy data and lack of past knowledge, predicting future behaviour of process characteristics using predictive models becomes a challenge.

4.5.2 CHALLENGE 6-ANOMALIES DETECTION USING PREDICTED BEHAVIOUR

Due to high variability of software process, specific patterns in multiple observations are of higher importance than single observations falling out of the control limit. Predicted values for software process characteristics have lower trustworthiness compared to observed values. Therefore, it is risky to initiate any action based on single observation failures within predicted behaviour. We need to find specific patterns of observations in order to consider any action. It is a challenge to determine which anomaly detection rules should be used and how they should be interpreted in respect of predicted values.

4.5.3 CHALLENGE 7-INTERPRETING DETECTED ANOMALIES

Anomalies detected within predicted data (i.e. predicted anomaly) may have different interpretation compared to anomalies detected in observed data (i.e. observed anomaly) of a process. Observed anomalies may influence interpretation of predicted anomalies and vice versa. Therefore, we need to develop a complete guideline that presents, how to interpret detected anomalies in consideration of observed anomalies influence on predicted anomalies and vice versa. Developing guidelines for such interpretation of detected anomalies is a challenge.

4.6 RESEARCH QUESTIONS

If we can find a solution for these three proposed challenges, monitoring process like SPC will be able to predictively detect anomalies in software process. This will also facilitate early investigation of an anomaly to prevent it from occurring or to minimize its influence on process performance. Due to uncertainties and high variability of software process, this approach increases the risk of false alarms in anomaly detection. However, authors in [11] analyzed project managers experience and states that, project managers prefer to receive false alarms instead of missing an alarm and detect faults at a later stage. In order to address these three proposed challenges we studied three research questions (RQ) in this report. We formulated our research questions using a goal oriented template [48]. Details of our research questions are presented below-
Research Question 1

| Analyze (Object of Interest) | selected process characteristic of a software process, which is monitored using statistical process control (SPC) technique in XmR control chart |
| In order to (Purpose) | predict future behaviour over time for the selected process characteristic |
| With respect to (Focus) | available observations of the software process |
| From the point of view of (Perspective) | Project Manager |
| For the environment (Context) | of an operational development process of a software project. |

Research Question 2

| Analyze (Object of Interest) | predicted behaviour for selected process characteristic (i.e. selected in RQ1) of a software process |
| In order to (Purpose) | understand the success of predicting anomalies |
| With respect to (Focus) | tests available for anomaly detection within observed value of a software process |
| From the point of view of (Perspective) | Project manager |
| For the environment (Context) | of an operational development process of a software project. |

Research Question 3

| Analyze (Object of Interest) | anomalies detected within predicted behaviour of selected process characteristic (selected in RQ1 & RQ2) |
| In order to (Purpose) | understand the influence on interpretation of anomalies |
| With respect to (Focus) | anomalies and process shifts detected within observed behaviour of the software process |
| From the point of view of (Perspective) | Project manager |
| For the environment (Context) | of an operational development process of a software project. |

Three proposed research questions address three proposed challenges of this research. Solution of this research questions can guide us towards three key contributions presented earlier. RQ1 investigates, whether or not it is possible to predict behaviour of a software process characteristic, utilizing available observed data of the process. This question addresses proposed challenge 5. The second research question RQ2 analyzes predicted behaviour of the selected process characteristic. It attempts to find a set of tests that can detect anomalies utilizing predicted behaviour. RQ2 addresses proposed challenge 6. The third research question RQ3 focuses on the interpretation of detected anomalies. It attempts to understand how interpretation of the predicted anomalies is influenced by observed anomalies and vice versa. RQ3 addresses proposed challenge 7 and develops a guideline for combined interpretation (i.e. considering both observed and predicted anomalies) of detected anomalies. In next section, we present our proposed solution approach that addresses all three research questions presented here.

5. PROPOSED SOLUTION APPROACH

Proposed solution approach attempts to find a solution for three research questions and corresponding challenges. However, proposed challenges are an extension to the four key challenges of applying SPC in software context. Before we can address proposed challenges, we require a solution for these four key challenges first. These four challenges are well known and well studied in literature. Instead of investigating solution for them, we select an available solution approach as our starting point. We extend this approach to address our proposed challenges.
“Monitoring problem – SPC based solution” approach [12], [13] addresses four key challenges. We introduce predictive anomalies detection into this approach. It makes SPC capable of predictively detecting future anomalies and initiating actions to prevent anomalies or minimize their influence on process performance. We propose three extensions to the current approach. How proposed extensions fit with the current approach is illustrated in Figure 2. First, we introduce a new branch that deals with the problem “predicting behaviour” of the software process. The solution is achieved through a predictive simulation approach. We add one additional sub-branch to the anomalies detection branch. Here, we focus on anomalies detection utilizing predicted values of the software process. The solution approach attempts to detect anomalies in both observed and predicted values. We add another sub-branch under the cause investigation branch. Here, we focus on interpreting detected anomalies in consideration of both observed and predicted anomalies. We provide guidelines for combined interpretations of detected anomalies in our solution approach. Details of our proposed solution approach are presented in this section.

![Figure 2 Proposed solution approach](image)

Figure 3 below presents step-by-step actions proposed in our solution approach. All these steps are discussed in five sub-groups (Figure 2). They are i) Process characterization, 2) Predicting future behaviour for selected process characteristics, 3) Anomalies detection, 4) Assignable causes Investigation and 5) Tuning control limits. Each of these sub-groups and corresponding action steps are discussed in this section. First, we briefly inform how proposed approach works and how sub-groups relate with each other below.

To monitor and control a software process, first we need to characterize the software process. This is accomplished through process characterization discussed in Section 5.1. We identify the measurement objective, select process characteristics and related measures for this objective. We need to select an appropriate control chart based on the selected process characteristics. We identify an initial reference set and test if the reference set is stable. If the reference set is unstable, we need to remove all assignable causes and test for stability of the reference set again. After the stable reference set is determined, we can calculate our control limits. We can utilize these determined control limits to create our control chart and test observations.

If a new observation is available, we will use this new observation in two different actions in parallel. On one side, we verify and collect the observation of the process characteristic and plot it on the control chart. On the other side, based on all observed values (including the new observation), we conduct a predictive simulation to predict future values for selected process characteristics. In order to accomplish this predictive simulation we follow three major steps i) Identifying uncertainty factor and response factors, ii) Define uncertainty ranges and distributions, iii) Conducting process simulation. Details regarding all these simulation steps are discussed in Section 5.2. Objective of our research is to predict future anomalies using monitoring process e.g. SPC. To achieve this objective we introduced three key challenges (presented in section 4.5). Predictive value generation for selected process characteristics is a part of our solution approach that addresses challenge 1 and corresponding research question RQ1. This contribution is considered as an extension to the “Monitoring problem-SPC based solution” approach.

Next, we initiate the anomaly detection process within both observed and predicted data. We utilize eight statistical test for process monitoring known as “Run test” for this purpose. This approach addresses challenge 2 and corresponding RQ2. Details of this approach are presented in 5.3. If no anomalies are detected, we consider the data set is stable and under control. Therefore, we concentrate on collecting next observation. If an anomaly is detected,
Figure 3 Flow chart for predictively detecting anomalies in software process using SPC
whether in observed or predicted values, we need to interpret these anomalies. Interpretation will help to decide whether assignable cause investigation is required or not. In order to address interpretation of predicted anomalies, we proposed a set of interpretation rules in Table 5. These rules consider the influence of observed anomalies while interpreting predicted anomalies and vice versa. Table 5 facilitates assignable cause investigation for anomalies detected within observed and predicted values. This contribution addresses challenge 3 and corresponding research question RQ 3. Observed and predicted anomalies should no longer provide any isolated interpretation. Instead, they should provide collaborative interpretation for detected anomalies.

If anomalies are detected within observed values, we remove the assignable cause and start waiting for a new observation. If no assignable cause is found, we investigate if a process shift happened and requires tuning control limits. If process shifts are not found, we start looking for the next observation. However, if a process shift is detected, we redefine a reference set, re-calculate our control limits and start waiting for new observations. Details regarding this approach is available in Section 0. If anomalies are detected within predicted values, we initiate early investigation for these anomalies in order to prevent them or minimize their influence on future process. We remove any assignable cause found and start waiting for new observation. We do not conduct any control limit tuning based on predicted anomalies.

5.1 PROCESS CHARACTERIZATION

To monitor a software process over time, the process under observation need to be characterized. We need to find a set of observations that is stable and has not suffered from any assignable cause of variation. This set of observations represents the normal process performance and named as Reference Set. Once a Reference Set is ready, we can start monitoring our observations on the control chart in order to detect anomalies in the process. Process Characterization is performed in four steps as listed below.

5.1.1 IDENTIFY MEASUREMENT OBJECTIVE

We need to clarify our business goals and determine the process of interest that we want to monitor and control over time. Applying SPC for the entire software development process or complex sub-processes are ineffective. It burdens project and organization with measuring activities [11]. A quote from [3] illustrate the possible threat well.

“Some software organizations quickly grow frustrated with SPC because they start out using it to measure a big process comprising many sub-processes or even the organization’s entire software process.”

Considering our business goals, we must determine the process of interest that we want to evaluate. We need to determine appropriate measurement characteristics that describe performance of the process under observation. For example, in this report we performed our case studies on open source software (OSS) development process. Instead of focusing on the entire software development process, we focused only on contributions on master branch of the repository. The project is available in the OSS repository (e.g. Github[17]). The measurement characteristics we consider to monitor is daily contribution rate of the process. Details regarding these case studies are available in Section 6.

5.1.2 SELECTION OF CONTROL CHARTS

The next step in process characterization is to select an appropriate control chart. Different types of control charts are available for variable and attribute data. We need to select one that is useful in our context. Literature applied different types of charts in past. Due to its sampling based technique, X-bar chart are popular in monitoring manufacturing process. Authors in [13] presented a decision tree for control chart selection. We can use this decision tree (presented in Figure 4) as an initial guideline to select appropriate control charts.
Software process demonstrates some special characteristics compared to manufacturing process. Thus, control chart selection for software process is different from the manufacturing process. Due to scarceness of data in Software process, data points are individually plotted and evaluated. Software processes have high level of variation. Detecting isolated data points out of control limits are not the primary focus of the monitoring approach. Instead, any trend in plotted data points are considered more significant. Considering these characteristics, XmR and U charts are suitable for software processes. Details regarding there suitability for software process are available in [30], [47].

In this report, we selected XmR control chart for applying SPC in software process. In XmR chart, X chart represents single observation values and mR chart represents the moving range. XmR control chart has some predefined assumptions that closely resemble characteristics of software processes. For example, XmR chart consider one observation per period. Observations are measured in an interval scale and can be meaningfully added or divided. Observations are independent of each other and knowledge of one observation does not inform much about other observations. In literature, XmR control chart performed well in past and became popular for applying SPC in software process. Authors in [5], [8], [11], [41] demonstrated some successful use of XmR control chart for SPC in software process.

5.1.3 IDENTIFY THE REFERENCE SET

Determining the reference set is a key step towards applying SPC using control charts. Reference set is a set of observations collected in respect of the measurement characteristics of interest for the process under observation. Reference Set should express normal behaviour of the process considering affects from common causes of variations only. Software projects significantly vary from each other due to organization culture, project type or employee experience i.e. Process Diversity. Due to process diversity, past knowledge is rarely available and legacy data are ineffective to determine the Reference Set. Therefore, we need to observe process performance in respect of the selected measurement characteristics for a certain period. We can establish the Reference Set based on these observations and calculate corresponding control limits. Observed data are plotted on the control chart and tested to detect anomalies. If no assignable cause of variation is found, this observed data set is considered stable and selected as the reference set. Control limits are calculated and applied for test execution of future observations. On the other hand, if we detect anomalies in this initial set of observations, the process is considered as unstable. We need to find the assignable cause behind instability of the process and remove it. After all assignable causes are eliminated; a new control limit is calculated. This process is repeated until an acceptable reference set is found.

How much data is required to create the reference set? To answer this question authors in [2] considered two different utilization of reference set in control charts. One is to determine out-of-control observation due to assignable causes in the process and another one is to determine stability of the process. Typically, 25 observations are desirable to create a reference set and calculate control limits in order to detect assignable causes of variation in a software process. Achieving 25 observations requires time and might increase the risk of missing opportunity to early detection of anomalies. Assignable causes can be determined even with very few observations e.g. three or four subgroups of data even with minimal subgroup size of one. Increasing subgroup size tightens control limit and makes control limits more sensitive to small shifts. However, increasing sub group size requires more observation. We can calculate trial control limits using small reference set available. Trial limits are unreliable but they provide an opportunity to detect out-of-control observation early in a software process even before demonstrating the stability of the process.

Figure 4 Decision tree for control chart selection [13]
In case of determining stability of a software process, control limits are calculated for three major reasons: 1) determine process capability, 2) determine process behaviour, 3) compare with standards process behaviour or process requirement. Thus, we require higher confidence while determining control limits for stability analysis. In case of X-bar control chart 25 to 30 subgroups of data is required. For XmR control charts 40-45 individual observations values are desirable. Large number of observations reduces influence of few extreme values.

5.1.4 DETERMINE CONTROL LIMITS

XmR charts is a collection of two different charts X charts and mR charts. X chart plots individual observation values of the measurable process characteristics of the process. On the other hand, mR chart plots moving range or the difference between a pair of observations considered for the X chart. Centerline (CL) for both X and mR charts are calculated as an average of all observed values and all moving ranges respectively. Sigma (σ) for both X and mR charts are calculated as the standard deviation of all observed values and all moving ranges respectively. Upper Control Limit (UCL) and Lower Control Limit (LCL) are calculated correspondingly as (UCL=CL+3 σ), and (LCL=CL-3 σ).

For example, Table 2 presents 20 observations of day-to-day variation of effort spent on servicing the existing product of a company. Equations to calculate control limits for this example are presented in Table 3 below. A sample XmR chart for this example is presented in Figure 5.

