

**AN ASSESSMENT AND ANALYSIS TOOL FOR
STATISTICAL PROCESS CONTROL OF SOFTWARE PROCESSES**

**A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY**

BY

SERKAN KIRBAŞ

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING**

FEBRUARY 2007

Approval of the Graduate School of Natural and Applied Sciences

Prof.Dr. Canan ÖZGEN

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof.Dr. Ayşe KİPER

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Dr. Ayça TARHAN

Co-Supervisor

Assoc.Prof.Dr. Ali DOĞRU

Supervisor

Examining Committee Members

Prof.Dr. Volkan Atalay (METU, CENG) _____

Assoc.Prof.Dr. Ali Doğru (METU, CENG) _____

Dr. Ayça Tarhan (METU, IS) _____

Assoc.Prof.Dr. Onur Demirörs (METU, IS) _____

Assoc.Prof.Dr. Halit Oğuztüzün (METU, CENG) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Serkan Kırbaş

Signature: _____

ABSTRACT

AN ASSESSMENT AND ANALYSIS TOOL FOR STATISTICAL PROCESS CONTROL OF SOFTWARE PROCESSES

Kırbaş, Serkan

MS, Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Ali Doğru

Co-Supervisor: Dr. Ayça Tarhan

February 2007, 217 pages

Statistical process control (SPC) which includes very powerful techniques used in other mature engineering disciplines for providing process control is not used by many software organizations. In software engineering domain, SPC is currently utilized only by organizations which have high maturity levels according to the process improvement models like CMM, ISO/IEC 15504 and CMMI. Guidelines and software tools to implement SPC techniques should be developed for effective use and dissemination of SPC especially for low maturity organizations.

In this thesis, a software tool (SPC-AAT) which we developed to assess the suitability of software processes and metrics for SPC and use of SPC tools is presented. With SPC-AAT, we aim to ease and enhance application of SPC especially for emergent and low maturity organizations. Control charts, histograms, bar charts and pareto charts are the supported SPC tools for this

purpose. We also explained the validation of the tool over two processes of a software organization in three case studies.

Key Words: Statistical process control, software, measurement, control chart, pareto chart.

ÖZ

İSTATİSTİKSEL SÜREÇ KONTROLÜNÜN YAZILIM SÜREÇLERİNE UYGULANABİLİRLİĞİNİ DEĞERLENDİRME VE ANALİZ ARACI

Kırbaş, Serkan

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Ali Doğru

Ortak Tez Yöneticisi: Dr. Ayça Tarhan

Şubat 2007, 217 sayfa

Birçok güçlü tekniği içinde barındıran İstatistiksel Süreç Kontrolü (İSK), diğer olgun mühendislik disiplinlerinde süreç kontrolünü sağlamak için kullanılmasına rağmen, çoğu yazılım şirketi tarafından kullanılmamaktadır. Yazılım mühendisliği alanında İSK şuan yalnızca CMM, ISO/IEC 15504 ve CMMI gibi süreç iyileştirme modellerine göre yüksek olgunluk seviyelerine sahip organizasyonlar tarafından yararlanılmaktadır. İSK'nın özellikle düşük olgunluk seviyelerindeki organizasyonlar tarafından etkin kullanımı ve yaygınlaştırılması için yeni kılavuzların ve yazılım araçlarının geliştirilmesi şarttır.

Bu çalışmada yazılım süreç ve metriklerinin İSK için uygunluğunu değerlendirmek ve İSK araçlarını kullanmak için geliştirdiğimiz bir yazılım uygulaması (SPC-AAT) sunulmuştur. SPC-AAT ile İSK'nın özellikle gelişmekte olan veya düşük olgunluk seviyelerindeki kurumlar için uygulanmasını

kolaylařtırmak ve geliřtirmek hedeflenmiřtir. Kontrol grafikleri, histogramlar, bar grafikleri ve pareto grafikleri bu ama iin SPC-AAT tarafından desteklediėimiz İSK aralarıdır. Bu alıřmada ayrıca uygulamamızın bir yazılım řirketinin iki adet sureci zerinde  durum deėerlendirmesini aıklamaktayız.

Anahtar Kelimeler: İstatistiksel sure kontrol, yazılım, lme, kontrol grafiėi, pareto grafiėi.

To the memory of my grandfather,
Mehmet KIRBAŞ

ACKNOWLEDGEMENTS

I send my deepest regards and gratitude to Dr. Ayça Tarhan and Assoc. Prof. Onur Demirörs for their expertise, insight, and guidance, and for their support during my study.

I am grateful to S.Ş.S., Ö.Ö., A.Y., M.E., Y.G. and H.O.C. for their help and contribution during development of the tool and case study implementations.

Also, I would like to thank to my family for their limitless patient and love which motivated me to complete this study.

TABLE OF CONTENTS

ABSTRACT	iv
ÖZ	vi
ACKNOWLEDGEMENTS	ix
TABLE OF CONTENTS	x
LIST OF TABLES	xiv
LIST OF FIGURES	xv
LIST OF ABBREVIATIONS	xxii
CHAPTER	
1. INTRODUCTION	1
1.1. Problem Statement	2
1.2. Statistical Process Control Assessment & Analysis Tool (SPC-AAT)	3
1.3. Roadmap	5
2. BACKGROUND	6
2.1. Variability in Processes and Statistical Control	6
2.2. SPC Tools	8
2.2.1 Shewhart's Control Charts	13
2.3. SPC in Software Development	20

2.4.	Guidelines for Applying SPC in Software Development.....	23
2.4.1	Application of Statistical Process Control to Software Processes	23
2.4.2	Utilization of SPC in Emergent Software Organizations: Pitfalls and Suggestions.....	24
2.4.3	Statistical Process Control - Assessment Model (SPC-AM)	24
2.5.	Measurement Data Collection.....	29
2.5.1	INTERMEDIATE.....	29
3.	DETAILS OF SPC-AAT	31
3.1.	Overview of the System.....	31
3.2.	General Description.....	33
3.2.1	Product Perspective	33
3.2.2	Product Functions	34
3.2.3	Constraints, Assumptions and Dependencies.....	34
3.3.	Specific Requirements.....	34
3.3.1	Workspace Use cases.....	34
3.3.2	Process Metric Data Use cases	36
3.3.3	Process Execution Record Use cases.....	37
3.3.4	Process Execution Questionnaire Use cases	38
3.3.5	Process Similarity Matrix Use cases.....	39
3.3.6	Base Process Clusters Use cases	41
3.3.7	Process Attributes Description Use cases	43
3.3.8	Process Metrics Use cases.....	43
3.3.9	Metric Usability Questionnaire Use cases	45

3.3.10	Metric Usability Rating Use cases.....	46
3.3.11	Metric Usability Assessment Results Use cases	47
3.3.12	Process Clusters Use cases.....	48
3.3.13	Use cases related with Using SPC Tools	49
3.3.14	Process Control Status Use cases	52
3.3.15	Use cases related with Out-of-Control Points	53
3.3.16	Help Use cases.....	55
3.4.	Design of SPC-AAT.....	56
3.4.1	Architecture	56
3.4.2	Detailed Descriptions.....	57
3.5.	Example Scenario.....	63
3.5.1	Creating a New Workspace.....	63
3.5.2	Importing Metric Data	66
3.5.3	Update Process Execution Records	72
3.5.4	Operations on Process Similarity Matrix	77
3.5.5	Identifying Base Process Clusters	78
3.5.6	Handling Metrics	79
3.5.7	Displaying Usability Results for Metrics.....	81
3.5.8	Operations on Process Clusters	83
3.5.9	Using SPC Tools Supported.....	84
3.5.10	Overall Process Control Results.....	87
4.	CASE STUDY	89
4.1.	Fundamentals of the Case	89

4.2.	Context-1 (Case Study A).....	91
4.3.	Context-2 (Case Study B).....	118
4.4.	Context-3 (Case Study C).....	144
4.5.	User Evaluation.....	171
5.	CONCLUSION AND FUTURE WORK.....	173
5.1.	Conclusion	173
5.2.	Future Work	177
	REFERENCES.....	179
	APPENDICES.....	185
A.	SPC-AM ASSETS	185
B.	TOOL INFORMATION	191
C.	DETAILS OF CASE STUDY-A.....	204
D.	DETAILS OF CASE STUDY-B	205
E.	DETAILS OF CASE STUDY-C	206
F.	SPC-AAT EVALUATION QUESTIONNAIRES	207

LIST OF TABLES

TABLES

Table 1 Processes and Metrics used in Case Studies	4
Table 2 Processes and Metrics used in the Case Studies	90
Table 3 Process Metrics (Original and Derived) for Case A	96
Table 4 Initial Results from Charted Data in Context-1	107
Table 5 Assignable Causes for Out-of-Control Points in Context-1	107
Table 6 Results of Case Study A	117
Table 7 Process Metrics (Original and Derived) for Case B	123
Table 8 Initial Results from Charted Data in Context-2	133
Table 9 Assignable Causes for Out-of-Control Points in Context-2	133
Table 10 Results of Case Study B	143
Table 11 Processes and Data Sets (Original and Derived) in Context-3	149
Table 12 Initial Results from Charted Data in Context-3	162
Table 13 Assignable Causes for Out-of-Control Points in Context-3	162
Table 14 Results of Case Study C	170
Table 15 SPC-AAT Requirements	194