<table>
<thead>
<tr>
<th>Effort</th>
<th>Moving range</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.5</td>
<td>7</td>
</tr>
<tr>
<td>43.5</td>
<td>2</td>
</tr>
<tr>
<td>45.5</td>
<td>5.7</td>
</tr>
<tr>
<td>39.8</td>
<td>3.1</td>
</tr>
<tr>
<td>42.9</td>
<td>1.4</td>
</tr>
<tr>
<td>44.3</td>
<td>0.6</td>
</tr>
<tr>
<td>44.9</td>
<td>2</td>
</tr>
<tr>
<td>42.9</td>
<td>3.1</td>
</tr>
<tr>
<td>39.8</td>
<td>0.5</td>
</tr>
<tr>
<td>39.3</td>
<td>9.5</td>
</tr>
<tr>
<td>48.8</td>
<td>2.2</td>
</tr>
<tr>
<td>51</td>
<td>6.7</td>
</tr>
<tr>
<td>44.3</td>
<td>1.3</td>
</tr>
<tr>
<td>43</td>
<td>8.3</td>
</tr>
<tr>
<td>51.3</td>
<td>5</td>
</tr>
<tr>
<td>46.3</td>
<td>1.1</td>
</tr>
<tr>
<td>45.2</td>
<td>2.9</td>
</tr>
<tr>
<td>48.1</td>
<td>2.4</td>
</tr>
<tr>
<td>45.7</td>
<td>1.6</td>
</tr>
<tr>
<td>44.1</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Daily Product Support Effort

<table>
<thead>
<tr>
<th>Effort</th>
<th>Moving range</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.5</td>
<td>7</td>
</tr>
<tr>
<td>43.5</td>
<td>2</td>
</tr>
<tr>
<td>45.5</td>
<td>5.7</td>
</tr>
<tr>
<td>39.8</td>
<td>3.1</td>
</tr>
<tr>
<td>42.9</td>
<td>1.4</td>
</tr>
<tr>
<td>44.3</td>
<td>0.6</td>
</tr>
<tr>
<td>44.9</td>
<td>2</td>
</tr>
<tr>
<td>42.9</td>
<td>3.1</td>
</tr>
<tr>
<td>39.8</td>
<td>0.5</td>
</tr>
<tr>
<td>39.3</td>
<td>9.5</td>
</tr>
<tr>
<td>48.8</td>
<td>2.2</td>
</tr>
<tr>
<td>51</td>
<td>6.7</td>
</tr>
<tr>
<td>44.3</td>
<td>1.3</td>
</tr>
<tr>
<td>43</td>
<td>8.3</td>
</tr>
<tr>
<td>51.3</td>
<td>5</td>
</tr>
<tr>
<td>46.3</td>
<td>1.1</td>
</tr>
<tr>
<td>45.2</td>
<td>2.9</td>
</tr>
<tr>
<td>48.1</td>
<td>2.4</td>
</tr>
<tr>
<td>45.7</td>
<td>1.6</td>
</tr>
<tr>
<td>44.1</td>
<td></td>
</tr>
</tbody>
</table>

Table 3 Equations for control limit calculation

<table>
<thead>
<tr>
<th>Equations</th>
<th>Sample value</th>
<th>Eq. No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( CL_x = \bar{X} )</td>
<td>45.34</td>
<td>(1)</td>
</tr>
<tr>
<td>( CL_{mr} = mR = \frac{1}{m-1} \times \sum_{i=1}^{m-1}</td>
<td>x_{i-1} - x_i</td>
<td>)</td>
</tr>
<tr>
<td>( 3\sigma = 2.660 \times mR )</td>
<td>9.54</td>
<td>(3)</td>
</tr>
<tr>
<td>( UCL_X = \bar{X} + 3\sigma = \bar{X} + 2.660 \times mR )</td>
<td>54.87</td>
<td>(4)</td>
</tr>
<tr>
<td>( LCL_X = \bar{X} - 3\sigma = \bar{X} - 2.660 \times mR )</td>
<td>35.80</td>
<td>(5)</td>
</tr>
<tr>
<td>Zone1_σ = ( \bar{X} ± σ = \bar{X} ± (2.660 \times mR)/3 )</td>
<td>48.51, 42.16</td>
<td>(6)</td>
</tr>
<tr>
<td>Zone 2_σ = ( \bar{X} ± 2σ = \bar{X} ± 2 \times (2.660 \times mR)/3 )</td>
<td>51.69, 38.98</td>
<td>(7)</td>
</tr>
<tr>
<td>( UCL_{mr} = 3.2678 \times mR )</td>
<td>11.71</td>
<td>(8)</td>
</tr>
<tr>
<td>( LCL_{mr} = 0 )</td>
<td>0</td>
<td>(9)</td>
</tr>
</tbody>
</table>
Predictively analyzing future behaviour of selected process characteristic is a key step of our solution approach. In respect of the selected process characteristic (i.e. task completion rate), predicted behaviour is generated with aid of a work schedule simulation. This is a combination of a simple work scheduling method and Monte-Carlo simulation. Simulation takes a set of task as input and creates a work schedule as output. The work-scheduling method is responsible for assigning task to the developers, calculate completion time, maintain constraints and dependencies (e.g. task and developer dependency), and prepare the task schedule. We consider task completion time as the response factor (i.e. output parameter) for the simulation. We can calculate task completion rate from individual task completion time, which is directly achievable from the task schedule. On the other hand, Monte Carlo simulation is responsible for modeling the uncertainty of the process. Based on selected uncertainty factors and corresponding range and distribution, Monte-Carlo simulation represents unforeseen uncertainties in the process. Varying input value of these uncertainty factors represents uncertain future scenarios. If we run numerous trials of the simulation, corresponding response factor values will help us to create a probable behaviour pattern (i.e. distribution) for the response factor. We utilize this behaviour pattern to predictively analyze future behaviour of the process characteristic. This process is accomplished in three major steps discussed below.

5.2 PREDICTIVELY ANALYZE FUTURE BEHAVIOUR

Predictively analyzing future behaviour of selected process characteristic is a key step of our solution approach. In respect of the selected process characteristic (i.e. task completion rate), predicted behaviour is generated with aid of a work schedule simulation. This is a combination of a simple work scheduling method and Monte-Carlo simulation. Simulation takes a set of task as input and creates a work schedule as output. The work-scheduling method is responsible for assigning task to the developers, calculate completion time, maintain constraints and dependencies (e.g. task and developer dependency), and prepare the task schedule. We consider task completion time as the response factor (i.e. output parameter) for the simulation. We can calculate task completion rate from individual task completion time, which is directly achievable from the task schedule. On the other hand, Monte Carlo simulation is responsible for modeling the uncertainty of the process. Based on selected uncertainty factors and corresponding range and distribution, Monte-Carlo simulation represents unforeseen uncertainties in the process. Varying input value of these uncertainty factors represents uncertain future scenarios. If we run numerous trials of the simulation, corresponding response factor values will help us to create a probable behaviour pattern (i.e. distribution) for the response factor. We utilize this behaviour pattern to predictively analyze future behaviour of the process characteristic. This process is accomplished in three major steps discussed below.

5.2.1 IDENTIFYING UNCERTAINTY FACTORS AND RESPONSE FACTORS

Uncertainty factors are the sources of uncertainty and risk in a process. Typically, input parameters of a process that has stochastic values are considered as uncertainty factors. Correct values for these parameters are unknown at the beginning of the process and dynamically change during the process. To predict future observations of selected characteristics of the process we require to understand the uncertainty factors they depends on. The selected characteristics of a process are considered as response factors. Considering uncertainty factors as input parameters we can achieve the response factors (as output parameters) by simulating the process. Due to the uncertainty of input parameters the response parameters varies as well. Predictive simulation model helps us to analyze these variations and predict the unforeseen future events.

Despite lack of legacy data, past knowledge and scope we need to restrict our analysis of uncertainty factors. Instead of considering all uncertainty factors, we limit ourselves only within factors that cause major risks for the particular process under observation. Other uncertainty factors are assumed constant throughout the process. For example, in this report we analyze OSS project from GitHub repository. We focus on the merging process of the OSS project. We are particularly interested to analyze the contribution rate (i.e. task completion rate) over time. We need
to simulate this process in order to predict future task completion rate in the process. Major uncertainty factors we considered here are – Task arrival, Task size, Developers productivity etc. In order to perform the simulation analysis these uncertainty factors are varied within defined range and distribution to predictively simulate unforeseen events of future process. Details regarding the range and distribution selection are presented in next section. A brief details regarding the uncertainty factors we considered are presented below. More details regarding these uncertainty factors in context of a GitHub project are discussed in Section 6.3.7.

**Task Completion Rate:** Task refers to the smallest unit of work that a team explicitly breaks out and assigns to developers. Features/User stories are logical collection of tasks that represent the amount of work required to solve an individual requirement. In this report, we focus on merging process of contributions in GitHub. Each contribution is considered as a task. The number of tasks completed in a given period is considered as task completion rate.

**Task Arrival:** Tasks may include, update or delete at any time of the process. Task arrival is the time when a task arrives and a developer can start working on it. Task completion rate is dependent on their arrival. Thus, task arrival is a key uncertain factor to consider.

**Task size:** In this report, we consider task size (in LOC) as an representation of the effort required to solve each task. Efforts are estimated at the beginning of the process. Initial effort estimations might vary during the actual process. Bohem in his Cone of Uncertainty [49] theory presented that, initial effort estimation can vary between 4 times to 1/4 times later. Based on the effort requirement, task completion time might vary. Thus, uncertainty in estimated effort (i.e. task size) is a key uncertainty factor to consider.

**Developer’s productivity:** Developers are the key force to continue the software process. In context of GitHub projects we consider collaborators as developers (details are available in 6.3). Developer productivity refers to the amount of tasks individual developers can perform within a given period. Task completion time depends on the productivity of developers. Productivity estimation may vary during the process. Thus, developer productivity is a key uncertainty factor to consider.

### 5.2.2 DEFINE UNCERTAINTY RANGES AND DISTRIBUTIONS

In order to perform predictive simulation (i.e. predict future values of selected process characteristics by simulating the process) we already determined our uncertainty factors and response factor. We vary input values for uncertainty factors and record corresponding response factor values. These response factor values help us to analyze and predict future values of the selected process characteristics. We need to systematically vary uncertain factors within defined range and distribution. For individual uncertain factors, we need to establish a variation range and a distribution function that provides the probability of a certain value within this range. Two popular ways to determine these range and distributions are 1) Expert opinion, 2) Experience database.

In case of unavailable empirical data, experts are interviewed to define a distribution for uncertainty factors. For example, we can ask experts to provide the most probable, minimum and maximum value for each individual uncertainty factor in order to construct a Triangular Distribution for each factor.

On the other hand, we can apply experience database to find distribution for uncertainty factors. If empirical data are available, we can try to fit available probability functions to find distribution of the underlying data. For applying SPC we have to construct a reference set. We can utilize the reference set as expert database to determine the distribution for uncertain factors. Available statistical tools like Minitab or Stat:fit can perform this action automatically. We utilized the EasyFitExcel [18] tool to accomplish this task.

### 5.2.3 CONDUCTING PROCESS SIMULATION

The project is considered as a collection of available tasks to complete. In the simulation, we primarily focus on predicting future values for selected response factor (i.e. task completion time). Response factor is uncertain and depended upon a pool of stochastic input parameters (i.e. uncertainty factors) e.g. size of tasks, arrival of tasks, developer’s availability, developer skill etc. We can only provide imprecise and incomplete measures for these uncertain factors. We utilize varying values for uncertainty factors within determined ranges and distribution. We achieve a range of possible values for the response factor. Varying values in input parameters represent the unforeseen events possible to happen in the process. Output values for response factors provide the prediction for its future values in different circumstances. These values help to determine predicted value for the selected response factor.

For example, in this report we focus on task completion rate in a software process. To predict these values, we primarily focus on predicting future observations of the task completion time. This is considered as the response
factor for our simulation that depends on uncertainty factors like size of tasks, arrival of tasks, and developer productivity. We utilize varying values for the uncertainty factors within determined ranges and distribution. Output values for response factor are recorded. It provides completion time for the project and for each individual task. This helps us to detect the likelihood of completion of a certain task by a certain time.

To achieve these results we applied Monte-Carlo simulation along with a work-scheduling model. Our simulation considers a set of arriving tasks along with a pool of developers. It develops a work schedule while maintaining all dependencies and constraints. The underlying work-scheduling model is responsible for assigning developers to task, compute the time-span required for each task to complete and generate work schedule. We consider dependency in task arrival and dependency among developers. All the assumptions, dependencies and constraints are considered based on each individual projects we consider in simulation. Details regarding these assumptions are available in section 6.3.9. Above-mentioned uncertainty factors are considered as input parameters. The ranges and distribution for these uncertainty factors are determined earlier. Building a work schedule out of available task sets where all input parameters are specified by a determined distribution is accomplished through the Monte Carlo simulation. Monte Carlo simulation is responsible for modeling uncertainty in the work scheduling process. The simulation randomly explores a set of work schedule in multiple \((t)\) simulation trials. Typically, \(t\) is a large number e.g. 1000 or 10000. In each of these simulation trials, it randomly considers values for input parameters within the determined range using the provided distribution. In each simulation trial, a work schedule is generated for the given set of tasks by assigning task to randomly chosen developers. In \(t\) simulation trials, it generates \(t\) different work schedules. We do not have any intention to find an optimal solution by searching the generated solution space. Instead, we would like to develop a distribution of completion time for each \(f^{th}\) \((f=1 \ldots m, m\) is the total number of tasks considered\). This distribution demonstrates the likelihood of completion of a certain task by a certain time.

After completion of \(t=1000\) trials we can find a certain completion time \(CT\) that shows \(X\%\) likelihood for \(f^{th}\) task \(CR_f\) to solve by time \(CT\). For example, \(f=1, X=75\) and \(CT=36\) means, the first task completion has 75\% likelihood that it will complete by the time \(CT =36\). We utilize these predicted values as future values for the response factor (i.e. task completion time). We can calculate our predicted task completion rate from these predicted task completion time and plot them in the control chart.

5.3 ANOMALIES DETECTION

We plot two different set of data points for the selected process characteristic (i.e. task completion rate) on the control chart. Commonly all plotted points on the control chart are referred as data points. However, to analyze process behaviour, we need to distinguish among multiple type of data points plotted in the control chart.

**Definition 1:** Observed data points are the observed values of selected process characteristic of the process under monitoring of SPC. Observed data points are responsible for determining ranges and distribution of uncertainty factors. These points are denoted as ODP.

**Definition 2:** Predicted data points are generated using the predictive simulation. These points inform future behaviour of the process characteristic. These points are denoted as PDP.

**Definition 3:** Observed anomalies are anomalies (i.e. run test failures) detected within the observed data points. ODP. Observed anomalies are denoted as OA.

**Definition 4:** Predicted anomalies are anomalies (i.e. run test failures) detected within the predicted data points PDP. Predicted anomalies are denoted as PA.

We plotted two set of data points on control chart in order to monitor and control the software process. We examine the control charts to find anomalies and instabilities in the process. Values falling outside the control limits or unusual patterns indicate presence of assignable cause of variation. Software process demonstrates significant variability on timely basis. Thus, systematic patterns in control chart are of higher interest than single point exceptions. Question for investigation is, can we apply similar anomalies detection techniques for both observed and predicted data points to detect anomalies. However, guideline for their interpretation and corrective actions will be different for OA and PA.

Western electric handbook listed four effective and popular tests for SPC control. These tests are known as “detection rules” [44], [50]. These are the most popular and often the only used SPC tests in software context. In Table 4 below these tests are listed as RT1 to RT4. Emphasizing detection of systematic patterns in control chart, eight rules were proposed in [14]–[16] known as “Run Rules”. Run rules are presented in Table 4 below as RT1 to RT8. These rules were not popular earlier in SPC for software process. Considering the significant variability
software process demonstrates on timely basis, “Monitoring problem-SPC based solution” approach proposed applying Run Rules for SPC in software context.