LIST OF FIGURES

FIGURES

Figure 1 Example Check Sheet	8
Figure 2 Example Cause-and-Effect Diagram	9
Figure 3 Example Scatter Diagram	10
Figure 4 Example Run Chart	10
Figure 5 Example Histogram	11
Figure 6 Example Bar Chart	12
Figure 7 Example Pareto Chart	12
Figure 8 Example Control Chart	13
Figure 9 Example Control Chart	15
Figure 10 Control Charts supported for attributes data	18
Figure 11 Assessment Process of SPC-AM.....	28
Figure 12 General Use cases for SPC-AAT	32
Figure 13 Working Environment of SPC-AAT	33
Figure 14 Workspace Use cases.....	35
Figure 15 Process Metric Data Use cases.....	36
Figure 16 Process Execution Record Use cases.....	37
Figure 17 Process Execution Questionnaire Use case.....	38

Figure 18 Process Similarity Matrix Use cases.....	40
Figure 19 Base Process Clusters Use cases	42
Figure 20 Process Attributes Description Use cases	43
Figure 21 Process Metrics Use cases.....	44
Figure 22 Metric Usability Questionnaire Use cases	45
Figure 23 Metric Usability Rating Use cases.....	46
Figure 24 Metric Usability Assessment Results Use cases	47
Figure 25 Process Clusters Use cases.....	48
Figure 26 SPC Tools Use cases.....	50
Figure 27 Process Control Status Use cases	52
Figure 28 Out-of-Control Points Use cases	54
Figure 29 Help Use cases.....	55
Figure 30 Component Diagram.....	56
Figure 31 Class Diagram	57
Figure 32 Create Workspace.....	64
Figure 33 Create Assessment.....	64
Figure 34 Newly created Workspace.....	65
Figure 35 Import Metric Data	66
Figure 36 Open Metric Data File	67
Figure 37 Imported Metrics	68
Figure 38 Metric Data.....	69
Figure 39 Process Execution Records created	70
Figure 40 Metric Usability Assessments created	71

Figure 41 Process Execution Record Details.....	72
Figure 42 Inserting new Inputs	73
Figure 43 Input created.....	74
Figure 44 Roles of a Process Execution Record	75
Figure 45 Saving Process Execution Record.....	76
Figure 46 Process Similarity Matrix for Inputs.....	77
Figure 47 Re-identify Process Clusters	78
Figure 48 Base Metrics	79
Figure 49 Base Metric Questionnaire.....	80
Figure 50 Reporting Metric Usability Evaluation.....	81
Figure 51 Four Kinds of Reports.....	82
Figure 52 Metric Usability Evaluation Report.....	82
Figure 53 Menu Items on a Process Cluster	83
Figure 54 SPC Tools supported	84
Figure 55 Choosing Metrics to be Charted.....	85
Figure 56 A Control Chart drawn.....	85
Figure 57 Process Execution Questionnaire	86
Figure 58 Exclusion of a Metric Value.....	87
Figure 59 Assessment Results Summary.....	87
Figure 60 Control Status Report.....	88
Figure 61 eEPC for Bug Fixing Process (Case A)	93
Figure 62 Process Similarity Matrix for Bug Fixing (Case A)	95
Figure 63 Base Process Clusters for Bug Fixing Process.....	96

Figure 64 Derived Metrics Identified in Context-1	97
Figure 65 Metric Usability Questionnaire for “Bug Aging” Derived Metric of “Bug Fixing” Process.....	98
Figure 66 Metric Usability Ratings for “Bug Aging” Derived Metric of “Bug Fixing” Process.....	99
Figure 67 Metric Usability Report for Bug Fixing process	99
Figure 68 Cluster Distances for base Process Clusters.....	100
Figure 69 Process Clusters after first merge	101
Figure 70 Cluster Distances after first merge	101
Figure 71 Process Version B_C – Bug Aging Control Chart	103
Figure 72 Control Chart drawn for “Version B_C – Person Hours” pair.....	104
Figure 73 Control Chart drawn for “Version B_C – Person Hours” pair.....	105
Figure 74 Final Control Chart drawn for “Version B_C – Person Hours” pair..	106
Figure 75 Control Chart for Combined Data of Bug Aging	108
Figure 76 Control Chart for Combined Data of Person Hours	109
Figure 77 Final Control Chart for Combined Data of Bug Fixing.....	110
Figure 78 Final Control Chart for Combined Data of Person Hours	111
Figure 79 Process Version B_C – Bug Aging Histogram	112
Figure 80 Process Version B_C – Person Hours Histogram	113
Figure 81 Process Version D_E_B_C_A – Bug Aging Histogram	114
Figure 82 Process Version D_E_B_C_A – Person Hours Histogram.....	115
Figure 83 Bar Chart for Status	116
Figure 84 eEPC for Recruitment Process (Case B).....	120
Figure 85 Process Similarity Matrix for Recruitment (Case B).....	122

Figure 86 Base Process Clusters for Bug Fixing Process.....	123
Figure 87 Derived Metrics Identified in Context-2.....	124
Figure 88 Metric Usability Questionnaire for “Procurement Time Variance” Derived Metric of “Procurement” Process.....	125
Figure 89 Metric Usability Ratings for “Procurement Time Variance” Derived Metric of “Procurement” Process	126
Figure 90 Metric Usability Report for Procurement process.....	126
Figure 91 Process Clusters after the merge.....	127
Figure 92 Control Chart for Version B – Procurement Time Variance pair	128
Figure 93 Rules for OCPs (Case B)	129
Figure 94 Control Chart for Version B – Actual Procurement Time pair	130
Figure 95 Control Chart drawn for “Version B – Actual Procurement Time” pair	131
Figure 96 Control Chart for Version A_B – Procurement Time Variance pair..	134
Figure 97 Control Chart for Combined Data of Recruitment	135
Figure 98 Final Control Chart for Combined Data of Recruitment	136
Figure 99 Final Control Chart for Combined Data of Recruitment	137
Figure 100 Version B – Actual Procurement Time Histogram	138
Figure 101 Version B –Procurement Time Variance Histogram.....	139
Figure 102 Version A_B – Actual Procurement Time Histogram.....	140
Figure 103 Version A_B –Procurement Time Variance Histogram	141
Figure 104 Bar Chart for Position	142
Figure 105 eEPC for Bug Fixing Process (Case C)	146
Figure 106 Process Similarity Matrix for Bug Fixing	148

Figure 107 Base Process Clusters for Bug Fixing Process	149
Figure 108 Base and Derived Metrics Identified in Context-3	150
Figure 109 Metric Usability Questionnaire for “Bug Aging” Derived Metric of “Bug Fixing” Process.....	151
Figure 110 Metric Usability Ratings for “Bug Aging” Derived Metric of “Bug Fixing” Process.....	152
Figure 111 Metric Usability Results (Case C)	153
Figure 112 OCP Rules for Case C.....	154
Figure 113 Version B – Bug Aging Control Chart	155
Figure 114 Version B – Bug Aging Final Control Chart.....	156
Figure 115 OCP Rules for Case C (Final)	157
Figure 116 Control Chart drawn for “Version A – Bug Aging” pair.....	158
Figure 117 Control Chart drawn for “Version A – Bug Aging” pair.....	159
Figure 118 Final Control Chart drawn for “Version A – Bug Aging” pair.....	160
Figure 119 Process Clusters after the merge.....	160
Figure 120 Control Chart for Combined Data of Bug Fixing.....	161
Figure 121 Version A – Bug Aging Histogram	163
Figure 122 Version B – Bug Aging Histogram	164
Figure 123 Version A_B – Bug Aging Histogram.....	165
Figure 124 Bar Chart for Test Results.....	166
Figure 125 Version A_B – SB Found Pareto Chart	167
Figure 126 Version A_B – Error Reason Pareto Chart	168
Figure 127 Version A_B – Problem Source Pareto Chart.....	169
Figure A.1 Process Execution Record.....	185

Figure A.2 Process Execution Questionnaire	186
Figure A.3 Process Similarity Matrix.....	187
Figure A.4 Process Attributes Description	188
Figure A.5 Metric Usability Questionnaire for Base Metrics.....	189
Figure A.6 Metric Usability Questionnaire for Derived Metrics.....	190

LIST OF ABBREVIATIONS

CMM:	Capability Maturity Model
CMMI:	Capability Maturity Model Integrated
CMU:	Carnegie Mellon University
eEPC:	Extended Event-Driven Process Chain
FO:	Feature Owner
GQM:	Goal-Question-Metric
IEC:	International Electrotechnical Commission
IRS:	Interface Requirements Specification
ISO:	International Standards Organization
İSK:	İstatistiksel Süreç Kontrolü
L3:	Maturity Level 3
L4:	Maturity Level 4
LCL:	Lower Control Limit
LOC:	Lines of Code
M3P:	Model Manage Measure Paradigm
MUF:	Metric Usability Factor
MUQ:	Metric Usability Questionnaire
PSM:	Practical Software Measurement

QPM: Quantitative Process Management

SEI: Software Engineering Institute

SEPG: Software Engineering Process Group

SG: Specific Goal

SP: Specific Practice

SLOC: Source Lines of Code

SPC: Statistical Process Control

SPC-AM: Assessment Model for Statistical Process Control

SPC-AAT: Statistical Process Control Assessment & Analysis Tool

SW: Software

TL: Team Leader

UCL: Upper Control Limit

VE: Verification

XmR: X (Individual) and Moving Range

CHAPTER 1

INTRODUCTION

It is a well known fact that there are lots of failure stories of Software Projects [49]. The basic reason for this is that Software Engineering is not a mature engineering discipline as Civil Engineering, Electrical Engineering, etc. Especially, measurement is an open area to enhance in the maturing process of software engineering. High number of failed software projects is not surprising if we remember an old management adage; “You can't manage what you don't measure”.

In the future, systems in Software Engineering will be much more complex and controllability will decrease [32]. Keeping this in mind, the pressure on software engineering industry to find mature ways to measure and control software processes and product quality is increasing. In the past, measurement has been treated as an additional and extra task in software industry [16]. But now software measurement is considered to be a basic software engineering practice, as evidenced by its inclusion in the Level 2 maturity requirements of the Software Engineering Institute's Capability Maturity Model Integration (CMMI) [9] products and related commercial software process standards [32].

As it is stated implicitly before, measurement is not a target, it is just a tool in order to control and manage software projects. In the mature manufacturing industries, SPC (Statistical Process Control) has been widely used for this purpose. SPC was originated by the studies of Walter Shewhart in 1930s [45]. W.

Edward Deming had also major contributions to SPC [11] [12]. The basic principle of SPC is that by establishing and sustaining stable levels of variability, processes will yield predictable results [17]. According to SPC, almost all characteristics of processes and products display variation when measured over time.

1.1. Problem Statement

SPC used in other mature disciplines to control processes is also recognized by software industry and embedded into process improvement models like CMM [38], ISO/IEC 15504 [27] and CMMI [9]. The companies that are using one of these models start to implement SPC [8] [10], as a requirement of high maturity levels (level 4 and above). Besides these process improvement models, some researchers contribute to this trend by providing approaches to utilize SPC techniques for software industry [5] [16] [17]. In the literature, there are also a number of articles and tutorials that discuss the reasons of difficulties and provide suggestions on implementation of SPC for software [6] [7] [14] [18] [20] [31] [40] [41] [50]. But we lack satisfactory guidelines for software companies to implement SPC techniques with convincing information. Realizing this need, Sargut reported a study of applying SPC to an emergent software organization and prepared guidelines to apply SPC techniques [43]. Then Tarhan proposed an assessment model (SPC-AM) to evaluate the suitability of SPC for software processes and metrics with the aim of providing guidelines to direct SPC implementation [47].

Despite these studies on providing guidelines to direct SPC implementation there is no tool to guide and start SPC implementation in a software organization. Without proper software tools, it is still not easy to utilize SPC especially for emergent organizations. Current techniques are cumbersome, hard to apply and difficult to follow the results without the support of software tools. And because of that, experts are needed to guide organizations to apply SPC techniques. Moreover, consistency and correctness of the results are depending on the human

being. Without software tool support, external statistics tools (Minitab Statistical Software [33], Matlab, etc.) should be used for generating the statistical charts. The metric data should be entered and arranged manually to the related statistics tools. Also it is not possible to relate metric data to process executions with statistics tools. This needs high effort and consumes lots of time besides being very error-prone. As a result; with the current techniques, it is not easy to continue applying SPC for the people other than experts. This hinders the dissemination of SPC in the emergent organizations.

1.2. Statistical Process Control Assessment & Analysis Tool (SPC-AAT)

In this study, we investigated how to ease and enhance applying SPC to the emergent organizations and reduce the time required. In order to do this, we developed an SPC assessment and analysis tool which is called **SPC-AAT**. SPC-AAT automates the assessment process of SPC-AM to guide especially emergent organizations to apply SPC and it is used for statistical analysis. Control charts, histograms, bar charts and pareto charts are the supported SPC tools for this purpose. With SPC-AAT, we will contribute to effective use and dissemination of SPC among emergent software organizations. Therefore, feedback loops can be provided easily regardless of the maturity level of the software organization. Also SPC-AAT is one of the few tools which relate process metric data to process executions for statistical analysis.

The basic functionalities that SPC-AAT provides are importing process metric data to SPC-AAT, organizing process metric data for SPC analysis, defining process metrics, creating new derived metrics from existing base and/or derived metrics, assessing processes and process metrics for applicability of SPC, performing rational sampling automatically according to assessment results, applying SPC tools on the processes and process metrics, providing questionnaires to find out the reasons for variation of the processes, supporting what-if analysis for different rational sampling choices, reporting and printing the assessment and analysis results.

To validate SPC-AAT, we implemented three case studies at a project-based working software organization having CMMI L3. We worked on recruitment and bug fixing processes (for two different projects) of the organization and related metrics of these processes. These processes and the metrics used in the case studies can be seen in Table 1.

Table 1 Processes and Metrics used in Case Studies

Process Name	Metric Name
Bug Fixing (Project A)	Bug Aging
	Person Hours (Effort)
	Status
Recruitment	Actual Procurement Time
	Procurement Time Variance
	Position
Bug Fixing (Project B)	Bug Aging
	Estimated Bug Aging
	Estimation Variance
	Estimation Capability
	Problem Source
	Error Reason
	Should-be found
	Status

Case studies showed us that using SPC-AAT we could utilize SPC in an emergent organization besides assessing the usability of SPC for the processes and the related metrics in hours. After that company staff can also continue monitoring the analyzed processes with importing newly generated process metric data and can use SPC tools easily on the process metric data. Besides these, we could also detect improvement opportunities for the analyzed processes and SPC-AAT during the case studies.

1.3. Roadmap

In Chapter 2, we provide the details about the related research concerning this study. Statistical Process Control (SPC) and SPC implementations for software are explained. The tools used to support SPC are described here. Especially control charts are explained in detail. The basic components and assets of SPC-AM are also given in this chapter.

In Chapter 3, we provide the details related to the tool we developed, SPC-AAT. We describe the requirements of the tool as UML use case diagrams. We also present the design of SPC-AAT application by using UML class diagrams and UML component diagrams. Finally, usage of our tool is described over one scenario.

In Chapter 4, we mention the validation of SPC-AAT by the case studies implemented and questionnaires performed. We provide the details related to each case study implementation. The results of these implementations are also presented in this chapter. As a last thing, we provide the results of the questionnaires held about SPC-AAT.

Finally in Chapter 5, we provide our conclusions on our study and portray overall findings. In this chapter, we also describe potential subjects for future work.

CHAPTER 2

BACKGROUND

As old management adage, “You can't manage what you don't measure”, points measurement is very important in order to control and manage software projects. In the mature manufacturing industries, SPC (Statistical Process Control) has been widely used for this purpose. SPC was originated by the studies of Walter Shewhart in 1930s [45]. W. Edward Deming had also major contributions to SPC [11] [12]. Then Donald J. Wheeler followed Shewhart's and Deming's studies [51]. In this section we will give details about SPC, SPC tools and SPC in software industry. Besides we will described a study performed on measurement data collection.

2.1. Variability in Processes and Statistical Control

Statistical process control principles hold that by establishing and sustaining stable levels of variability, processes will yield predictable results [45] [46]. Then we can say that the processes are under statistical control. Controlled processes are stable processes, and stable processes enable you to predict results [17][20].

According to SPC, almost all characteristics of processes and products display variation when measured over time and there are two types of the variation [46]:

- common cause variation
- assignable cause variation

Common cause variation is variation in process performance due to normal or inherent interaction among the process components (people, machines, material, environment, and methods). It is naturally existent within the defined processes and can only be avoided by performing improvement programs.

The other type of variation in process performance is due to assignable causes. Assignable cause variations arise from events that are not part of the normal process. They represent sudden or persistent abnormal changes to one or more of the process components [17] [20]. For example, if developers start to use a new IDE for software development then source lines of code produced a day may be lower during adaptation period. This can be explained as the assignable cause variation in a process. In equation form, the concept is

$$\text{[total variation]} = \text{[common cause variation]} + \text{[assignable cause variation]}$$

When all assignable causes have been removed and prevented from reoccurring in the future so that only a single, constant system of chance causes remains, we have a stable and predictable process. Then we can expect the outcome will be within certain limits for the same process. In this way, we can prepare achievable plans, meet cost estimates and scheduling commitments, and deliver required product functionality and quality with acceptable and reasonable consistency. Several attributes or variables are defined to represent the outcomes of the process in order to measure the variance in process behavior over time. Then the variability in process behavior can be tracked through these measures. Errors found during system test, effort spent for bug fixing, SLOC produced during a project may all be examples to represent outcomes of the related processes.

Although a process is stable (under control) it may not be capable. In other words, process performance may not be satisfactory according to the objectives of organization or project. If this is the case, process should be improved to make the process capable.

To conclude, with statistical process control we first aim to make the process stable by detecting assignable causes of variation and removing them. As the second step, we aim to provide a capable process by demonstrating the chance causes and improving the process if necessary. To achieve these aims, SPC provides powerful tools to analyze the processes. SPC tools are described in the following section.

2.2. SPC Tools

The basic tools used for statistical process control are described below [25] [34]:

Defect Density				
Project Name:				
Date	Software Component	SLOC	Number of Defects	Defect Density

Figure 1 Example Check Sheet

Check Sheet: Check sheets are good means for collecting data efficiently, reliably and easily. As the detail and characteristics of data are different, check sheets are designed specifically considering the particular needs. Metric datasheets are used extensively in order to represent the data in the desired format.

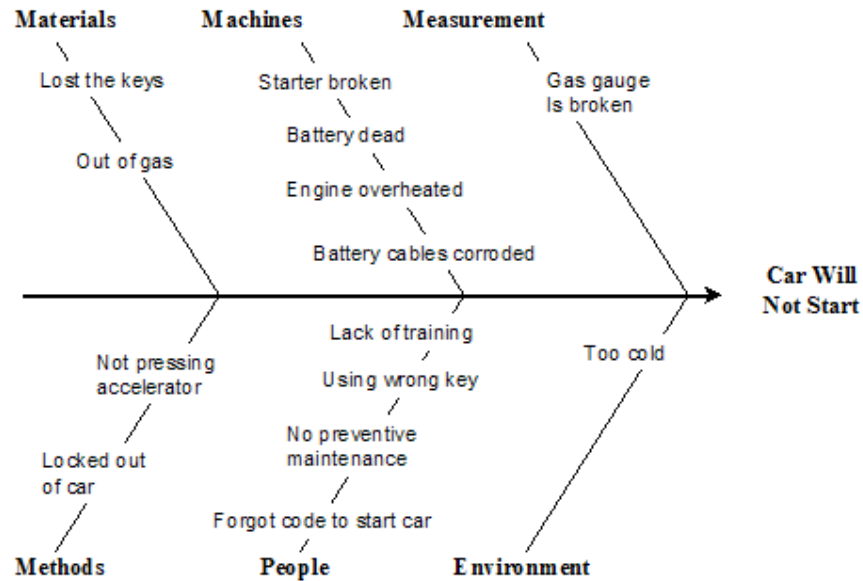


Figure 2 Example Cause-and-Effect Diagram

Cause-and-Effect Diagram: Cause-and-effect diagrams are useful tools to visualize, categorize and rank potential causes of a problem, a situation or any outcome. They are also named as fishbone diagrams because of their shapes and are usually formed as a result of a discussion or a brainstorming session of a group of people.