Run Rules (i.e. RT1 to RT4) are created based on statistical reasoning and past observations. Probability of an observation to fall beyond the upper/lower control limit is 0.00195. Each of these run rules has a pattern that includes multiple observations plotted in the control chart. The total probability to occur this pattern is equal to the probability of an observation to fall outside the 3sigma control limit. Details regarding statistical interpretation of all these run tests are available in [51]. Applying these run test has been successful in software context. Thus, in our proposed approach we consider these Run Test to detect anomalies in a control chart both in ODP and PDP. For ease of understanding of these run tests the UCL and LCL area around CL are divided in convenient zones each of one sigma distance e.g. Zone1σ, Zone2σ. Moving range (mR) charts shows lack of symmetry around CL. Thus, Zone tests are applicable in X chart only. All eight run tests and how they should be applied in ODP and PDP are discussed below. We are interested in investigating whether these run tests are capable of detecting anomalies within PDP as well or not. Run tests are presented in three categories based on the type of information they provide.

<table>
<thead>
<tr>
<th>Run Test</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT1: Three Sigma</td>
<td>1 point beyond a control limit (±3sigma)</td>
</tr>
<tr>
<td>RT2: Two Sigma</td>
<td>2 out of 3 points in a row beyond (±2sigma)</td>
</tr>
<tr>
<td>RT3: One Sigma</td>
<td>4 out of 5 points in a row beyond (±1sigma)</td>
</tr>
<tr>
<td>RT4: Run above/below CL</td>
<td>7 consecutive points above or below the centerline</td>
</tr>
<tr>
<td>RT5: Mixing/Overcontrol</td>
<td>8 points in a row on both sides of the centerline avoiding ±1sigma area</td>
</tr>
<tr>
<td>RT6: Stratification</td>
<td>15 points in a row within ±1sigma area</td>
</tr>
<tr>
<td>RT7: Oscillatory Trend</td>
<td>14 alternating up and down points in a row</td>
</tr>
<tr>
<td>RT8: Linear Trend</td>
<td>6 points in a row steadily increasing or decreasing</td>
</tr>
</tbody>
</table>

5.3.1 SIGMA TESTS

Sigma tests are responsible to detect presence of assignable cause. Three tests RT1-RT3 falls under this category. RT2 and RT3 i.e. one and two sigma tests are zone tests and thus only applicable to X chart.

Three Sigma Test: If any observation value falls beyond the control limits (i.e. UCL or LCL) of the control chart three sigma test fails. It indicates an assignable cause is present in the process. Three sigma tests are applicable to both X and mR charts. Due to high variation in software process and lower trustworthiness of predicted values in PDP three sigma test is considered of lowest importance.

Two Sigma Test: If two observation values out of three successive points falls on the same side of the centerline beyond two sigma distance, two sigma test fails. This failure indicates an out of control situation and notifies early about shift in process. This is a zone test and thus not applicable for mR charts.

One Sigma Test: If four out of five consecutive points fall on the same side of the centerline beyond one sigma level this test fails. It also indicates out of control situation and notifies process shift early in Software process. As this is a zone test it is not applicable for mR charts.

5.3.2 LIMIT TESTS

Run Above or Below CL Test: If 7, 8 or 9 consecutive observations fall above or below the centerline, this test fails. Such failure indicates a shift of the centerline. This indicates process mean is changed. This is also considered as a zone test and thus not applicable for mR charts.

Mixing/Over Control Test: If eight successive points fall on either side of the centerline beyond the zone1σ level, this test fails. It is a zone test as well and only applicable in X chart.

Stratification Test/Reduced Variability Test: If 15 consecutive points fall on either side of the centerline within zone1σ area, this test fails. This test indicates an out of control condition. Due to its dependency on zone1σ area this test is not applicable in mR charts.

5.3.3 TREND TESTS

Oscillatory Trend Test: If 14 consecutive points systematically oscillate up and down, this test fails. It signals a systematic trend in the process and thus considered as a trend test. This test is applicable in both X and mR chart.
Linear Trend Test: If six consecutive points show a systematic increase and decrease in the process, this Linear Trend test fails. It presents increasing or decreasing trend in the process. It does not consider zones of the chart area and thus applicable in both X and mR charts.

5.4 INTERPRET ANOMALIES AND CAUSES INVESTIGATION:

Statistical process Control is a time-series-analysis technique for process monitoring. SPC is capable of detecting out-of-control situations, anomalies or instabilities in the process. However, it cannot provide any guideline for assignable cause investigation or corrective actions to prevent anomalies. “Monitoring problem-SPC based solution” approach investigated on this challenge and proposed guidelines for causes investigation process. Our proposed approach is an extension to these earlier guidelines. Similar guidelines for cause investigation are followed, when anomalies are present only within observed data points (ODP). We proposed an additional set of guidelines for cause investigation when anomalies are detected within both ODP and PDP. A run test (RT) failure can occur in three different ways- i) within ODP ii) within PDP or iii) within a combination of ODP and PDP. Based on the location where the RT failure has occurred, interpretation of the failure can be significantly different. Interpretation of different RT failures occurred in different scenarios are summarized in Table 5. A brief discussion of the proposed cause investigation process is presented below.

5.4.1 SIGMA TEST (RT1, RT2 & RT3) FAILURE

Three sigma test (i.e. RT1) fails if a single observation data point falls out of control limits (i.e. UCL or LCL). Due to high variability in software processes, such failures occur numerous times. These failures are considered as an early alarm and point towards less meaningful assignable causes e.g. employee absence. One or two sigma test (i.e. RT2 & RT3) failures also indicate early alarm of potential anomalous trend. These might undertake assignable causes. However, in software context, specific trends or patterns among a group of observations are more meaningful than isolated observations falling out of the control limits.

Sigma test failures are early indication towards changes or anomaly in the process. No cause investigation is initiated for these failures, due to lack of understanding what type of assignable cause or process change they indicate. PDP can play a role of indicator in this context. Suppose a sigma test (RT1-RT3) failure in ODP is followed by a Limit or Trend test (i.e. RT4-RT8) failure in PDP. Predicted anomalies inform which type of possible failure sigma test indicated. It confirms the early alarm of sigma test and helps early investigation of the assignable cause by indicating what type of failure need to be investigated. If sigma test failure is visible in PDP, it is considered as an early alarm and we need to wait for more observation before we start any investigation.

5.4.2 RUN ABOVE/BELOW CL TEST (RT4) FAILURE

In software processes, a sequence of points following a pattern indicates towards occurred changes in the process. If eight consecutive points fall on one side of the centerline, RT4 fails. This test is considered as Limit test. If a RT4 failure occurs in observed value, it is a clear indication that the process mean has been shifted. It initiates a calculation for new mean. Based on the performance improvement or deterioration the cause for these changes might be desirable or undesirable.

If a RT4 test fails in PDP, it indicates a process mean shift in future. However, it does not trigger CL re-calculation process. Instead, if an undesirable shift in the process performance is predicted, RT4 failure initiates an early investigation to find the assignable cause. If RT4 failure happens in combination of ODP and PDP, PDP help to early detect the RT4 failure. If RT4 failure in PDP is detected as a continuation of an observed anomaly, it provides a strong evidence for the earlier observed failure.

5.4.3 MIXING / OVER CONTROL TEST (RT5) FAILURE

Mixing/over control test failure indicates over control (hyper-adjustments) or increased variability of the process. This failure is typically followed by induced improvements in the process. Any RT5 failure in observed values indicates the increased instability due to induced improvement until the improvement is fully acquired by the organization and employees.

We need to properly present the induced improvement offered to the process in our predictive simulation model. Predicted data should indicate towards the future effects of this improvement. If an RT5 failure occurs in PDP, it re-confirms the influence of the induced improvement. It also notifies the amount of variability that will follow the induced improvement in future. However, if no known induced improvement is present in the process, then RT5
Table 5 Interpretation of detected anomalies in consideration of observed and predicted behavior of the process.

<table>
<thead>
<tr>
<th>Run Test Failure</th>
<th>Interpretation of observed anomalies</th>
<th>Interpretation of predicted anomalies</th>
<th>Additional interpretations</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT1</td>
<td>Early Alarm</td>
<td>Early Alarm, No Action</td>
<td>If RT1 failure in ODP is followed by RT 4-8 failure in PDP, an investigation should initiate</td>
</tr>
<tr>
<td>RT2</td>
<td>Early Alarm</td>
<td>Early Alarm, No Action</td>
<td>If RT2 failure in ODP is followed by RT 4-8 failure in PDP, an investigation should initiate</td>
</tr>
<tr>
<td>RT3</td>
<td>Early Alarm</td>
<td>Early Alarm, No Action</td>
<td>If RT3 failure in ODP is followed by RT 4-8 failure in PDP, an investigation should initiate</td>
</tr>
<tr>
<td>RT4</td>
<td>New Mean</td>
<td>Early indicates the type of change expected in mean. If undesirable changes are predicted, an early investigation is initiated.</td>
<td>If RT4 failure occurs within ODP and PDP, it provides early indication for the type of change in mean.</td>
</tr>
<tr>
<td>RT5</td>
<td>Increased Variability</td>
<td>If followed by any failure in ODP, it provides an early indication for type of change (increase/ decrease) to expect in variability. Otherwise indicates existence of assignable cause. Initiate investigation.</td>
<td>If RT5 failure occurs within ODP and PDP, it provides early indication for RT5 failure.</td>
</tr>
<tr>
<td>RT6</td>
<td>Decreased Variability</td>
<td>If followed by RT4 failure in ODP, indicates decreased variability. If followed by RT5 failure in ODP, indicates type of variability based on values. Otherwise indicates existence of assignable cause. Initiate investigation.</td>
<td>If RT6 failure occurs within ODP and PDP, it provides early indication for RT6 failure.</td>
</tr>
<tr>
<td>RT7</td>
<td>New sources of variability</td>
<td>Indicate new sources of variability. Initiate investigation</td>
<td>If RT7 failure occurs within ODP and PDP, it provides early indication for RT7 failure.</td>
</tr>
<tr>
<td>RT8</td>
<td>Ongoing Phenomena</td>
<td>If followed by RT5 failure in ODP, provides early indication of the maturity effect. Otherwise, initiate investigation for assignable cause.</td>
<td>If RT8 failure occurs within ODP and PDP, it provides early indication for RT8 failure.</td>
</tr>
</tbody>
</table>
failure is unexpected. It indicates towards an unknown assignable cause present in the process. Therefore, an assignable cause investigation process should initiate to find and remove the cause.

5.4.4 STRATIFICATION TEST (RT6) FAILURE

This test refers to the maturity effect on a software process. Induced improvement (e.g. introducing a new technology) increases variability of the software process. With more knowledge and experience, process variability decreases over time i.e. the maturity effect. RT6 failure in ODP detects the presence of an assignable cause. However, overall impact of an RT6 failure can be positive as it indicate towards maturity effect.

If an RT6 failure in PDP is preceded by a RT5 (i.e. mixing and over control test) failure in ODP, it is a positive indication for the process. While RT5 indicates increased variability due to induced improvement, RT6 informs increased stability after the improvement is properly acquired. If RT6 failure within PDP is preceded by an RT4 failure within ODP, it indicates decreased variability of the process. Otherwise, if no indications are present in ODP an RT6 failure within predicted values will indicate the existence of an unknown assignable cause and requires early investigation.

5.4.5 OSCILLATORY TREND TEST (RT7) FAILURE

This test indicates an ongoing change or just occurred change. If two alternating causes create different results, RT7 fails. This test requires 14 consecutive observations. PDP help to achieve this 14 observation and indicate the RT7 failure early. If RT7 failure observed solely in the PDP or ODP values, it indicates the same meaning. We need investigation to isolate alternating causes and assignable cause responsible for RT7 failure.

5.4.6 LINEAR TREND TEST (RT8) FAILURE

Linear Trend test typically fails due to increasing or decreasing trends in observations. This failure is preceded by induced improvement such as introducing new technology or maturity effect in the software process. This test requires six consecutive data points, but more observations help to achieve higher confidence on the detected trend. This test is important to understand the variability in the process.

If an RT8 failure occurs in PDP, it helps to achieve higher confidence in observed linear trends (RT8 failures). If RT8 failure is preceded by RT5 failure in ODP, it provides an early indication towards the maturity effect. If none of them is the case, RT8 failure will indicate towards ongoing change and requires investigation to detect the assignable cause if the change is undesirable.

5.5 TUNING CONTROL LIMITS

SPC control limits are the key indicators in control charts to detect anomalies in process variation. SPC control limits should represent the underlying process and need to be tuned when process changes. Too tight control limits creates too many false alarms while too wide control limits creates the risk of missing anomalies in the process. Due to high variability in software processes, control limits require continuous tuning. Two major steps in tuning control limits are 1) identifying process performance changes, 2) re-calculating control limits according to identified changes.

In our proposed approach, we predicted future behaviour for process characteristics to facilitate early discovery of potential future anomalies. This process helps to initiate early investigation for an assignable cause, prevent the anomaly or minimize influence of the anomaly on software process. Tuning control limit is performed only after the process experienced changes. Predicted behaviour should not have any direct impact on tuning control limits. Predicted behaviour initiates corrective mechanism so that process never experiences such changes or experience changes in a controlled way. In tuning control limits, we solely rely on ODP. Therefore, we consider the “Monitoring problem-SPC based solution” approach to tune control limits in a software process. However, predicted values and early investigation might help to calculate trial limits (i.e. temporary control limits). Tuning control limit based on “Monitoring problem-SPC based solution” approach are briefly discussed below-

5.5.1 OCCASIONAL CHANGES

In case of occasional changes, if an assignable cause is detected and removed, control limits will remain same. However, if assignable cause is made a part of the process, control limits need to be re-calculated. However, further observations are required for re-calculating control limits. If predicted values are available, and it presents a trend or limit failure, it can help us to calculate trial limits (i.e. temporary control limit). We can predict the trend and type of changes the process will encounter, due to considering assignable cause as part of the process. However, control
limits should be calculated only after real world data are available. If predicted data are available but no trend or limit test failure is present in it, we have to wait and gather more observations to tune our control limits.

5.5.2 OCCURRED CHANGES

Occurred changes are already observed changes. They require changes in the control limit. If observed values show the process mean or variability are already changed then corresponding control limits should be re-calculated as well. Identification of any new source of variability requires these new sources to be investigated and represented in different charts if required. Predicted values in case of occurred changes, confirm observed failures and create stronger evidence for observed anomalies. However, if occurred change is detected solely in PDP, it calls for an early investigation to find the assignable cause. However, it will not cause any changes in the control limit.

5.5.3 ONGOING CHANGES

In case of ongoing changes, actual observations express change. These observations are no longer suitable as reference set. New reference set and corresponding control limits need to be determined. However, it requires additional observations. If PDP are present and reconfirm observed anomalies (process change), PDP can help in this regard. Predicted process changes can provide indications regarding the trend of ongoing changes. We can calculate early trial limits based on these indications. However, control limits will be calculated only after real observations are available.

6. CASE STUDY

In this report, we introduced predictive analysis of software process anomaly in “Monitoring problem-SPC based solution” approach [12], [13] as an extension. This approach can predictively detect future anomalies or confirm the future impact of observed anomalies. Due to unavailability of in-house software development data, we attempted to validate our proposed approach with open source software (OSS) projects. Mining OSS repositories in order to achieve extensive amount of data for a prediction model is a challenge itself. We attempted to validate our proposed approach with the limited data available in the OSS repository. In this section, we present seven scenarios along with illustrative examples. First, we discuss our data sources, data collection method, challenges and limitation in data collection along with the validation methodology.