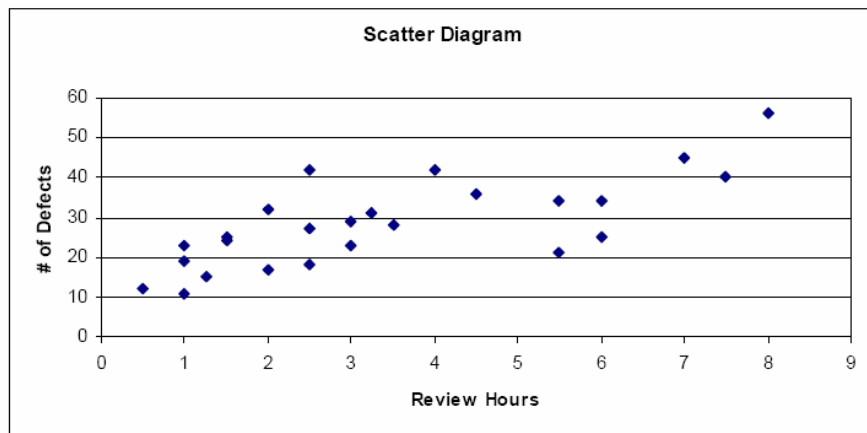


Figure 3 Example Scatter Diagram

Scatter Diagram: In a scatter diagram, data for two variables are collected in pairs (x_i, y_i) , and each point y_i is plotted against corresponding x_i . This is a useful plot for identifying a potential relationship between two process characteristics. Scatter diagrams may be used for regression analysis.

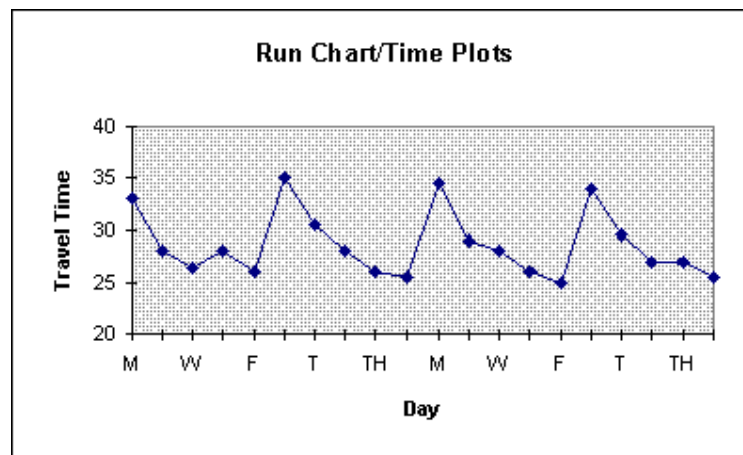


Figure 4 Example Run Chart

Run Chart: Run charts are specialized, time-sequenced form of scatter diagrams that can be used to examine data quickly and informally for trends or other patterns that occur over time. They dynamically observe performance of one or more processes over time. They are useful for visualizing performance after a process change.

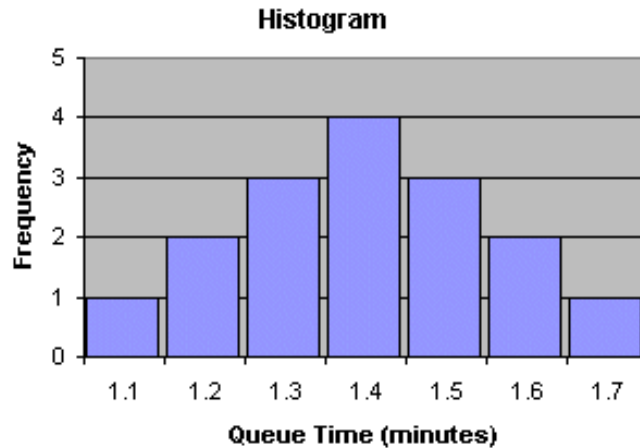


Figure 5 Example Histogram

Histogram: Histograms show the frequency distribution of data in a sample. The first step to draw a histogram is to categorize the data into classes with equal ranges. Then the number of data in each class is found and depicted with bars on the graph. The data represents the state of a system at a certain time; thus there is no time dimension. Histograms are quite practical to visualize central tendency and skewness of an attribute.

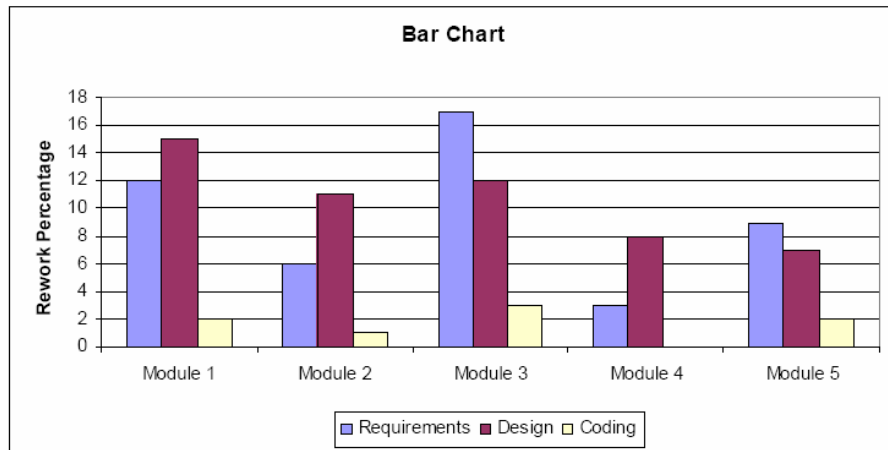


Figure 6 Example Bar Chart

Bar Chart: Bar charts are like histograms. But they are not only used for depicting the frequencies of occurrences, but also for showing any numerical value of the attribute.

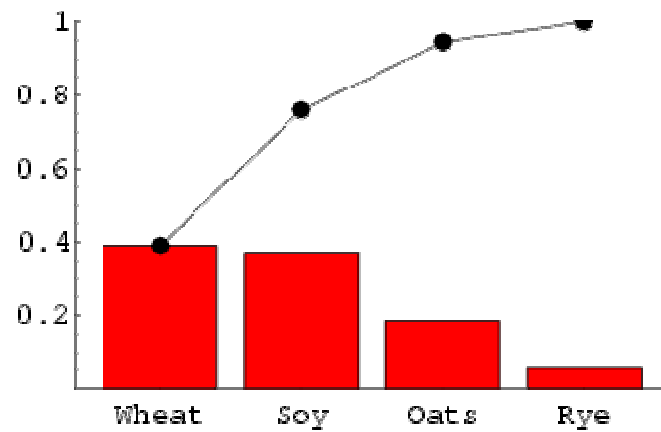


Figure 7 Example Pareto Chart

Pareto Chart: Pareto chart is another form of bar chart. However, the occurrences are ordered with respect to their frequencies. Pareto charts are good means to visualize the ranking of an attribute among different categories.

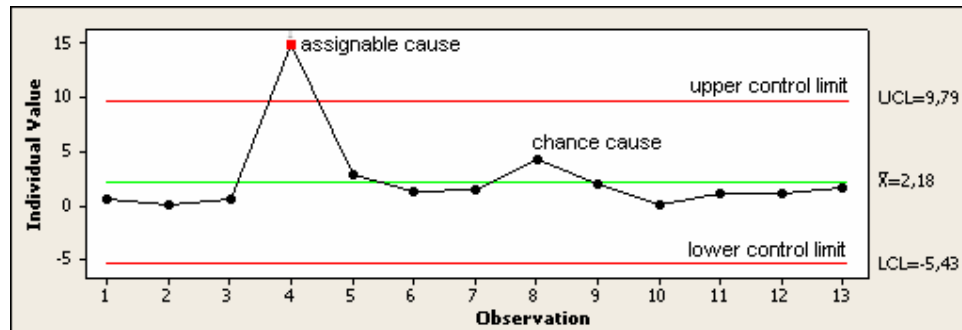


Figure 8 Example Control Chart

Control Chart: Control charts are sophisticated statistical analysis tools, which include upper and lower limits to detect any outliers. They look like run charts, but with the control limits and center line. They are frequently used in SPC analyses and described in detail in the following section.

2.2.1 Shewhart's Control Charts

The control chart was invented by Walter A. Shewhart while working for Bell Labs in the 1920s. The company's engineers had been seeking to improve the reliability of their telephony transmission systems. Because amplifiers and other equipment had to be buried underground, there was a business need to reduce the frequency of failures and repairs. By 1920 they had already realised the importance of reducing variation in a manufacturing process. Shewhart framed the problem in terms of Common- and special-causes of variation and introduced the control chart as a tool for distinguishing between the two in 1924. Shewhart stressed that bringing a production process into a state of statistical control, where

there is only common-cause variation, and keeping it in control, is necessary to predict future output and to manage a process economically [52].

Shewhart created the basis for the control chart and the concept of a state of statistical control by carefully designed experiments. While Shewhart drew from pure mathematical statistical theories, he understood data from physical processes never produce a "normal distribution curve" (a Gaussian distribution, also commonly referred to as a "bell curve"). He discovered that observed variation in manufacturing data did not always behave the same way as data in nature (Brownian motion of particles). Shewhart concluded that while every process displays variation, some processes display controlled variation that is natural to the process, while others display uncontrolled variation that is not present in the process causal system at all times [52].

In 1924 or 1925, Shewhart's innovation came to the attention of W. Edwards Deming. Over the next half a century, Deming became the foremost champion and exponent of Shewhart's work. Deming spread Shewhart's thinking, and the use of the control chart, widely in Japanese manufacturing industry throughout the 1950s and 1960s. More recent use and development of control charts in the Shewhart-Deming tradition has been championed by Donald J. Wheeler.

Shewhart control chart model depends on hypothesis testing. First of all, a sample of data (sufficient enough to represent the whole) is collected for the subject measure (i.e. number of defects in a piece of code). Then, its mean and variance are calculated. The lower and upper control limits (LCL and UCL) are derived from the mean and variance by the formula " $\text{Mean} \pm 3 \text{ Standard Deviation}$ " and data is analyzed using the statistical evidence on hand. By analyzing the data values with respect to upper and lower control limits together with their location in the zones, assignable causes are detected. Then necessary actions are taken and measurements are repeated. The charts are redrawn with the existing data values, and this process is repeated until no evidence remains for the existence of

assignable causes. Once the process is brought under control, further improvement activities are implemented to minimize the effect of common causes [43].

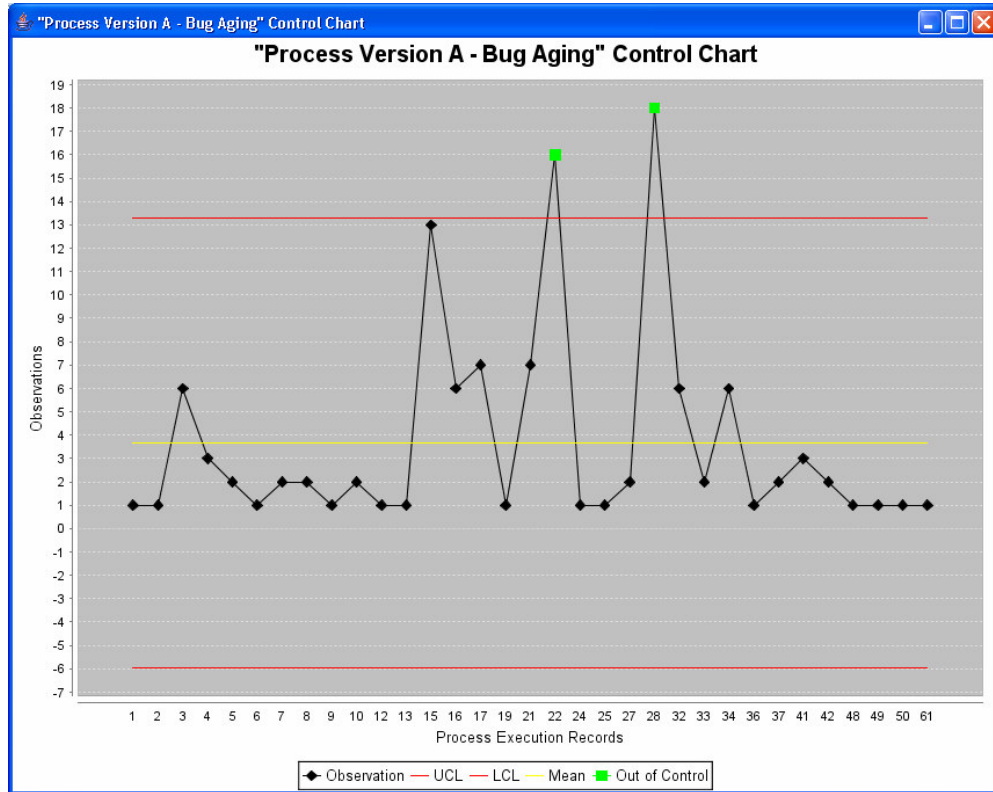


Figure 9 Example Control Chart

The measurement can be performed by means of either variables or attributes. Burr and Owen [5] define a variable as “measure of a product that can have any value between the limits of the measurement”, while an attribute as “count of things which may or may not be present in the product”. The nature of these two measurement categories necessitates different statistical analyses. Therefore, there are different types of control charts.

For variables data we have:

- X-Chart

- XmR Chart
- Xbar-R Chart
- Xbar-S Chart

For attributes data we have:

- p-Chart
- np-Chart
- c-Chart
- u-Chart
- XmR Chart

It is recommended to use Xbar-R chart or Xbar-S chart for subgroups of, and X-chart or XmR chart for individuals of variables data. p-charts, np-charts, c-charts, and u-charts as well as XmR charts are used for counts or rates of attributes data (see Figure 10). Below are further explanations on these control charts [17].

Xbar-R Chart: Averages and range chart is used to portray process behavior when we collect multiple measurements within a short period of time under basically the same conditions. Measurements are then grouped into self-consistent sets (subgroups) that can reasonably be expected to contain only common cause variation. The results of the groupings are used to calculate process control limits.

Xbar (average) charts answer the questions as “what is the central tendency of the process?” and “how much variation has occurred from subgroup to subgroup over time?”. The corresponding R (range) charts indicate the variation (dispersion) within the subgroups. It is advised that range charts be used only when there are 10 or less observations in each subgroup.

Xbar-S Chart: Averages and standard deviation chart is used instead of Xbar-R charts when subgroup size is larger than 10. S charts based on averages of the standard deviation within subgroups give tighter control limits, which brings

increased sensitivity to assignable causes. As the size of the subgroup increases, it becomes increasingly difficult to ensure homogeneity of the subgroup. Therefore, for reliability, selection of the subgroup size should be dictated first by the homogeneity of the subgroup and second by the subgroup size.

X-Chart: When measurements are spaced widely in time or when measurement is used by itself to evaluate or control a process, a time-sequenced plot of individual values, rather than averages, appears. This means that the subgroup size is 1.

An individual plot can detect more readily the following conditions than an Xbar-R chart: cycles (regular repetitions of patterns), trends (continuous movement up or down), mixtures (presence of more than one distribution), grouping or bunching (measurements clustering in spots), and relations between the general pattern of grouping and a specification.

XmR Chart: Individuals chart is frequently complemented by a corresponding moving range chart which depicts successive two-point moving ranges. This combination of charts for individual observations and moving ranges is called an XmR chart. XmR charts are especially useful to view trends in the process.

The idea behind XmR chart is that, when subgroups can easily include nonrandom components, we minimize the influence that nonrandom effects have upon estimates for sigma by keeping the subgroups as small as possible. The smallest possible subgroup size is 1. There is no way to estimate sigma from a single measurement so that we do the next best thing: We attribute the changes that occur between successive values to the inherent variability in the process. The absolute values of these changes are called two-point moving ranges.

When median moving range is used instead of the average moving range to compute the limits for an XmR chart, then we have “X and median mR” chart. The median moving range is frequently more sensitive to assigned causes when the moving range contains several very large values relative to the rest of the moving range values. Several high range values unduly inflate the average moving range and cause the upper and lower limits to expand.

Data characterized by a binomial model		Data characterized by a Poisson model		Other data based on counts	
Area of Opportunity		Area of Opportunity		Area of Opportunity	
n constant	n variable	constant	variable	constant	variable
np chart or XmR	p chart or XmR	c chart or XmR	u chart or XmR	XmR charts for counts	XmR charts for rates

Figure 10 Control Charts supported for attributes data

np-Chart: An np-chart is used when the count data are binomially distributed and all samples have equal areas of opportunity. For example, when there is 100% inspection of lots of size n (n constant) and the number of defective units in each lot is recorded.

p-Chart: A p-chart is used instead of an np-chart when the data are binomially distributed but the areas of opportunity vary from sample to sample. A p-chart is appropriate in the inspection example given for np-chart, if the lot size n were to change from lot to lot.

c-Chart: A c-chart is used when count data are samples from Poisson distribution and the samples have equal-sized areas of opportunity. C-charts are suggested, for example, when tracking the number of defects found in lengths, areas, or volumes of fixed (constant) size.

u-Chart: A u-chart is used instead of a c-chart when the count data are samples from a Poisson distribution and the areas of opportunity are not constant. Here, the counts are divided by the respective areas of opportunity to convert them to rates. A u-chart is more flexible than a c-chart because the normalizations that it employs enable it to be used when the areas of opportunity are not constant.

An XmR chart can be used in any of the above situations described for attributes data as well as when neither a Poisson nor a binomial model fits the underlying phenomena or when little is known about the underlying distribution. However, an XmR chart is not a reasonable choice when the events are so rare that the counts are small and values of zero are common (then the discreteness of the counts can affect the reliability of the control limits). If the average of the counts exceeds 1.00, an XmR chart offers a feasible alternative to the traditional attributes charts. In our study, we have used X charts for both attribute and variable data.

Wheeler suggests the following tests for detecting the assignable causes in a control chart [51] (“sigma” means standard deviation):

- Test-1: A single point falls outside the 3-sigma control limits.
- Test-2: At least two out of three successive values fall on the same side of, and more than two sigma units away from, the centerline.
- Test-3: At least four out of five successive values fall on the same side of, and more than one sigma unit away from, the centerline.
- Test-4: At least eight successive values fall on the same side of the centerline.

Tests 2, 3, and 4 are called *run tests* and are based on the presumptions that the distribution of the inherent, natural variation is symmetric about the mean; that the data are plotted in time sequence; and that successive observed values are statistically independent. The symmetry requirement means that the tests are designed primarily for use with X-bar and individuals charts. Strictly speaking, they are not applicable to R charts, S charts, or moving range charts [17]. Using test 1 avoids the need to make assumptions about the distribution of the underlying natural variation.