6.1 DATA SOURCE

Our data source is a popular online open source software (OSS) repository called GitHub[17]. Github is a web based hosting service for projects that use Git as a revision control system. Git is an open source program for tracking changes in files. Initially launched in April, 2008, GitHub repository already gained popularity in open source software community and hosts significant number of OSS projects. Along with providing revision control facilities GitHub also offers social networking functionalities for better collaboration e.g. feeds, followers, network graph etc. Before we enter into the technical details of data collection methodology, some terminologies and concepts of GitHub need to be made clear. We briefly introduce some essential concepts of GitHub repository below.

- **Git**: Git is a revision control mechanism. It is an open source program for tracking changes in files. The core technology of GitHub is built on top of Git. GitHub provides online hosting service for OSS projects that use git as a revision control system.
- **Repository**: Repository is the projects folder, where project files related to individual GitHub projects relies. Revision histories for all individual project files are stored with a unique id. Public and private repositories offer different access rights to projects contributors.
- **Commit**: Commit means individual change (revision) to a file or set of files. Every time a revision is saved a unique ID is created. This unique id helps to track all changes made in a specific revision, when and by whom. Additionally it also stores commit message that provides a summary of the changes.
- **Collaborator**: Collaborators are the maintainer of the project. Typically, collaborators are invited to contribute to a project repository. Collaborators have write access if the repository is public, or read, write access if the repository is private
- **Contributor**: Contributor contributes to a project by having issue creation, issue solution, or a pull request merged. They might not have collaborator access to the repository.
- **User**: Users have personal GitHub accounts. They can create and contribute. Users can be, invited to join or collaborate one or multiple repositories.
• **Branch**: Branch is a parallel version of a repository. It resides within the repository. Branch allows contributors to work freely without affecting the master branch or the live version. After finalizing all changes in a branch, the branch can be merged back to the master branch.

• **Issue**: Issues are suggested improvements, tasks or questions related to the repository by contributors. Issues are created and solved by any User. Collaborators are responsible for moderating and merging issues in main branch. Issues can be assigned to specific contributors. Each issue has a discussion forum associated with it for related discussions and comments. An issue is marked closed after it is solved and merged into the main branch.

• **Push**: Contributors perform changes in their local copy of the repository. To make changes available in the remote repository, contributors need to push their changes to the remote repository e.g. GitHub.com.

• **Pull Request**: Pull requests are proposed changes to a repository submitted by a user and accepted or rejected by repository collaborators. After completion of changes to a personal copy of the repository, a pull request is created to merge committed changes in the main repository. Collaborators accept the pull request and merge changes in master branch or reject the pull requests.

• **Pull**: Pull is the process of taking in changes from remote repository to personal local repository. Contributors work on their local repositories. To keep local repositories updated they require to pull the remote repository frequently.

• **Fork**: Fork is a personal copy of a repository created from the remote repository. It allows working parallel on the repository and making changes freely. Forks are connected with the remote repository and can submit pull request to master branch when required.

### 6.2 AVAILABLE DATA IN GITHUB REPOSITORY

GitHub repositories offer data related to revision history of hosted projects. Required information are not always readily available. Utilizing OSS development culture, assumptions and calculation with repository data help us to achieve required information. We briefly discuss some major type of information available in GitHub repository.

#### 6.2.1 COMMIT RELATED INFORMATION

GitHub repository records revision (commits) history of each file in the repository. Some key information that a commit saves include unique commit id, author (i.e. the committer), authors id and email, assigned contributor, number of commits related with the issue, number of files changed, committers worked on the issue, status of the issue, number of lines added or deleted, closing date and discussion forum. Optionally GitHub also stores information regarding which commits related to which issues. Each issue are optionally marked with labels that presents category of the issue (e.g. design, development etc.), its priority level and milestones related with it. These optional information are not always available.

#### 6.2.2 ISSUE RELATED INFORMATION

Issues are suggested improvements, tasks or questions related to the repository by any user, if the repository is public. Stored information regarding issue include creation date and time, unique issue number, creator name, id and email, assigned contributor, number of commits related with the issue, number of files changed, committers worked on the issue, status of the issue, number of lines added or deleted, closing date and discussion forum. Optionally GitHub also stores information regarding which commits related to which issues. Each issue are optionally marked with labels that presents category of the issue (e.g. design, development etc.), its priority level and milestones related with it. These optional information are not always available.

#### 6.2.3 PULL REQUEST RELATED INFORMATION

Pull requests are a special kind of request made by contributors to the collaborators of the project. Contributors do not have write access to the repository. They fork a repository and create changes on their personal copy of the repository. After the changes are final, they create a pull request. Collaborators check integrity of the code and test it against the existing software. If everything works perfect, collaborator merge requested changes on the main repository and mark the pull request as closed. GitHub maintains a complete list of all open and closed pull requests. GitHub stores pull request related information that includes who issued the pull request, who is assigned to solve it, when issued, when closed, status, commits related to the pull request, files and code changed, parent file etc. Additionally each pull request also has a discussion forum where related conversations are recorded.

#### 6.2.4 OTHER INFORMATION

GitHub is a web-based tool that facilitates collaboration in OSS development. Two primary challenges in maintaining development collaboration are to keep all the members informed about other people work, and resolve conflict among different peoples work. To address the first challenge, GitHub introduced an important visualization
for repositories called Network Graph Visualizer. It helps contributors to understand what others are doing and collaborate accordingly.

![Network Graph Visualizer](image_url)

**Figure 6 GitHub Network Graph Visualizer for “octokit.rb” project**

Above we can see a network graph for a sample project “octokit.rb” [52]. Network graph presents all the contributors on left hand side in individual rows. Commits are presented across from respective contributors. In respect of a selected contributor, the network graph is drawn considering that contributor as root. For this example in Figure 6, we selected user “pegwaynn” to draw the network graph. “pegwaynn” is the owner of this repository. He owns the main repository presented as “octokit” in the network graph. “pegwaynn” works as an collaborator in this project. The network graph considering “octokit” as root helps to track advancement in the main repository.

Commits from all contributors are presented in this network graph (Figure 6). Commits, which are not merged into the main repository (octokit) yet, still exist on personal forks of each contributor. These commits are drawn across respective contributors. If a pull request is closed and merged into “octokit”, all commits related to the pull request are drawn from “octokit”. Merged commits are no longer drawn across the individual contributors as they are already reflected into “octokit” (i.e., main repository). However, information regarding contributors are still available in the commit history. Network graph provides a clear view on when, which commit is performed, by whom, on which branch, when respective pull request was opened and closed, when merge happened, by whom etc.

To address the second challenge regarding collaboration GitHub provides a facility called merge conflict. If two merges are requested with conflicting codes, Github is capable to detect these conflicts automatically. It will list all conflicts to the collaborator responsible for the merge. Collaborator can individually go through each of these conflicts, test them, communicate with the committer and decide how to resolve the conflict. Github recently introduced new visualization facilities for revision history that includes individual contributor’s contribution over time, commit activity over time, code frequency over time etc. For this research, we have not used any of these visualizations. Hence, no details are provided regarding these visualization facilities in this report.

### 6.3 DETAILS DESCRIPTION OF SCENARIO 1 (ILLUSTRATIVE EXAMPLE)

#### 6.3.1 DATA FOR SCENARIO 1

GitHub OSS repository is our selected data source. For our case study, we have chosen “octokit.rb” repository. Octokit is a ruby toolkit development project for Github API. API wrappers are responsible for reflecting the idioms of a specific language. Octokit.rb wraps the GitHub API in a flat API client. It follows Ruby conventions and requires little knowledge of others. “Octokit.rb” managed its revision history using Git and available online in GitHub. This project started on Dec 8, 2009, has 81 releases, 1385 commits, 402 issues and, 100 contributors. 394
issues are already solved and 8 are still open. This project is popular among GitHub users as 257 users currently forked this repository and 68 users are watching this repository.

### 6.3.2 DATA COLLECTION PROCESS

In our case study, we focused on the amount of contribution performed in a project over time. Before we determine our measurement objective for the case study, we need to understand what do we mean by contribution and contribution requests, how they occur and solved in a GitHub project. We consider contribution and contribution request as follows.

**Definition 5:** Contribution are considered as the code changes (addition or deletion) in main repository performed by contributors and accepted by collaborators. Contributions are performed on the main repository. It is different from commits, as commits can occur on local forks of the main repository.

After studying the GitHub project in depth, I categorized contribution occurrence in main repository in four major categories. These are briefly discussed below.

**Category 1:** Issues are suggested improvements, tasks or questions related to the repository by any user (in case of public repository). Issues are created by user, solved by contributors, moderated and closed by collaborators. A contributor is either assigned or voluntarily work to solve an issue. Contributors fork the main repository and independently commit on their personal copy of the repository. After the issue is solved, a pull request is generated to merge changes (i.e. the improvement made) in the main repository. Collaborators analyze and test the pull request and merge code changes into the main repository. The issue is marked as solved. In this category, we know when an issue is opened and closed along with when the pull request was created and closed. Based on the open and close date of an issue it is not possible to calculate the amount of time/effort required for the issue.

**Category 2:** Contributors of a project always act towards improvement of the project. After substantial amount of commits are performed on the personal copy of the repository, contributors create a pull request. These pull requests reflects recent improvements in the main repository and makes them available to all. These pull requests are moderated, tested and merged into the main repository by collaborators. In this contribution category, no official issue is listed. Instead, a pull request is requested that informs the required time to solve and merge the pull request.

**Category 3:** Collaborators work towards improvement of the project or solution of an issue. After commits are performed on personal copy of the main repository, a pull request is created. Typically, collaborators have write access to the main repository and can solve merge requests by merging code changes. These pull requests are typically solved and merged by the collaborator who requested for it. Thus they typically take minimal time.

**Category 4:** Collaborators work towards improvement of the project. Collaborators typically have write access to the main repository. Instead of performing commits on a personal local copy of the repository and pull changes later, they can commit directly on the main repository as well. It is an contribution as code changes occur in the main repository, but no information regarding issues or pull request are available.

Some definitions related to contribution are utilized throughout our case study. To understand the data collection process they are required as well. Therefore, these definitions are listed below.

**Definition 6:** Contribution Size refers to the number of LOC changed (i.e. added or deleted) on the main repository during the contribution. Lines of code (LOC) changes performed while development can be substantially different from LOC changes accepted for merging into the main repository. Contributions are changes performed in the main repository. Thus, Lines of code (LOC) changes accepted to merge into the main repository should define the size of each contribution. Contribution size refers to the task size discussed earlier.

We assume total \( m \) contributions are performed within time \( T \). For each contribution \( C_i \) \( (i = 1 \ldots m) \) if the LOC added or deleted are \( ADD(C_i) \), \( DEL(C_i) \) respectively, then size of this contribution \( Size(C_i) = ADD(C_i) + DEL(C_i) \).

**Definition 7:** Contribution rate (\( C \)) is the number of contributions merged into the main repository within a specific timeframe \( T \). For example if \( T = 24 \) hour, the number of contributions merged daily is the contribution rate \( (C=m/T) \). Contribution rate is an indication of the project growth. Contribution rate refers to the task completion rate discussed earlier.

**Definition 8:** Contribution Request (\( CR \)) is a request of code changes (addition or deletion) on the main repository created by contributors or collaborators. If accepted by collaborators these contribution requests are reflected on the main repository and becomes visible to all. Size of a contribution request is the amount of code
changes performed on the main repository after acceptance from a collaborator. In category 1, 2, and 3 the pull request and in category 4 the commit on main repository are considered as the contribution request (CR). Contributions requests refers to the task discussed earlier in this report.

In all these contributions we can clearly distinguish between two types of efforts. These are development effort and merging effort. Related definitions are provided below-

**Definition 9**: Contributors and collaborators work towards solving an issues or performing certain improvement of the project. Effort spent during these actions is considered as Development effort. Development effort is denoted by (DE).

**Definition 10**: Collaborators work towards solving a pull request, testing authenticity of the code, analyzing possible risk in integration of the change, removing merge conflict and merging code into the main repository. Effort spent in these actions considered as merging effort. Merging effort is denoted by (ME)

**Definition 11**: Merging time is the time required to merge a contribution. This time is calculated as the difference between open and merging date of a pull request. Time required to merge a contribution \( C_i \) is calculated as the difference between starting date i.e. \( \text{Start}(C_i) \) and merging date \( \text{Merge}(C_i) \). Thus we consider merging time \( \text{MTime}(C_i) = \text{Start}(C_i) - \text{Merge}(C_i) \). Merging time refers to task completion time discussed earlier in this report.

In this case study, we focus on the contribution rate of the project. Accurate quantitative information regarding the amount of development or merge effort required for each contribution is not available. We consider contribution size as a representation of the amount of effort required for each contribution. Contribution requests (CR) arrive after one or multiple contributors worked on their local repositories. Other than category 1 contributions, we do not have information regarding the starting or closing time or required effort for the development of a contribution. However, once a CR has arrived, it is possible to gather quantitative data regarding merging time and merging effort of the CR. For simplicity, we only consider merging effort for each contribution in this study. Development effort is considered out of scope for this study.

The amount of merging effort required for a contribution is considered equivalent to the contribution size. Only collaborators can perform merge action and thus we consider collaborators responsible for all merging effort. For example in “octokit.rb” we have two collaborators responsible for merging all CR within the time considered for our study. Utilizing all closed pull request information, we can calculate the skill level for each collaborator.

**Definition 12**: Skill level for a collaborator \( D_i \) is denoted as \( \text{Skill}(D_i) \). Utilizing a contribution request \( C_i \) (i = 1..n) information merged by collaborator \( D_i \), we can calculate skill level as

\[
\text{Skill}(D_i) = \frac{\text{Size}(C_i)}{\text{Time}(C_i)} = \frac{\text{ADD}(C_i) + \text{DEL}(C_i)}{\text{Start}(C_i) - \text{Merge}(C_i)}
\]

We collected our case study data from the above mentioned data source. Four year project history is beyond our scope to consider. For this case study, we consider development history between 12 August 2013 to 21 December 2013 on the master branch of “octokit.rb”. Within this time period “octokit.rb” experienced 60 pull request issues, 50 individual commits on the master branch performed by 20 contributors and 2 collaborators.