In our tool, it is possible to configure the run tests to be performed while drawing control charts. Each run test added increases our chances of detecting and out-of-control condition; however, it also increases our chances of getting a false alarm. Here the important point is that the decision to use a test should be given before

looking at the data. Determining the frequency with which a specific test leads to false alarms would be wise to identify its effectiveness.

2.3. SPC in Software Development

On software engineering discipline Humphrey can be regarded as a reflection of quality management. He describes a framework for software process management, outlines the actions to provide higher maturity levels and acts as a basic guide to improve processes in a software organization. In this book, Statistical Process Control appears as a means of data analysis technique for level 4 organizations. Humphrey emphasizes that measures should be robust, suggest a norm, relate to specific product and process properties, suggest an improvement strategy and be a natural result of the process. He also mentions that it is essential to have a model, but believing it too implicitly can be a mistake [43].

As SPC is more regarded in software industry, additional studies are being performed by the researchers. Lantzy is one of primary authors that mention the application of SPC concepts for software.. In his paper [31], he summarizes the concept of SPC and gives some practical examples from manufacturing industry. Then he offers a set of transformations on these principles via software quality characteristics revealing the uniqueness of software products. After giving the process-product relationship, he outlines a seven-step guideline for successful SPC implementation in a software organization. This study reveals four important points for the application of SPC to software processes:

- Metrics should correlate to the quality characteristics of the products that are defined by the customer
- Metrics should be selected for the activities that produce tangible items
- SPC should be applied only to critical processes
- The processes should be capable of producing the desired software product

In his article [6], Card discusses the utilization of SPC for software by also considering some of the objections and mentioning about possible implementation problems. He states that, as one objection, software development process does not involve repeated delivery of equivalent services or the fabrication of identical products. Another objection is the lack of a perfect measure of the attributes, which actually underlies the importance of metric definition. However, he argues that SPC does not rely on having a perfect measure, since SPC analysis is meant only to give some insight into how the process is functioning and it does not have to provide total visibility. He recommends beginning with a model of the process and then selecting techniques to monitor performance, in implementing SPC. He provides an example of a control chart to track testing efficiency, related to his approach.

In their book [5], Burr and Owen describe the statistical techniques currently available for managing and controlling the quality of software during specification, design, production and maintenance. This book is one of the very few resources in the area as it is a full reference on statistical methods from technical background of statistics and measurement to managerial concerns in software industry. The main focus is given to control charts as beneficial SPC tools and guidelines are provided for measurement, process improvement and process management within software domain.

A similar work is performed by Florac and Carleton [20]. This guidebook is about using measurements to manage and improve software processes. It shows how quality characteristics of software products and processes can be quantified, plotted, and analyzed, so that the performance of activities that produce the products can be predicted, controlled, and guided to achieve business and technical goals. Although many of the principles and methods described in the guidebook are applicable to individual projects, the primary focus is on the enduring issues that enable organizations to improve not just today's performance, but the long-term success and profitability of their operations. They represent

CMM understanding on the utilization of Statistical Process Control for software process improvement.

Barnard and Carleton [2] explain the results from a cooperative effort where Software Engineering Institute and the Space Shuttle Onboard Software Project experiment applying SPC analysis to inspection activities. During the study; project process descriptions are reviewed, data definitions are verified and validated, and experimentation and analysis are conducted. Since SPC analysis assumes data come from different sources, six functional areas of the project are treated separately. Control charts are depicted and examined for the metrics in search of stability:

In their book [16], Fenton and Pfleeger provide an accessible and comprehensive introduction to software metrics, now an essential component in the software engineering process. It also takes account of the fast changing developments in software metrics, most notably their widespread penetration into industrial practice.

In his article [50], Weller provides a distinct case in his article by presenting details on SPC implementation to analyze inspection and test data in a software organization. He proposes that in order to regard defect density as an indicator of product quality, he first wants to be sure that inspection process is stable in the organization. He uses \bar{X} and moving range charts for the lines of code inspected per hour for each inspection, and achieves a stable inspection process after removing the outliers from the dataset. Then he draws u-chart for the defect density data for each inspection. By these findings, he makes reliable estimations for inspection effectiveness and gains an insight on when to stop testing. The results of the analysis are discussed with the project teams at their weekly meetings, for three main reasons: It sends a message that the data is being used to make decisions on the projects; keeping the estimates and data in front of the teams make them aware of the progress toward the quality targets; and they want

to avoid the problem of “metrics are going into a black hole” which causes metric programs to fail.

Radice [41] describes SPC techniques constrained within software domain and gives a detailed tutorial by supporting his theoretical knowledge with practical experiences. He states that all SPC techniques may not be applicable for software processes and gives XmR and u charts as possible techniques. He also explains the relevance of SPC for CMM Level 4 and regards back-off of control charts in Level 4 as a mistake. He states five problems with control charts: too much variation; unnecessary use of control charts; lack of enough data; lack of specification limits from the clients; the idea that control charts cannot be used with software processes [43].

2.4. Guidelines for Applying SPC in Software Development

2.4.1 Application of Statistical Process Control to Software Processes

In his article [31], Lantzy outlines a seven-step guideline for successful application of SPC principles to the software process:

- Negotiate a set of prioritized software quality characteristics with the customer.
- Design, specify, and implement a software process capable of producing the desired software product.
- Establish process owners and empower them.
- Establish metrics for processes that correlate to the quality characteristics established for the end-item software product.
- Employ control charting or comparable techniques to determine the stability of each process.
- Bring processes in control by eliminating all special causes of variation.

- Continuously improve processes in order to bring control limits within tolerances so that the end-item software product meets customer requirements.

2.4.2 Utilization of SPC in Emergent Software Organizations: Pitfalls and Suggestions

Realizing that the existing studies are far from being capable of providing sufficient guidelines for applying SPC techniques to software processes, Sargut provided guidelines for SPC. In this regard, Sargut revealed that:

- SPC is not applicable to all software processes.
- SPC should only be applied to critical processes in a software organization.
- Not all SPC techniques are applicable to software processes.
- The processes should be well-defined and stable so that we can apply SPC techniques successfully.
- SPC techniques are required for achieving CMM Level 4.
- Control chart is the most sophisticated and useful SPC technique.

In our study, we have also benefited from these guidelines revealed.

2.4.3 Statistical Process Control - Assessment Model (SPC-AM)

SPC-AM is an assessment model to evaluate the applicability of SPC for software processes. It aims especially the emergent organizations that lack satisfactory guidelines for SPC implementation. SPC-AM addresses two basic requirements, with the purpose of providing guidance on initiating SPC applications for software processes:

- Rational sampling of process executions and data
- Metric data utilization (or suitability) for statistical analysis

The first requirement, *rational sampling*, aims to obtain and use data that are representative of the performance of the process with respect to the issues being studied. Process executions should be homogenous enough to ensure a single and constant system of chance causes. Otherwise, we can not use the basic assumption that resides at the heart of SPC [17] [20]:

$$[\text{total variation}] = [\text{common cause variation}] + [\text{assignable cause variation}]$$

SPC-AM proposes a clustering method to help grouping the process executions so that variations within any given group all come from the same system of chance causes. This clustering method is based on the following attributes of process executions:

- Inputs
- Outputs
- Activities
- Roles
- Tools and Techniques

Input is an entity that have been entered into the process or expended in its operation to achieve one or more outputs. The process has a number of inputs to each execution.

Output is an entity that have been produced by the process or created in its operation to fulfill process purpose. The process has a number of outputs from each execution.

Activity represents a distinct step within the process, when completed, supports transformation of input(s) into output(s) to achieve process purpose. The process has a number of activities that are carried out within each execution.

Role represents the actions assigned to or required of a person or group to carry out the activities within the process. The process allocates responsibility to a number of roles that participates in one or more process activities.

Tools and Techniques represent an implement used in or a practical method applied to some particular activity to support its completion. The process holds a number of tools and techniques that are used in one or more process activities.

Process executions are checked against the similarity in terms of these attributes. More similar the attributes more possible that they are from a single system of chance causes. Therefore, it is assumed that process executions in each group are consistently performed. This part of SPC-AM is also called as “*Process Consistency Assessment*” since consistency of the process executions is assessed to ensure the correct results from SPC implementation, regardless of process maturity or capability.

The second requirement is *metric utilization*. In the scope of this requirement, SPC-AM evaluates metrics’ usability for applying SPC. SPC-AM proposes to use six attributes which are called as “*Metric Usability Attributes*” for this purpose:

- Metric Identity
- Data Existence
- Data Verifiability
- Data Dependability
- Data Normalizability
- Data Integrability

Metric Identity includes general characteristics of a metric such as scale type, unit, formula, data type, range. Especially, scale type is important since control charts can not be used for nominal and ordinal scale metrics.

Data Existence is related with the availability of enough metric data points (20 at a minimum) for statistical analysis.

Data Verifiability focuses on the consistency in metric data recording and storage among process executions. Here the assumption of the model is that if measurements follow the same procedures, results observed will be consistent.

Data Dependability is related with recording of metric data as close to its source and for a specific purpose. The idea of measuring for a specific purpose is the core of the quality models [4] [37].

Data Normalizability and Data Integrability are related with usefulness of a metric for process improvement.

SPC-AM developed questionnaires based on these attributes for base and derived metrics separately. These two types of questionnaires are called as “*Metric Usability Questionnaire*” in the model. Questionnaires include a rating system based on the answers of questions, and accordingly, evaluate the usability of a specific metric for applying SPC.

The assessment process to follow when applying the model is given in Figure 11.

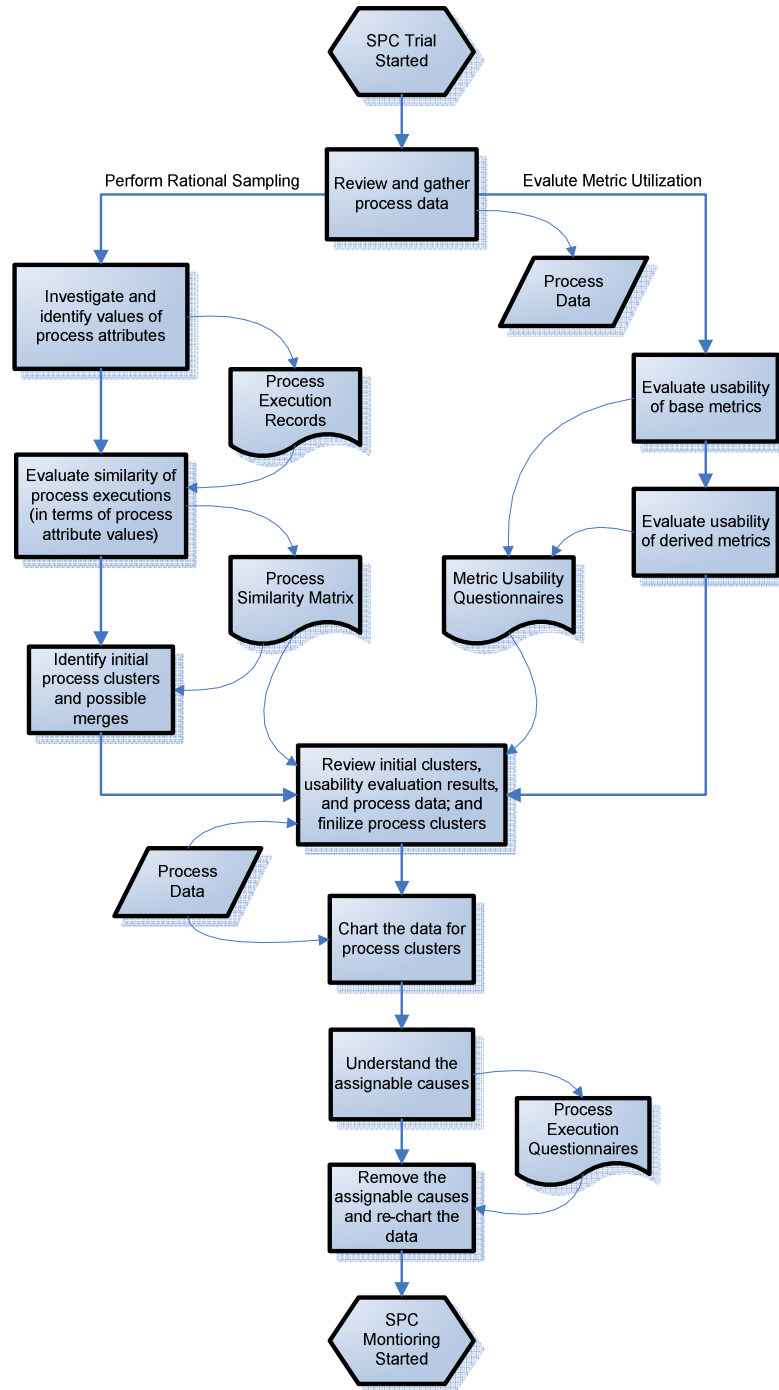


Figure 11 Assessment Process of SPC-AM

While performing assessment in accordance to SPC-AM, several assets shown as document object of eEPC in Figure 11 are used. The list of assessment assets are given below. Figures of these assets are provided in Appendix-A:

- Process Execution Record
- Process Execution Questionnaire
- Process Similarity Matrix
- Process Attributes Description
- Metric Usability Questionnaire for Base Metrics
- Metric Usability Questionnaire for Derived Metrics

In our study, we chose SPC-AM as the method to assess the suitability of process and metrics to use for statistical analysis.

2.5. Measurement Data Collection

SPC-AAT supports XML files generated by INTERMEDIATE tool [44] besides CSV and Excel files directly exported from third party tools holding metric data. INTERMEDIATE is described in the following section.

2.5.1 INTERMEDIATE

INTERMEDIATE [44] is a tool which is developed to integrate different measurement data and to provide necessary infrastructure to define new metrics. This tool can work together with commercial measurement tools, custom-made applications or directly databases. There are collectors defined in the framework between the tool and these different third party applications, to collect metrics. Collectors output XML files with a common DTD and Intermediate Tool parses these XML documents and the outputs are written to its database. Then the results of the metrics can be viewed by the user.

There are three kinds of collectors that collect metrics. They are differentiated according to the tools they interact. Therefore, one type for commercial

measurement tools, one for custom-made applications and one for databases. Intermediate Tool sends metric info that is going to be collected and collector sends this information to the related tool and the tool collects intended metric and return metric result to collector. Collector sends this metric value to Intermediate Tool and it places this value into Intermediate Tool's database.

INTERMEDIATE provides the facility of automatic data collection. User defines the period, date and times of the data collection and the system automatically triggers data collection operation when the time comes. If there is no need for the user's input then data collection is done full automatically behind the normal operations and user is only informed but if there is need for the user to enter some information then the collector should automatically pop-up on the window and user enters necessary information. Besides these properties, Intermediate Tool has the feature to define questionnaire-based forms that provide mapping between the entered information and related metrics.

CHAPTER 3

DETAILS OF SPC-AAT

The first section is the “Overview of the System” that describes the framework in which SPC-AAT is to be used. The second part is the “General Description” that is composed of “Product Perspective” and “Product Functions”. In Product Perspective section, user interfaces, software interfaces and operations are explained. In “Product Functions” section, information about general characteristics of users is given. Assumptions and constraints that affect design phase are also specified in this section.

In “Specific Requirements” section, the use cases of the system are explained by using UML use case diagrams. Use cases are grouped into related components and briefly described. In “Design of SPC-AAT” section, architecture of SPC-AAT tool and detailed description about main classes are given.

Finally in “Example Scenario” section, usage of SPC-AAT is explained over one example scenario.

3.1. Overview of the System

In the system there is only one user; SPC Implementer (or just User). SPC Implementer basically imports metric data, assesses process for applicability of SPC, applies SPC tools and displays assessment & SPC implementation results. This is described as a use case diagram in Figure 12.

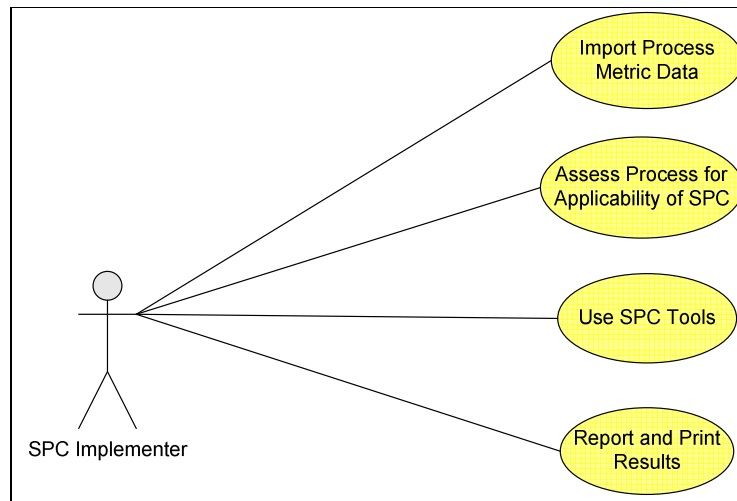


Figure 12 General Use cases for SPC-AAT

In the system, three kinds of metric data sources are defined to export metric values; commercial measurement tools, custom-made applications and connect directly to databases. There are also three kinds of collectors that collect metrics from a different metric data source. One way of getting data to SPC-AAT is via collectors with XML files. Collector1 sends the metric info that is going to be collected to commercial tool and commercial tool collects intended metric and return metric result to collector1. Collector1 saves these metric values into an XML file and SPC-AAT imports this XML file to use the metric data collected. Interaction of collector2 and collector3 through SPC-AAT is the same with collector1 except that collector3 interacts with a database and collector2 interacts with a custom-based application.

The second way to get metric data is importing Excel or CSV files generated by other applications. The environment that SPC-AAT works is described in the figure below.