### 6.3.3 DATA COLLECTION TOOLS

In order to collect required data from GitHub repository a combination of automated tools and manual effort was utilized. We developed a program in Java language using Eclipse IDE to control and fetch data from online GitHub projects. We utilized two third party GitHub API available for Java language. The GitHub Java API (org.eclipse.egit.github.core) library is part of the GitHub Mylyn Connector and aims to support the entire GitHub API v3. GitHub API v3 allows connection with Git Hub repository through Mylyn connector. It also allows to fetch and manipulate repositories from the java program. It facilitates flexibility and automation in GitHub data collection process. As we only have read access to the repository, we had to fulfill all our data requirements utilizing fetch actions available in GitHub API v3. We fetched some readily available information from the repository using this program. These information include commits, commit author name, commit date, LOC added/deleted in a commit, pull requests, issues associated with pull requests, opening and closing date for issues, opening and closing date for pull requests, assigned developers, contributors, merged date, closed date, collaborators, LOC added or deleted, number of files changed, contribution history, collaboration history etc.
We need to classify and select valid contributions on the main repository of the project. We utilized “GitSVM” and “GitHub for Windows” tools to create clone (i.e. a personal copy of the repository on local machine) of the repository and analyze it in more details. Downloaded log files from GitHub repository and comments available in discussion forums helped us in this analysis. We collected all four categories of contributions. These are considered as valid contributions. Some contributions are not listed as an issue or pull request. Network graph can clearly distinguish commits performed on master branch by collaborators. Therefore, we manually analyzed the network graph of the GitHub project. It helped us to understand the changes performed on the master branch in the main repository and find valid contributions. Contribution request less than 4LOC changes are not considered as significant contribution. After we select the valid significant contributions, all required data are fetched using the developed java program.

6.3.4 PROCESS CHARACTERIZATION

It is ineffective to apply statistical process control (SPC) for the entire software development process. Practical experience reports, it burdens project and organization with measuring activities [11]. Thus instead of selecting the complete development process or complex sub-process to monitor under SPC we require to choose process characteristics of our interest. We should clarify our business goals first, identify and prioritize process characteristics of interest based on this. For our case study, we consider contribution performed over time on master branch of main repository as our point of interest. It represents growth of the project. Next in consideration of the selected process characteristics, we need to select and define our measures. In this case study, we consider contribution rate as our selected measure.

6.3.5 SELECTION OF CONTROL CHART

Different control charts are available for variable and attribute data. These charts have special characteristics to become useful in different contexts. Figure 4 presents a tree of control charts to facilitate the selection process. Considering the context, we need to select an appropriate control chart. From our discussion earlier, we already came to know that XmR and U charts are suitable for software processes. In this case study, we have attribute data and the data distribution is unknown (it is determined dynamically at runtime). The area of opportunity is not fixed. It might change over time. Therefore, a suitable chart for our context is the XmR chart. In an XmR chart X chart represents single observation values and mR chart represents moving range values. We already discussed how XmR charts are useful in software context in Section 5.1.2. XmR chart has been popular in literature as well.

6.3.6 IDENTIFYING REFERENCE SET AND CONTROL LIMITS

We consider contribution rate as our selected process measure and XmR chart as selected control chart. For this case study, we consider development between 2 August 2013 to 21 December 2013 of “octokit.rb” project in GitHub repository. Each observed data point represents daily contribution rate (i.e. contribution rate in respect of 24 hour or a day). We consider first 35 observed data points as our reference set. Reference set data is presented in Table 6. After the reference set is determined, we plot these data in a control chart and calculate the control limits. In our case, one data point in the reference set presents RT1 (i.e. three sigma test) failure. We need to either remove this data point along with the underlying assignable cause or wait for a more stable set of data to re-calculate control limits. As we considered post mortem study of an OSS project, it is not possible to detect and remove the assignable cause. Therefore, we initiate monitoring with current control limits (i.e. trial limits) and tune these limits as soon as a more stable set of data points become available.

However, contribution rate in OSS software are typically unstable due to the underlying culture of OSS development and less control on the contributors. We selected an unstable reference set on purpose. We want to investigate, if initially stable dataset is not available, can we continue monitoring our project with trial limits. We will tune trial limits, as soon as stable data becomes available. This example also demonstrates, how tuning control limits influence the monitoring process.
After the control
r corresponding values for selected uncertainty factors. Initially we have 35 observed data points in the
, legacy data o
r past knowledge
dent. Uncertainty factors are the sources of uncertainty and
, distribution request arrival, collaborators,
malies. As soon as we
ere
represent
more contribution
and thei
are hard to achieve. Instead of creating the burden
to determine such ranges and distribution. Due to process diversity
factor. We need to establish a range for this variation and a distribution fun
6
simulated events of the process. In our solution approach
and distribution to predictively simulate unforeseen
collaborators skill level. To conduct the simulation analysis, these uncertainty factors
our case study.
Due to lack of data availability and scope, we have to restrict the number of
uncertainty factors on which
simulation, we consider completion time for contribution requests as the response factor. We
consider simulating
under SPC.
We want to predict future behaviour for the selected process characteristic i.e. contribution rate for the process
occur
within this range.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>17</td>
<td>1</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>10</td>
<td>3</td>
<td>18</td>
<td>0</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>11</td>
<td>10</td>
<td>19</td>
<td>0</td>
<td>27</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>12</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>21</td>
<td>2</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>14</td>
<td>0</td>
<td>22</td>
<td>2</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>15</td>
<td>1</td>
<td>23</td>
<td>2</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>16</td>
<td>1</td>
<td>24</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
</tbody>
</table>

We calculate control limits based on this reference set using equations presented in Table 3. Calculated control
limits for the given reference set are listed below in Table 7 for X and mR charts accordingly. In order to calculate
control limits, present data points, and control limits on control chart we used MS Excel 2010. After the control
limits are determined, the process is continuously observed. Observed data points are plotted on the control chart and
tested with the run tests presented in Table 4 to detect anomalies. As soon as we receive a more stable set of data
points these control limits will be re-calculated.

Table 6 Initial Reference Set values for case study-scenario 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>17</td>
<td>1</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>10</td>
<td>3</td>
<td>18</td>
<td>0</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>11</td>
<td>10</td>
<td>19</td>
<td>0</td>
<td>27</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>12</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>21</td>
<td>2</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>14</td>
<td>0</td>
<td>22</td>
<td>2</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>15</td>
<td>1</td>
<td>23</td>
<td>2</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>16</td>
<td>1</td>
<td>24</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7 Control limits in respect of the Reference Set for case study-scenario 1

<table>
<thead>
<tr>
<th>X Chart</th>
<th>mR Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCLx (3σ)</td>
<td>Zone 2σ</td>
</tr>
<tr>
<td>5.09</td>
<td>3.81</td>
</tr>
</tbody>
</table>

6.3.7 IDENTIFYING UNCERTAINTY FACTORS AND RESPONSE FACTORS

We want to predict future behaviour for the selected process characteristic i.e. contribution rate for the process
under SPC. We conduct a predictive simulation to accomplish this. To predict values for contribution rate, we
consider simulating completion time (i.e. merging time) for contribution requests (CR). Therefore, in predictive
simulation, we consider completion time for contribution requests as the response factor. We need to determine the
uncertainty factors on which this response factor is dependent. Uncertainty factors are the sources of uncertainty and
project risk. These factors are the input parameters of the predictive simulation model that have stochastic values.
Due to lack of data availability and scope, we have to restrict the number of uncertainty factors we can consider in
our case study.

Key uncertainty factors considered in this study are – contribution request size, contribution request arrival,
collaborators skill level. To conduct the simulation analysis, these uncertainty factors are varied with defined range
and distribution to predictively simulate unforeseen event of future and predict contribution rate in different
simulated events of the process. In our solution approach, we already presented details regarding the use of these
uncertainty factors.

6.3.8 DEFINING UNCERTAINTY RANGES AND DISTRIBUTIONS

We need to systematically vary uncertain factors in order to record corresponding behaviour of the response
factor. We need to establish a range for this variation and a distribution function that provides the probability of
certain value to occur within this range. Typically, analysis of legacy data or past knowledge of the process help to
determine such ranges and distribution. Due to process diversity in software context, legacy data or past knowledge
are hard to achieve. Instead of creating the burden of collecting additional information from past projects, we try to
make the maximum use of the observed data. To determine range and distribution, we utilize observed data points
and their corresponding values for selected uncertainty factors. Initially we have 35 observed data points in the
reference set. During collection of these 35 data points, 45 contributions were recorded. We collect corresponding
values for contribution size, contribution request arrival, collaborators, and collaborators skill level in respect of
these 45 contributions. These data help us to determine ranges and distribution for each uncertainty factor. Over time,
more contribution data are available. With more data the distribution calculation becomes more accurate and
represent the underlying software process better.
We defined our ranges and distribution for uncertainty factors using observed data points. We utilized a statistical fitting tool called “EasyFitExcel” to accomplish this action. This tool is capable of analyzing a set of data in respect of 20 different distributions and reports the best fitting distributions along with key parameter values. It presents an “goodness of fit” analysis using the Kolmogorov–Smirnov test (K–S test). This is a nonparametric test for the equality of continuous, one-dimensional probability distributions. It can compare a sample with a reference probability distribution (one-sample K–S test). This test quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution. The null distribution of this statistic is calculated under the null hypothesis that the samples are drawn from the reference distribution. The distributions considered under the null hypothesis are continuous distributions. We achieved a fitting distribution for each of our uncertainty factors using this tool. Uncertainty factors are varied within these given distribution during our predictive simulation trials. Distribution results for uncertainty factors i.e. which distribution was the best fit for each individual factors and key parameters of these distributions are presented in Table 8.

### Table 8 Selected distribution with key parameters for each uncertainty factor of case study-scenario 1

<table>
<thead>
<tr>
<th>Uncertainty Factor</th>
<th>Distribution Type</th>
<th>Key parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR Arrival</td>
<td>Gen Gamma</td>
<td>$k=0.88917, \sigma=0.40877, \beta=47.013, \gamma=0$</td>
</tr>
<tr>
<td>CR Size</td>
<td>Log Pearson 3</td>
<td>$\sigma=159.65, \beta=0.12014, \gamma=-16.062$</td>
</tr>
<tr>
<td>Collaborator skill level</td>
<td>Triangular</td>
<td>$m=1, a=0.36, b=15$</td>
</tr>
</tbody>
</table>

#### 6.3.9 Conducting Process Simulation

We consider completion time for contribution requests as the response factor in predictive simulation. Response factor depends on a set of uncertainty factors listed earlier in Section 6.3.7. Based on our observation we can provide imprecise and incomplete measures for these factors. We already calculated ranges and distribution for uncertainty factors and listed them in Table 8. Based on determined ranges and distributions we perform a work scheduling simulation. This simulation creates work schedule for available contribution requests while maintaining all constraints and dependencies. Due to lack of available data and limited scope, we could not consider all uncertainty factors. Therefore, we had to make assumptions for our simulation along with determining constant values for factors out-of-scope. All our assumptions are created based on the GitHub projects considered under SPC. Our major assumptions include:

- Total fifty contribution requests (CR) are considered for the predictive simulation.
- CR are prioritized based on their arrival (i.e. first come first serve).
- CR dependency information are not available.
- Two collaborators are available to perform merging (i.e. solve CR and merge it in main repository).
- Collaborators can only start a CR after it is formally arrived.
- Collaborators cannot multitask i.e. if a collaborator started a CR, she is responsible to complete it first.
- At most two contribution requests can be processed in parallel.
- Total 20 predicted contribution rate values are generated to plot on the control chart.
- Total 1000 simulation trials are performed.
- Predicted completion time $CT_1$ for certain contribution request $CR_1$ has 75% likelihood that $CR_1$ will complete by the time $CT_1$. 
Building work schedule simulation out of available contribution requests is accomplished through Monte Carlo simulation. Defined ranges and distribution specify variation of input parameters. Each simulation trial randomly considers values for input parameters within the defined range and distribution. Monte Carlo simulation randomly explores a set of work schedule in multiple simulation trials. Typically, a large number of \( t \) simulation trials are performed e.g. \( t=1000 \) trials in this case study. The work schedule is built for all contribution requests by assigning contribution requests to randomly chosen collaborators. At the end, we generated \( t \) different work schedules from \( t \) different simulation trials. We do not have any intention to find optimal solution out of this solution space. Instead, we would like to develop a distribution of completion time for each \( f \) contribution request by a certain time. We plot these predicted data points along with observed data points on the control chart.

6.3.10 ANOMALIES DETECTION AND INTERPRETATION

We attempt to detect anomalies in both observed and predicted set of values. Due to high variability in OSS development and lack of control on contributors, available contribution rate are unstable in most cases. Sometimes it is even hard to find 35 consecutive stable data points to utilize as reference set. In such case, we start using SPC with trial limits (i.e. temporary control limits) and as soon as more stable data is available we need to re-calculate our control limits.

To represent this challenge, we considered a dataset that initially presents instability in the process. We calculate trial control limits based on this dataset and readjust control limits later. We analyze our results in line with proposed challenges and research questions. Therefore four key issues analyzed in this report are -

- Analysis 0: Can we predictively generate future behavior of a selected process characteristic and plot it in a XmR control chart along with observed behavior of the process characteristic?
- Analysis 1: Can we use run tests to detect and interpret predicted anomalies using predicted behavior of the process characteristic?
- Analysis 2: Can we interpret detected anomalies in respect of both predicted and observed anomalies?
- Analysis 3: Can we validate predicted anomalies against real world anomalies?

As this is a post-mortem analysis of a developed project, we know the real world behaviour of selected project characteristics. Using this knowledge, we can verify whether predicted anomalies occurred in the real world process or not. If we predicted anomalies within time \( T1 \) to \( T2 \), we need to plot real world data within the same timeframe (i.e. \( T1 \) to \( T2 \)) to accomplish this verification. These real world observed data points (i.e. ODP) within timeframe \( T1 \) to \( T2 \) are plotted only for verification purpose. Therefore, they are denoted as verification data points (VDP). We plot VDP in the control chart in order to verify how accurately predicted anomalies represents future anomalies.

Analysis 0: We predicted completion time for contribution requests using predictive simulation. From these predicted values, we calculate the predicted contribution rate for next 20 days. In Figure 7, we plotted observed contributions rate for first 35 days (points 0-35) in the control chart. Along with this, we plotted next 20 days predicted contributions rate (points 36 to 56) in the control chart. Figure 7 presents that, we can predictively generate future behaviour of contribution rate and plot it in the same control chart along with observed data points. The presented control chart itself is an answer to the question asked in Analysis 0. Therefore, in our next scenarios we will not explicitly provide Analysis 0 section anymore.
**Analysis 1:** Predictive data points (PDP) are plotted in the control chart along with observed data points (ODP). All run test failures are marked using red ellipse in the control chart. We can detect eight (point 41 to 48) and seven (point 50 to 56) consecutive data points falling below the centerline. These patterns cause two RT4 (i.e. run above or below the CL) test failure in PDP. These are limit test failures and they strongly indicate that the process mean will change in future. The changing trend shows that CL value will become lower in future i.e. contribution rate will fall in future. This interpretation indicates to start an early investigation to find the assignable cause behind contribution rate fall. In this analysis, we utilized run tests and successfully detected anomalies in PDP and interpreted the results.

**Analysis 2:** Within the observed data points (ODP), point 12 to 20 and point 28 to 35 presents similar RT4 (i.e. run above or below the CL) test failure. These failures present strong indication that central mean value needs re-calculation. Predicted anomalies are in line with observed anomalies. As the control limits are not modified yet, predicted anomalies reconfirm the required change in CL value. Predicted anomalies also detect the trend of CL change. It suggests that the central mean will get lower i.e. the contribution rate will fall. We can find a clear link between observed and predicted anomalies. Observed and predicted anomalies support each other’s interpretation. Therefore, we can interpret anomalies with help of both predicted and observed anomalies. Due to evident support from observed anomalies we can consider immediate corrective action for these predicted anomalies.

**Analysis 3:** While predicting next 20 data points (i.e. PDP), we utilized distributions determined for uncertainty factors achieved from ODP. In house software development process typically follows a similar kind of distribution for uncertainty factors throughout the development process, unless there is an induced change in the process. For illustrative purpose, we consider a replication of above-mentioned scenario. Instead of considering real world data, we consider our VDP follow similar type of distribution (i.e. achieved from ODP) for the uncertainty factors. Based on these assumption, we will observe whether predicted anomalies can detect future anomalies or not. However, in OSS due to the development culture, similar distributions in uncertainty factors are rare to find. In scenario 4, 5, 6, and 7 we use real world data for VDP. In these scenarios, we compared our predicted anomalies against real world anomalies detected in VDP achieved from the GitHub project.
Figure 8 presents both PDP and ODP values along with VDP values for point 36 to 56. VDP and PDP data points are not exactly same as they differ in height data points. However, this variation does not influence the anomaly detection process. We can observe that RT4 failure areas are common in both PDP and VDP. This means, if uncertainty factors follow a similar distribution as in ODP, predictive anomalies can detect unforeseen future anomalies. Interpretation of predicted anomalies and VDP anomalies both suggests that the centerline should be re-calculated and have a lower value than earlier. The contribution rate will fall in future and requires investigation.

Our analysis in scenario 1 presents four conclusions i) it is possible to predict future behaviour of a process characteristic utilizing the available data observed in the process. ii) We can detect and interpret anomalies within predicted values using the run tests. iii) we can provide combined interpretation for anomalies based on predicted and observed anomalies. Predicted anomalies can reconfirm, or predict pattern for an observed anomaly. iv) predicted anomalies can predict future anomalies.

6.4 SCENARIO 2 (ILLUSTRATIVE EXAMPLE)

In Scenario 2, we consider experimenting with the same process considered in scenario 1. Therefore, data source, data collection method, data collection tools, control chart selection process, process characteristics selection, uncertainty factors and response factors are similar as scenario 1 and do not need a repeated discussion again. Scenario 2 is a continuation of scenario 1. We have observed data available for day 0 to day 56. Therefore, all PDP in scenario 1 are replaced with ODP in scenario 2. As we already received a more stable data set, we discard unstable data points and define a new reference set to re-calculate the control limits.

6.4.1 IDENTIFYING REFERENCE SET AND CONTROL LIMITS

Our initial reference set in scenario 1 presented signs of instability. In spite of RT1 and RT4 failures, we had to calculate our control limits based on this reference set and wait for a more stable data set. In scenario 1, we detected RT4 failure in both ODP and PDP. These anomalies suggested that the CL of the chart needs re-calculation as the process mean has been changed. It also predicts that the new process mean is lower than current CL. If control limits are not changed, we will have a wide process limit that encounters risk of missing anomalies. As soon as, we received a more stable dataset, we consider a new reference set and calculate our new control limits. Control limits calculated based on this new reference set (i.e. from day 21 to day 55) is provided in Table 9. This table present the control limits for X and mR control charts accordingly. In respect of the new control limits, this reference set is a stable data set. Therefore, we will not change these control limits until detected anomalies suggests a tuning action. We will continue to monitor our observed values using the prepared control chart utilizing run tests from Table 4 to detect anomalies in the software process.
6.4.2 DETERMINING UNCERTAINTY RANGES AND DISTRIBUTION

The underlying process for determining uncertainty ranges and distribution for uncertainty factors are same as scenario 1. With 15 day of additional observations, distribution and range value for uncertainty factors have changed. Due to more data points are available, calculated distribution should reflect the underlying data better. Calculated ranges and distributions for each of the uncertain input parameters are presented in Table 10 following.

<table>
<thead>
<tr>
<th>Uncertainty Factor</th>
<th>Distribution Type</th>
<th>Key parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR Arrival</td>
<td>Log Pearson3</td>
<td>$\sigma=10.515$, $\beta=-0.82737$, $\gamma=9.4808$</td>
</tr>
<tr>
<td>CR Size</td>
<td>Log Normal</td>
<td>$\sigma=1.9218$, $\beta=2.8631$, $\gamma=1.4312$</td>
</tr>
<tr>
<td>Collaborator skill level</td>
<td>Triangular</td>
<td>$m=1$, $a=0.36$, $b=15$</td>
</tr>
</tbody>
</table>

6.4.3 CONDUCTING PROCESS SIMULATION

We perform a predictive process simulation using Monte Carlo simulation technique similar as the one performed in scenario 1. Our focus is to predict future behaviour of Contribution rate. To achieve this purpose, we simulated completion time for each contribution request. We consider completion time as the response factor that depends on uncertainty factors stated earlier. This simulation creates work schedule for available contribution requests while maintaining all assumptions. Uncertainty factors, assumptions for this simulation are same as scenario 1 and already discussed earlier. Corresponding uncertainty values for these simulations are listed in Table 10. This simulation creates work schedule for available contribution requests while maintaining all assumptions stated in Section 6.3.9.

Due to illustrative purpose, we introduce changes in the process. We would like to observe if our simulation and detected anomalies react accordingly. We assume the company has introduced a new policy that allows more contributors to work on the project, thus higher number of contribution requests are received. The project received 30% higher number of CR than before. Simulation model is aware of this change in the process. We would like to see how it reacts to this change in predicting future behaviour of contribution rate. We perform predictive simulation to create PDP for day 56 to day 76. On the other hand, for illustrative purpose, we synthetically calculate the VDP for day 56 to day 76 reflecting the proposed change in process. We have 1000 trials that create a distribution of completion time for each CR under consideration. We only consider CR that completes within next 20 days (i.e. day 56 to day 76). We can find a certain completion time $CT$ that shows $X\%$ likelihood for $i^{th}$ contribution request $CR_i$ to solve by $CT$. Based on these values we calculate PDP for contribution rate in next 20 days. We plotted this PDP along with ODP in our control chart.

6.4.4 ANOMALIES DETECTION AND INTERPRETATION

As stated earlier, a process change has been performed synthetically in order to record the reaction of observed and predicted anomalies. Higher numbers of contributors increase the number of contribution requests by 30-50%. Our simulation is aware of this change and considered a 30-50% higher arrival rate while predictively simulating the process. We will analyze if this change is visible in ODP and PDP in the control chart. We perform discussion of our results based on four analysis strategy stated in Section 6.3.10. We plotted ODP and PDP both in Figure 9. This control chart itself already address Analysis 0. Other three analyses are presented below.
Analysis 1: In PDP we can find 15 data points (point 56 to 71) consecutively observed within the zone $\pm 1\sigma$ area of the control chart. This behaviour causes RT6 run test failure (i.e. the stratification failure). From our earlier discussion in Section 5.4.4, we know that stratification test failure arise from an induced change in process variability that has not been properly accounted in the X chart control limits. RT6 failure indicates presence of an assignable cause. It can also positively mean that the process behaviour is becoming more stable and predictable after the induced change happened.

The predicted anomaly of this example indicates towards our induced change. We have made a process change that increased the contribution request arrival by 30 to 50%. The control limits are not yet accounted with this process change. Stratification test failure in PDP indicates the presence of an assignable cause in the process. It also indicates, a process change has occurred and it is not yet accounted in control limits. Both of these indications are expected and in line with our performed change in the process.

Analysis 2: We generated new control limits based on a stable reference set. Therefore, we cannot find anomaly in ODP (i.e. the reference set). Our predicted anomaly is not preceded by any observed anomaly this time. However, predicted anomaly reconfirms the influence of an induced change performed on the process. The anomaly is expected, due to this known change applied on the process. Behaviour of the anomaly helps us to understand the influence of the change. If no known induced change is available, this failure in PDP will indicate towards an unknown assignable cause and requires investigation.

Analysis 3: Next, we plot our VDP values for day 56 to 76. VDP points resemble with plotted PDP points. Both of these data presents a RT6 (stratification) test failure at the same area. They both indicate the presence of an assignable cause in the process and requests for its reflection on the control limits. Thus, we can conclude that predicted values are capable of detecting anomalies using run tests. Predicted values can predictively detect and interpret future anomalies earlier than they occur.
6.5 SCENARIO 3

We will perform our case study on three real world scenarios considering the same project used in scenario 1. As we continued working on the same project our data source, data collection method, data collection tools, control chart selection process, process characteristics selection, uncertainty factors and response factors and assumptions are similar with the presented discussion in scenario 1. Earlier due to illustrative purpose, we synthetically induced changes in our project data. In following three scenarios, we do not perform any synthesized change on the dataset. We conduct our analysis based on real data achieved from GitHub repository. In scenario 3 we consider our reference set consist of day 0 to day 36. Selected data for the initial reference set is presented in Table 11. Control limits are calculated based on the selected reference set and presented in Table 12 below.

Table 11 Initial Reference Set values for case study-scenario 3

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>17</td>
<td>0</td>
<td>25</td>
<td>0</td>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>10</td>
<td>3</td>
<td>18</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>34</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>11</td>
<td>10</td>
<td>19</td>
<td>0</td>
<td>27</td>
<td>4</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>12</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>36</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>21</td>
<td>2</td>
<td>29</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>22</td>
<td>2</td>
<td>30</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>15</td>
<td>1</td>
<td>23</td>
<td>0</td>
<td>31</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>16</td>
<td>1</td>
<td>24</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 12 Control limits in respect of the reference set for Case Study-Scenario 3

<table>
<thead>
<tr>
<th>X Chart</th>
<th>mR Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCL_X (3σ)</td>
<td>Zone 2σ</td>
</tr>
<tr>
<td>2.62</td>
<td>1.89</td>
</tr>
</tbody>
</table>

Figure 10 Control chart for Case Study-Scenario 2 (showing ODP, PDP, VDP)
At day 36, we want to predict probable contribution rate for next 20 days (i.e. from day 37 to day 56). We determine ranges and distribution for our uncertainty factors utilizing the process discussed in Section 6.3.8. All these values are listed in Table 13 below. Based on these ranges and distributions we vary our uncertainty factors in the predictive simulation to achieve a distribution for completion time of each contribution request. The simulation process is discussed in detail earlier in section 6.3.9.

Table 13 Selected distribution with key parameters for each uncertainty factor of Case Study-Scenario 3

<table>
<thead>
<tr>
<th>Uncertainty Factor</th>
<th>Distribution Type</th>
<th>Key parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR Arrival</td>
<td>Gen Gamma</td>
<td>$k=0.88917, \sigma=0.40877, \beta=47.013, \gamma=0$</td>
</tr>
<tr>
<td>CR Size</td>
<td>Log Pearson 3</td>
<td>$\sigma=159.65, \beta=0.12014, \gamma=-16.062$</td>
</tr>
<tr>
<td>Collaborator skill level</td>
<td>Triangular</td>
<td>$m=1, a=0.36, b=15$</td>
</tr>
</tbody>
</table>

We perform predictive simulation to create PDP for day 37 to 56. 1000 simulation trials create distribution of completion time for each control request. We will consider those CR that completes within next 20 days (i.e. day 37 to day 56). From these completion time distribution, we can find certain completion time $CT$ that shows 75% likelihood for $f^{th}$ contribution request $CR_f$ to solve by $CT$. Based on these values we predict our PDP and plot them on control chart visible in Figure 11 below.

![Figure 11 Control chart for Case Study-Scenario 3 (showing ODP, PDP)](image)

6.5.1 RESULT ANALYSIS

As discussed earlier, we analyze our case study results in line with our proposed challenges and research questions. We follow analysis guidelines as presented in scenario 1. By predicting PDP and plotting them in the control chart we already shown that prediction was possible based on available data sets (Analysis 0). Next three analysis are presented below -

Analysis 1: Analyzing PDP points we found that two RT4 (run below/above the CL) failures are visible in point 37 to 44 and point 50 to 56. These limit test failures indicate towards changes in the process mean. These two failures provide a strong evidence that CL value will become lower than current CL value. These anomalies indicate to start an early cause investigation to find the assignable cause behind them. It will help us to know why contribution rate will fall in future and take corrective action to prevent this anomaly from occurring.

Analysis 2: Observed data points (ODP) presents two RT4 (i.e. run above or below the CL) test failure. Observed anomaly indicates shift in process means and suggest re-calculating the control limits. Predicted anomalies reconfirm
observed anomalies. Predicted anomalies detected the same trend as the control limits have not been changed yet. Both suggest that, contribution rate will fall in future. This is alarming for the project and requires early investigation. Predicted anomalies helped to reconfirm observed anomaly here and to early detect the trend of the observed anomaly. We can start early investigation to prevent this anomaly or minimize its influence on the process.

![Control chart for Case Study-Scenario 3 (showing ODP, PDP, VDP)](image)

**Analysis 3:** In this analysis, we are interested to find out whether our predicted anomalies can detect future anomalies or not. We compare our predicted anomalies with real observation data available from GitHub (Figure 12). At day 36, we predicted PDP for day 37 to day 56. As we are working with post-mortem of a development project, we have real observations available for day 37 to day 56. These data points are collected from Github repository using the similar data collection process stated earlier. We refer to these values as VDP. We detect anomalies in VDP and compare them with our predicted anomalies.

VDP and PDP data points are not same. VDP values vary in different points compared to PDP values. However, these variations do not influence anomalies detection process. Both RT4 failures predicted in PDP are also visible in VDP points. VDP and PDP both suggest that CL will shift to lower value in future. If we initiate an early investigation based on predicted anomaly, it can prevent the anomaly from occurring or minimize its influence on the process.

### 6.6 SCENARIO 4

Scenario 4 is a continuation of scenario 3. Scenario 3 considers the initial reference set (i.e. observations from day 0 to 36) and predicts data points for day 37 to 56. In scenario 4, observation data ODP are available from day 0 to 56. All predicted data in scenario 3 are replaced with ODP in scenario 4. In scenario 4 we utilize predictive simulation approach to predict data points for next 20 days (i.e. day 57 to 76) and plot them on the control chart along with ODP. Predicted data points will help us to predict future anomalies and take early corrective action if required.

Data source, data collection method, data collection tools, control chart selection process, process characteristics selection, uncertainty factors and response factors and assumptions are similar with the presented discussion in scenario 1 (section 6.3). Therefore, we do not discuss them in details again. However, ranges and distribution for uncertainty factors may become different due to growth in observed values. Thus, we re-calculate ranges and distribution again to have a better representation of underlying process. Distributions and ranges respective to each uncertainty factor are calculated using the EasyFitExcel tool (details in Section 6.3.8). Achieved range and distribution for uncertain variables are presented in Table 14 below.
Table 14 Selected distribution with key parameters for each uncertainty factor of Case Study-Scenario 4

<table>
<thead>
<tr>
<th>Uncertainty Factor</th>
<th>Distribution Type</th>
<th>Key parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR Arrival</td>
<td>Kumaraswamy</td>
<td>$\sigma_1=0.32\ ,\ \sigma_2=0.71\ ,\ a=0.00333\ ,\ b=191.1$</td>
</tr>
<tr>
<td>CR Size</td>
<td>Fatigue life 3P</td>
<td>$\sigma=1.9487\ ,\ \beta=24.22\ ,\ \gamma=2.0216$</td>
</tr>
<tr>
<td>Collaborator skill level</td>
<td>Triangular</td>
<td>$m=1\ ,\ a=0.36\ ,\ b=15$</td>
</tr>
</tbody>
</table>

Initial reference set was unstable. Therefore, as soon as we receive a new stable dataset, we need to re-calculate our control limits. Updated dataset offers a more stable reference set within point 21 to 56. Based on this new reference set we re-calculate our control limits as presented in Table 15 below. After we gather all these required data, we conduct predictive simulation following the simulation guidelines presented in section 6.3.9. PDP along with ODP are plotted in Figure 13. We analyze our control chart with three analysis strategy presented in scenario 1.

Table 15 Control limits in respect of the Reference Set for Case Study-Scenario 4

<table>
<thead>
<tr>
<th></th>
<th>X Chart</th>
<th>mR Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UCL$_X$ (3$\sigma$)</td>
<td>Zone 2$\sigma$</td>
</tr>
<tr>
<td></td>
<td>1.83</td>
<td>1.38</td>
</tr>
</tbody>
</table>

Figure 13 Control chart for Case Study-Scenario 4 (showing ODP, PDP)

6.6.1 RESULT ANALYSIS

Analysis 1: Within PDP, we find more than eight consecutive data points on either side of the centerline avoiding the zone1sigma area. This pattern refers to a RT5 run test failure (i.e. mixing/over control test). This failure means increasing variability. Typically, this failure occurs after an induced improvement has occurred and continues until the improvement is fully acquired by the developers or organization.

This failure requires an early investigation if no known improvement has performed on the process. In such case, an RT5 failure is unexpected and must be a result of unknown assignable cause present in the process. Early investigation can find out the cause and help to prevent it. On the other hand, if a known induced improvement has occurred, and RT5 failure is expected, no additional investigation is required.
**Analysis 2:** Detected anomaly within predicted data points are in line with observed anomalies. Within observed data points, we can detect two RT1 (i.e. three sigma) failures at point 36 and 44. RT1 failures are an early indication of changes that will occur in future. However, it cannot initiate investigation until further observations are available. Predicted anomalies help us to understand the type of possible anomaly these RT1 failures refers to and initiate early investigation to prevent it. For example, in scenario 4 the RT1 is an indication towards an RT5 (i.e. mix/over control) failure and we can initiate cause investigation for this failure type.

We are working with post mortem of a past project and we don’t have any ability to initiate a cause investigation for RT5 failure or prevent it. However, looking at resources like network graph, and discussion forums we found that development within this timeframe was highly derived by releases. Within 20 days, this project faced two consecutive releases. This might have facilitated this RT5 failure.

![Figure 14 Control chart for Case Study-Scenario 4 (showing ODP, PDP, VDP)](image)

**Analysis 3:** In Figure 14 we plotted both PDP and VDP for day 57 to 76. As this is a post mortem study, we already knew the contribution rate from day 56 to 75 in real world. We will compare our predicted anomalies with VDP anomalies here. Though VDP presents a completely different dataset compared to PDP, they have similarity in detected anomalies. VDP also detects two RT5 failures within the given timeframe. VDP values point the same areas that PDP considered responsible for RT5 failure. This analysis presents that predicted values can well detect future anomalies. Early investigation can help us to prevent this anomaly or minimize its influence on the process. Additionally, RT5 failure suggests to find a new control limit to control underlying data. For this action, we need to wait for real values to be obtained. We already stated earlier that, tuning action should be performed only based on observed values.

**6.7 SCENARIO 5**

Scenario 5 is a continuation of scenario 3 and 4. In scenario 5, observation data ODP are available from day 0 to 76. We replaced all PDP data of scenario 4 with ODP. We utilize predictive simulation approach to predict data points for next 20 days (i.e. day 77 to 96) and plot them on the control chart along with ODP. Predicted data points will help us to predict future anomalies and take early corrective action if required.

Data source, data collection method, data collection tools, control chart selection process, process characteristics selection, uncertainty factors and response factors and assumptions are similar with the presented discussion in scenario 1 (section 6.3). Therefore, we do not discuss them in details again. However, ranges and distribution for
uncertainty factors may become different due to growth in observed values. Thus, we re-calculate ranges and distribution again. Distributions and ranges respective to each uncertainty factor are presented below in Table 16

Table 16 Selected distribution with key parameters for each uncertainty factor of Case Study-Scenario 5

<table>
<thead>
<tr>
<th>Uncertainty Factor</th>
<th>Distribution Type</th>
<th>Key parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR Arrival</td>
<td>Power Function</td>
<td>$\sigma=0.24878, a=0.00333, b=78.31$</td>
</tr>
<tr>
<td>CR Size</td>
<td>Log Normal 3P</td>
<td>$\sigma=1.7452, \mu=3.0343, \gamma=2.7874$</td>
</tr>
<tr>
<td>Collaborator skill level</td>
<td>Triangular</td>
<td>$m=1, a=0.36, b=15$</td>
</tr>
</tbody>
</table>

As suggested in scenario 4, we need to re-calculate our control limits. Until new observations are available, we cannot reflect the mix/overcontrol failure in our control limit re-calculation. However, it is possible to discard two RT1 failure points (i.e. point 27, 36) from our reference set. It will help to achieve a better control limit. Thus, we re-calculate our control limits considering new reference set as point 42 to 76. Updated control limits are presented in Table 17 below. After all required data are gathered we generated our predicted values by using the predictive simulation and analyze them on control chart.

Table 17 Control limits in respect of the Reference Set for Case Study-Scenario 5

<table>
<thead>
<tr>
<th>X Chart</th>
<th>mR Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCL$_X$ (3$\sigma$)</td>
<td>Zone 2$\sigma$</td>
</tr>
<tr>
<td>1.15</td>
<td>0.86</td>
</tr>
</tbody>
</table>

Figure 15 Control chart for Case Study-Scenario 5 (showing ODP, PDP)

6.7.1 RESULT ANALYSIS

Analysis 1: Analyzing PDP from point 77 to 96 (Figure 15), we can find two RT2 (two sigma test) failure. In both case two among three consecutive points fall out of two sigma area. Both of these refers to presence of an assignable cause and requires cause investigation. We will not initiate any cause investigation at this point, as two sigma failure does not provide clear understanding on what type of trend failure analysis we should investigate. We need to wait for further observations.
Analysis 2: If we analyze ODP, we can find an indication of RT4 (run below the control line) failure within points 64 to 70. This indicates, the centerline has been changed and we need to recalculate our centerline based on the changed process mean. Unfortunately, our predicted anomalies are not in line with the change detected in the observed values. We cannot create any interpretation from our predicted anomalies based on these observed anomalies.

Figure 16 Control chart for Case Study-Scenario 5 (showing ODP, PDP, VDP)

Analysis 3: If we plot VDP for data point 77 to 96 in the control chart, we can find two three sigma failures occurred in (Figure 16). Both of these three sigma failures are indicated earlier by two sigma failures in PDP. These predicted anomalies are in line with VDP anomalies. However, in VDP we also detect one RT4 (run below centerline failure) failure. This is a continuation of the observed RT4 failure. Unfortunately, our predicted data was not capable to detect this RT4 failure. Thus though VDP and ODP both suggest change in centerline, PDP does not suggest anything. It only indicates two random failures that occurred in reality as RT1 failure.

6.8 SCENARIO 6

6.8.1 SCENARIO DATA

In this scenario we considered a different project “NuGetDocs”[53] from the same data source i.e. GitHub OSS repository. NuGet is a visual studio extension. It focuses on adding, removing and updating libraries and tools in visual studio projects. NuGetDocs lists documentation to help use of NuGet packages. NuGetDocs is a different type of project compared to “octokit.rb”, as the major focus of this project is to develop software documentation for a software development project (i.e. NuGet). We primarily wanted to investigate, if we can apply our proposed approach out of the scope of development and testing contributions. Thus, we selected NuGetDocs project that focuses on software documentation of a developing software project.

NuGetDocs managed its revision history using Git. It is an open source public repository available online in GitHub. This project started on Oct 23, 2011. It has 492 commits, 160 issues and, 57 contributors. 121 issues are already solved and 39 issues are still open. This project is from the same data source (i.e. GitHub OSS repository) used in earlier scenarios. Similar to previous scenarios, we consider the same process characteristic (i.e. contribution rate) to monitor and control under SPC. Thus, data collection method, data collection tools, control chart selection process, process characteristics selection, uncertainty factors, response factors and assumptions are similar with
initially presented discussion in scenario 1 (Section 6.3). We will not discuss details of these steps again for scenario 6.

### 6.8.2 IDENTIFYING REFERENCE SET AND CONTROL LIMITS

For our study, we consider development between 29 May, 2012 to 10 April 2013 of “NuGetDocs” project. The growth of NuGetDocs project is comparatively slower than “octokit.rb”. Therefore, we slightly change the calculation of contribution rate. Contribution rate is considered as the number of contributions merged in the main repository each 100 hours. This change is made in order to avoid too much null contribution data points on the control chart. We consider first 34 observed data points as our reference set. Reference set data are presented in Table 18 below.

**Table 18 Initial Reference Set values for Case Study-Scenario 6**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>9</td>
<td>1</td>
<td>17</td>
<td>0</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>18</td>
<td>1</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>19</td>
<td>0</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>12</td>
<td>1</td>
<td>20</td>
<td>0</td>
<td>28</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>13</td>
<td>2</td>
<td>21</td>
<td>0</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>14</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>16</td>
<td>1</td>
<td>24</td>
<td>0</td>
<td>32</td>
<td>1</td>
</tr>
</tbody>
</table>

We calculate our control limits based on this reference set using equations presented in Table 3. Control limit values for X and mR charts are listed below in Table 19. We plotted the reference set data in the control chart. We detected a RT1 (three sigma test) failure in the reference set. Thus, we consider these control limits as trial limits and as soon as we receive stable dataset we will select new reference set and tune control limits. After control limits are determined, we can observe the process continuously. We plot observed data points on the control chart and test it with run tests presented in Table 4. These tests detect anomalies in the software process.

**Table 19 Control limits in respect of the Reference Set for Case Study-Scenario 6**

<table>
<thead>
<tr>
<th>X Chart</th>
<th>mR Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCLX (3σ)</td>
<td>CLmR</td>
</tr>
<tr>
<td>Zone 2σ</td>
<td>UCLmR</td>
</tr>
<tr>
<td>Zone σ</td>
<td>.85</td>
</tr>
<tr>
<td>.5</td>
<td>2.77</td>
</tr>
</tbody>
</table>

### 6.8.3 DEFINING UNCERTAINTY RANGES, DISTRIBUTIONS AND CONDUCTING SIMULATION

The underlying process for determining uncertainty ranges and distribution for uncertainty factors are same as scenario 1. However, as we selected a new reference set and a new project, all related observations for uncertainty factors are different. Thus, we need to re-calculate ranges and distribution again to represent the underlying uncertainties. Distributions and ranges respective to each uncertainty factor are calculated using the EasyFitExcel tool. Achieved range and distribution for uncertain variables are presented in Table 20 below.
Table 20 Selected distribution with key parameters for each uncertainty factor of Case Study-Scenario 6

<table>
<thead>
<tr>
<th>Uncertainty Factor</th>
<th>Distribution Type</th>
<th>Key parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR Arrival</td>
<td>Gen Pareto</td>
<td>$k=276.41$, $\sigma=-33.512$, $\mu=-33.512$</td>
</tr>
<tr>
<td>CR Size</td>
<td>Fatigue life 3P</td>
<td>$\sigma=0.75106$, $\beta=6.4047$, $\gamma=0.82818$</td>
</tr>
<tr>
<td>Collaborator skill level</td>
<td>Triangular</td>
<td>$m=450$, $a=1$, $b=2000$</td>
</tr>
</tbody>
</table>

We perform a predictive simulation to create PDP for point 35 to 54. 1000 simulation trials will create distribution of completion time for each contribution request. We will consider only those CR that completes within next 20 data points. From these completion time distributions, we can find certain completion time $CT$ that shows $75\%$ likelihood for $f^\text{th}$ contribution request $CR_f$ to solve by time $CT$. Based on these values we predict our PDP and plot them on control chart visible in Figure 17 below.

![Figure 17 Control chart for Case Study-Scenario 6 (showing ODP, PDP)](image)

**Analysis 1:** Two run test failures are visible within predicted data points. From data point 35 to 53, 18 consecutive values are present within the sigma 1 area. It causes RT6 (i.e. stratification test) failure. This failure indicates reduced variability in the process. RT6 failure occurs when an induced improvement has occurred in the process and the process variability has not been properly accounted in the control limits. However, it also indicates that process behaviour became more predictable compared to earlier process behaviour.

From data point 39 to 53, we can find 15 consecutive points alternating up and down around the centerline. This pattern causes RT7 (i.e. Oscillatory trend test) failure. This failure indicates that two systematically alternating causes are creating different results. Both of these failures indicate presence of an assignable cause in the process. We should initiate investigation to isolate alternating causes responsible for different results and find induced improvements that cause the stratification failure.

**Analysis 2:** We can find RT1 (three sigma test) failure at point 6 within observed data points. This is an early indication about future changes in the software process. From point 14 to 27 we can find 14 consecutive values within the sigma1 area. RT6 failure requires 15 consecutive points within Sigma1 area. Theoretically, we cannot claim it as RT6 (stratification test) failure. However, we will consider it as an indication towards future stratification failure.
Two failures in our predicted data points are partially in line with observed failures. Three-sigma test failure indicates a future process change. 14 consecutive points in sigma 1 area provides a hint for future stratification failure. We can find the stratification failure in our predicted data. However, there was no related failure to oscillatory trend test in observed value that can help us to interpret the predicted RT7 failure.

![Control chart for Case Study-Scenario 6](image)

Figure 18 Control chart for Case Study-Scenario 6 (showing ODP, PDP, VDP)

*Analysis 3:* We are working with a post mortem project. Thus, we already have observations available for point 35 to 55. We plot these data to see how accurately predicted data finds possible anomalies. Both predicted and future data are plotted in Figure 18. If we look at the observed data we can find from point 33 to 47, 15 consecutive values are within sigma 1 area. This clearly indicates RT6 (stratification test) failure that resembles with our predicted failure. In fact, predicted data (i.e. 35 to 47) helped to early detect stratification failure in the process.

However, observed value do not have any failure directly related to the oscillatory trend failure. Three sigma failure at point 48 is an early indication of change and a non-significant failure. The oscillatory trend failure indicates the presence of an assignable cause and requires investigation.

### 6.9 SCENARIO 7

Scenario 7 is a continuation of scenario 6. In this scenario, we consider observation data are available for point 0 to point 54. All predicted data in scenario 6 are replaced with ODP in scenario 7. We generate PDP for next 20 points i.e. point 55 to 74. Predicted data points will help us to predict future anomalies and take early corrective action if required. We don’t discuss data collection method, data collection tools, control chart selection process, process characteristics selection, uncertainty factors and response factors and assumptions in details again as they are similar with the presented discussion in scenario 1 (section 6.3). However, as we observed new data, ranges and distribution for uncertainty factors may became different. We re-calculate ranges and distribution again to have a better representation of the underlying data. New range and distribution for uncertain variables are presented in Table 21 below.
Table 21 Selected distribution with key parameters for each uncertainty factor of Case Study-Scenario 7

<table>
<thead>
<tr>
<th>Uncertainty Factor</th>
<th>Distribution Type</th>
<th>Key parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR Arrival</td>
<td>Gen Extreme value</td>
<td>$k=0.00673$, $\sigma=121.08$, $\mu=120.51$</td>
</tr>
<tr>
<td>CR Size</td>
<td>Burr</td>
<td>$k=0.40159$, $\sigma=2.3902$, $\beta=6.6041$, $\gamma=0$</td>
</tr>
<tr>
<td>Collaborator skill level</td>
<td>Triangular</td>
<td>$m=450$, $a=1$, $b=2000$</td>
</tr>
</tbody>
</table>

Initial reference set was unstable due to a RT1 failure in this dataset. Later we received more stable data points. Thus, we select a new reference set (i.e. point 7 to point 43). Based on this new reference set we re-calculate our control limits as presented in Table 22 below. After we gather all these required data, we conduct predictive simulation following the simulation guidelines presented in Section 6.3.9. PDP along with ODP are plotted in Figure 19. We analyze our control chart with three-analysis strategy presented in scenario 1.

Table 22 Control limits in respect of the Reference Set for Case Study-Scenario 7

<table>
<thead>
<tr>
<th>X Chart</th>
<th>mR Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCL $X$ (3σ)</td>
<td>Zone 2σ</td>
</tr>
<tr>
<td>2.33</td>
<td>1.69</td>
</tr>
</tbody>
</table>

Figure 19 Control chart for Case Study-Scenario 7 (showing ODP, PDP)

6.9.1 RESULT ANALYSIS

Analysis 1: In scenario 7, we found RT6 (stratification test) failure within point 55 to 75 in PDP. This failure indicates an induced change in software process that has not been properly accounted in the control chart and fully acquired by the developers and organization. However, it also indicates a positive note that the variability of the process is reducing over time.

In scenario 6, a stratification failure was already detected. It indicated the importance to initiate a cause investigation to detect the assignable cause. As this is a post mortem analysis of a developed project, we cannot practically initiate a cause investigation process and remove the assignable cause. Therefore, in scenario 7, the
assignable cause still exists and we receive the stratification failure again. This failure indicates to initialize a cause investigation process.

**Analysis 2:** We can find an indication of stratification failure in the observed data as well. From data point 29 to 47, all 18 points are plotted within sigma 1 area and caused RT6 (stratification test) failure. This provides strong evidence for an induced change in the process that has created variability. Over time, the project will become more predictable. Our predicted failure is in line with this observed failure. However, though indication claims increasing predictability of the process, some random three sigma test failures still occurs. For example, at point 48 we can again find a three sigma test failure within the observed data points.

![Control chart for Case Study-Scenario 7 (showing ODP, PDP, VDP)](image)

**Analysis 3:** If we plot the real world data for data points 56 to 75, we can find three individual RT1 (three sigma test) failures. All other data points (17 random points) remain within the sigma 1 area. As they are non-consecutive data points, run test failure does not occur. However, they indicate towards an increasing predictability of the process. Three random RT1 failures indicate that expected stability is not yet achieved. In software context single values falling out of control limits may point towards assignable cause but they are of less importance.

### 7. THREATS TO VALIDITY

*Construct validity* of a study refers to two issues: i) how well concepts are defined in the study, and ii) how well these concepts are referred in the measures utilized in the study. This study is an abstraction of the real world. All underlying assumptions impose threats to validity of this study. In this study, we simulate a work-scheduling problem with Monte Carlo simulation. Selected response factor is the contribution rate of the project. Contribution rate is a significant measure for OSS project as it provides a clear indication about the projects growth. We consider our response factor dependent on three uncertainty factors i.e. CR arrival, CR Size and Collaborators skill level. These three factors are selected based on development culture and data availability of GitHub projects. Significance of these factors are primarily confirmed by GitHub as they are responsible for uncertainty in GitHub development context. Due to lack of available data and limited scope, we cannot consider all uncertainty factors possible. Consideration of more uncertainty factors will help us to increase validity of the study.
Uncertainties of these factors are defined using Experience Database. This is a reliable process to determine uncertainty. However, due to lack of legacy projects and past knowledge we had to rely on a small dataset to fit our distributions and define uncertainties. The available dataset grew over time, but initial results of this approach might not accurately representative to the underlying uncertainty.

Collection of these underlying dataset suffers from assumptions. Due to data unavailability, we had to limit our contribution size calculation only to the code merged into the main repository. Collaborators are responsible for merging effort in GitHub. Size of the merged code (in LOC) is considered as a representation of the required merging effort. However, size is not necessarily the best representation of the effort requirement.

We considered merging effort in our study. Merging effort is one of the significant type of efforts related to the contributions in main repository. GitHub does not exactly represent the development effort spent by contributors in their personal repositories. Thus, we considered development effort out of our scope. Consideration of development effort will help to better replicate the project development process.

Our work scheduling simulation is performed utilizing a simple scheduling method. We had to restrict this simulation with assumptions in order to replicate GitHub development culture and available data. For example, CR are prioritized based on their arrival time, we do not accept multitasking, CR dependency information is not available etc. These assumptions impose threats to validity of our study.

Internal validity is concerned about how a study warrants the causal conclusion of the study. This kind of validity primarily depends on minimizing biases or systematic error. Predictive simulation constitutes a major part of our proposed approach that predicts future behaviour of selected process characteristics. As our predictive simulation relies on computerized simulation model, internal validity is easy to maximize. Internal validity asks for treatment-outcome construct and our study already performs such actions. We considered varying values for uncertain parameters in order to check different behaviour of the response parameter in this study.

External validity is concerned about what extent the study findings are applicable in different settings other than the local settings they are generated in. Typically, we can maximize external validity by replication of the study. In our case study external validity is limited. We experimented with GitHub OSS repository. Thus, our underlying assumptions reflect traditional development culture of GitHub OSS project. Our study is dependent on a predictive simulation model. This simulation considers uncertainty and response factors based on the OSS development context (in GitHub). We can replicate our study with different projects that use similar development context (i.e. projects available in GitHub). Replicating our study with different set of uncertainty and response factors will help us to increase the external validity of our study.

8. DISCUSSION AND CONCLUSION

8.1 DISCUSSION

In this report, we studied seven scenarios using Open Source Software (OSS) projects from GitHub repository. Two mid-size projects from GitHub OSS repository (i.e. octokit.rb and NuGetDocs) were selected for this purpose. In first two scenarios, collected data from GitHub were synthetically manipulated for illustrative purpose. These studies allowed us to observe and validate the predicted anomalies (i.e. generated using proposed approach) in respect of known changes made to the process and expected anomalies. Next five scenarios are performed using real data collected from GitHub repository. As this is a postmortem study of developed projects, real world behavior of the process is available for the timeframe we predicted process behavior. In our earlier analysis, we denote these real world behavior data as VDP. We validated predicted anomalies against real world observed anomalies (i.e. VDP anomalies).

To address the objective of this research, we proposed three challenges (listed in section 4.5) and investigated corresponding three Research Questions. Our case study results should answer our research questions. We proposed four analysis questions in Section 6.3.10 to guide analysis of our results in this direction. Results from each scenario are individually analyzed earlier, in respect of these analysis questions. In this section, we aggregate these analyses to answer our research questions. This report partially validates our proposed approach and our objective. However, a more detailed analysis of the proposed approach is still outstanding.
8.1.1 DISCUSSION OF RESULTS IN RESPECT OF RQ1

In this study, “contribution rate” is the selected process characteristics monitored and controlled using the proposed solution approach. We applied a work-scheduling simulation to predictively determine the process behavior (i.e. next 20 data points in the control chart). In consideration of varying inputs in three uncertainty factors (i.e. contribution rate, contribution size and collaborators skill level) we recorded varying output values for the response factor (i.e. contribution completion time). Output values present a probable distribution (i.e. predicted behavior) of contribution completion time. Contribution rate is calculated from these values and monitored against the control limits. Predicted anomalies are detected if predicted values fail any of the run tests.

In all seven scenarios, we generated predicted values for the process characteristic. We utilized experience database based technique to determine ranges and distribution of uncertainty factors. Instead of creating burden of collecting past knowledge or legacy data, we considered available observed value as experience database. Small size of the experience dataset may affect distribution calculation. Predicted values do not exactly match VDP values. However, they present similar patterns. Thus in six out of seven scenarios similar anomalies were detected in both predicted and VDP values.

In all seven scenarios, predicted values were generated using available observed values. Therefore, in lack of past knowledge and legacy data, it is possible to generate predicted behavior of selected process characteristic using observed dataset only. We can plot these predicted values on the XmR chart along with the observed values. We can use predicted values to detect predicted anomalies as well. However, accuracy of this prediction completely depends on the underlying prediction model and available data. In this study, we utilized a simple prediction model. Comprehensive evaluation of our proposed approach using better prediction models is still outstanding.

8.1.2 DISCUSSION OF RESULTS IN RESPECT OF RQ2

In our case study scenarios, we applied Run Tests (discussed in section 5.3) to detect anomalies in the software process. Run tests were applied to both observed and predicted behavior of the process. In RQ2, the concern is to investigate, whether or not run tests are capable of detecting and interpreting anomalies within predicted data. In this report, run tests detected anomalies using predicted data in all seven scenarios. To verify these anomalies, we compared predicted anomalies with VDP anomalies. We detected 11 predicted anomalies in all our scenarios. In eight cases, similar anomalies were detected in VDP data as well. In three cases where the predicted anomalies do not match VDP anomalies is due to wrong prediction about process behavior. However, these detected anomalies were meaningful as well. Therefore, we can conclude that run tests can successfully detect anomalies using both predicted and observed behavior of the process. However, due to high variability in nature and low trustworthiness of predicted behavior, specific patterns in observation values i.e. trend and limit tests are only considered as significant type of run tests. Sigma tests are considered as less important and does not require investigation if detected in isolation.

8.1.3 DISCUSSION OF RESULTS IN RESPECT OF RQ3

In presence of predicted values, we no longer provide isolated interpretation of observed anomalies. Instead, we consider the influence of observed anomalies on predicted anomalies and vice versa. This helps us to better interpret detected anomalies, and provides more information to investigate assignable causes. These interpretations follow our proposed guidelines in Table 5 and help us to early detect anomalies or understand their behaviour. Therefore, we can initiate actions to prevent these anomalies or minimize their influence on process performance.

We provided combined interpretation for all cases where we detected predicted anomaly preceded by observed anomaly. For example, if similar run test failure occurs in both observed and predicted data points, predicted anomalies confirm the effect of observed anomalies. If an assignable cause is present in observed value, it continues to influence the process until it is removed. Therefore, similar run test failure occurs in the predicted behavior and reconfirms the future influence of the assignable cause. For example, in scenario 1, 3, and 7 we can find similar observed anomaly preceded the predicted anomaly. In our analysis we described this behavior as a reconfirmation of the observed anomaly. We need to initiate investigation to remove this assignable cause.

If a predicted anomaly is preceded by different type of observed anomaly, we can interpret them with aid of the proposed guideline in Table 5. In such cases, predicted anomalies help to early detect the type and influence of an observed anomaly. Based on this information we may or may not need to initiate an early investigation. For example, in scenario 4 an RT4 (Run above/below CL test) failure is preceded by RT1 (three sigma test) failure. Typically, no investigation is initiated due to RT1 failures. We need to wait for more observation as the type of anomaly is unclear.
However, in this scenario predicted anomaly RT4 early indicates the type of failure. Therefore, we can initiate early investigation to prevent the anomaly.

If predicted anomaly is not preceded by any observed anomaly, it indicates presence of an assignable cause. The anomaly is initiated by an assignable cause in the process which is unknown and possibly undesirable. Therefore, an early investigation will help to find the cause and prevent the anomaly from occurring. For example in scenario 2 we can find an RT6 failure in the predicted anomaly. This failure happened due to an induced change in the process.

### 8.2 CONCLUSION

Statistical Process Control (SPC) has recently gained popularity in software process monitoring and control. Due to special characteristics of software process, we need to customize SPC before applying it in software context. Available literature focused on four key challenges in this regard. SPC can reactively detect software process anomalies. Software process presents high variability and dynamism in nature. Therefore, reactive detection and solution of process anomalies can be risky and costly. In this research, our objective is to enhance the capability of a process monitoring technique (e.g. SPC) by introducing predictive analysis of software process anomaly. Predictive detection of software process anomaly allow us to initiate appropriate action to prevent the anomaly or minimize its influence on the process performance. To accomplish this objective we proposed three new challenges and corresponding research questions (available in Section 4). We proposed a solution approach that predictively generates future behaviour of a process, predicts anomalies and interprets them. We extended the available “Monitoring problem-SPC based solution” approach in our proposed solution approach. Our key contributions include –

i) We predictively determined future behaviour of selected process characteristics. In this report, we selected contribution rate of OSS project as the process characteristic to monitor and control. Proposed approach allows us to predict future behaviour of this process characteristic utilizing available observed dataset. Rather than creating the burden of collecting legacy project data or past knowledge, this approach allows us to predict behaviour based only on observed data of the process. Details regarding prediction of behaviour are discussed in our proposed solution approach. In our validation approach, we plotted predicted data along with observed data to detect and analyze process anomalies.

ii) We detected anomalies using the predicted behaviour of the process characteristics. In this report, we applied statistical run tests (Table 4) to detect anomalies in both observed and predicted data. We investigated whether run tests are capable of detecting anomalies using predicted data or not. We presented successful outcomes of using run tests in this regard.

However, predicted behaviour has lower trustworthiness compared to observed behaviour. Therefore, while detecting anomalies, behavioral trends (i.e. specific pattern of multiple observations) are treated with higher importance compared to single observation failure. We considered limit and trend tests as significant anomaly test for predictive data points.

iii) We developed guidelines for interpretation of detected anomalies in respect of both observed and predicted anomalies. In this report, we no longer consider isolated anomaly interpretation. Instead, we provide a collaborative interpretation of anomalies in respect of both observed and predicted anomalies. We also guided required actions corresponding to these interpretations. These guidelines are available in Table 5. This interpretation technique allows us to detect process anomalies or their pattern early and take corrective actions to prevent them or minimize their influence on the process. It also provides more information to help cause investigation process.

This report is not a solution to process monitoring or a solution to apply SPC in software process. This report should be considered as an initial step towards predictive detection of process anomalies using process-monitoring technique SPC. We developed guidelines for predictive data generation, anomalies detection, anomalies interpretation and required actions in this regard. Next steps of this research include, i) Enhancement of the proposed approach using a sophisticated prediction mechanism, ii) Comprehensive study with a full length software development project, iii) Consideration of more uncertainty factors iv) Better representation of uncertainty for uncertainty factors. v) Validation of the proposed approach with different software development contexts.
9. REFERENCES