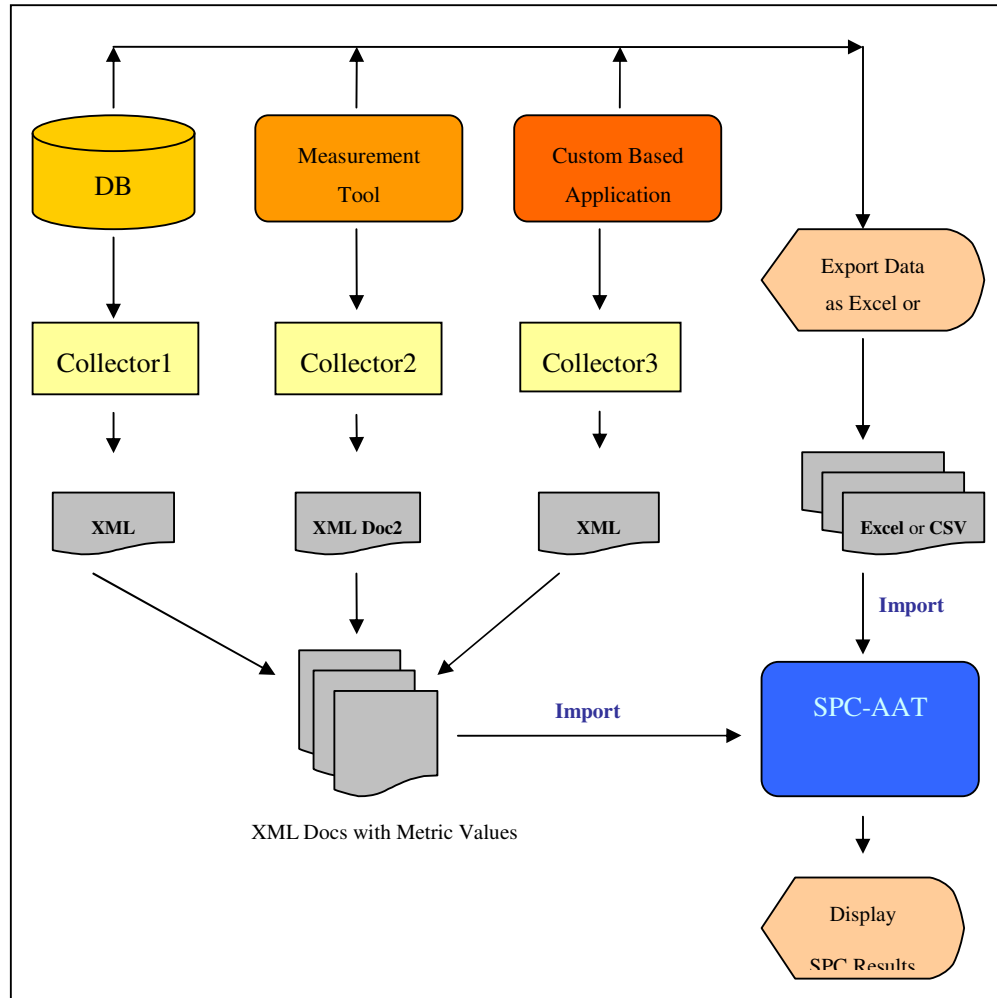


Figure 13 Working Environment of SPC-AAT

3.2. General Description

This section presents general factors that affect SPC-AAT requirements.

3.2.1 Product Perspective

SPC-AAT software is developed in Java programming environment, Java Development Kit version 1.5. SPC-AAT is platform independent and can run on any operating system. Analysis data can be stored in XML files and then can be

restored. Database is not used to store data. For generating reports, a free java library which is called JFreeReport is used. For charting utilities, another free java library JFreeChart is used.

3.2.2 Product Functions

SPC Implementer (simply User) is the only actor defined within the system. Use cases related to this actor are summarized in this chapter.

3.2.3 Constraints, Assumptions and Dependencies

The following assumptions have been made for the requirements specification:

- *It is assumed that only one process at a time will be analyzed with SPC-AAT*
- *Process executions entered to SPC-AAT should be in time sequence or should be sorted later by using sort feature of SPC-AAT*

3.3. Specific Requirements

Use case method is used for elicitation of the requirements for the SPC-AAT tool. This section presents all of the specific requirements of SPC-AAT software tool. Emphasis is placed on the functional requirements, which are explained as groups of related use cases.

3.3.1 Workspace Use cases

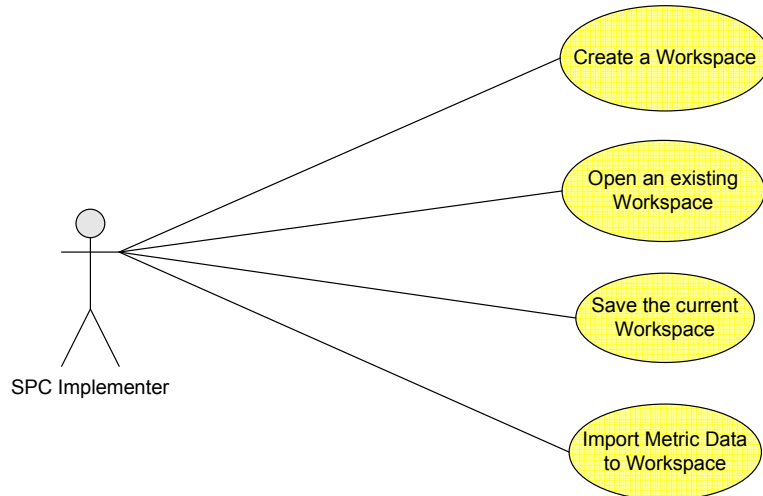


Figure 14 Workspace Use cases

Create a Workspace

In the scope of this use case, a new workspace is created to start a new assessment on the usability of SPC tools for one process and to apply supported SPC tools on the process. The name and the type (retrospective or prospective) of the process are requested from the User before creation of the workspace. All existing data is cleared when a new workspace is created.

Open an existing Workspace

Our tool stores workspace data for a process in a XML file. In other words, for each process analyzed with our tool there will be one XML file that stores the workspace data for the process. Therefore in the scope of this use case, User selects an XML file from the file system and then all workspace data saved in the XML file is restored and displayed to the User on GUI. User could see exactly the same data as it is in the save time.

Save the current Workspace

Our tool does not store the workspace data to a database. Instead of this, XML files are used for persistent storage. For each process analyzed with our tool there will be one XML file that stores the workspace data for the process. Therefore in

the scope of this use case, User selects a directory from the file system and enters a file name as the name of the XML file used for save operation. After that, all workspace data and configuration are saved in the XML file specified. When this XML file is restored, User could see exactly the same data as it is in the save time.

Import Metric Data to Workspace

In the scope of this use case, User selects a CSV, MS Excel or XML file from the file system and then all metric data in the selected file is imported and workspace is modified accordingly. The files containing the import data should specify all metric names to be imported as well as the metric values for the process executions. The formats for the import files are defined by examples in Appendix B.

Process execution records, base metrics and metric data are updated according to the data imported. If there are existing “process executions” defined in the workspace, then a metric from the imported metrics is used to map existing process executions with the ones come with imported data. If no metric name is chosen for mapping, new process execution records created for the imported data are appended to the end of the existing ones.

3.3.2 Process Metric Data Use cases

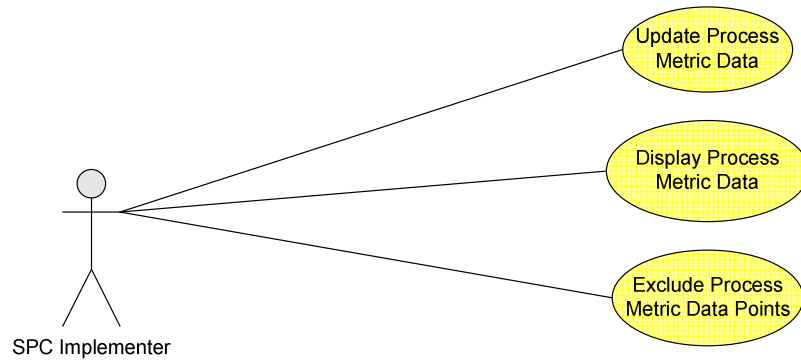


Figure 15 Process Metric Data Use cases

Update Process Metric Data

For each process execution defined in our tool, it is possible to enter metric values for all process metrics defined so far.

Display Process Metric Data

For each process execution defined in our tool, User can see metric values for all process metrics defined so far.

Exclude Process Metric Data Points

In the scope of this use case, User can exclude metric values of process executions defined in the tool. The purpose of this functionality is not to include metric values of Out-of-Control points in the statistical analysis.

3.3.3 Process Execution Record Use cases

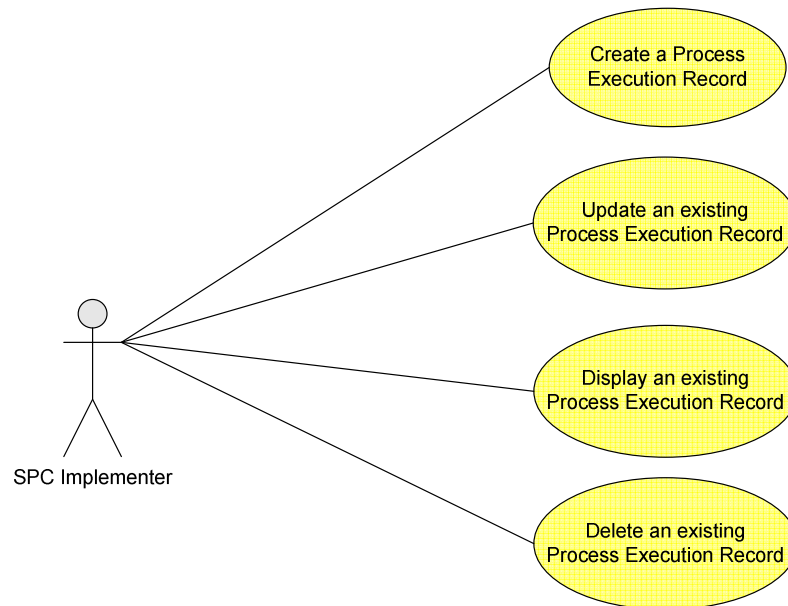


Figure 16 Process Execution Record Use cases

Create a Process Execution Record

Besides creating process execution records during import of metric data from files, User could also create process execution records manually from the GUI. User can define inputs, outputs, activities, roles, tools & techniques of the created process execution besides the name of the recorder and date of recording.

Display an existing Process Execution Record

In the scope of this use case, User can see inputs, outputs, activities, roles, tools & techniques of the selected process execution besides the name of the person who recorded this process execution and date of recording.

Update an existing Process Execution Record

In the scope of this use case, User can update inputs, outputs, activities, roles, tools & techniques of the selected process execution besides the name of the person who recorded this process execution and date of recording. To be clearer, User can add new inputs or delete existing inputs or change the existing inputs. This is also true for other process attributes.

Delete an existing Process Execution Record

In the scope of this use case, User can remove a selected process execution. All data related with the removed process execution should be cleared: metric data points, inputs, outputs, activities, roles, tools & techniques.

3.3.4 Process Execution Questionnaire Use cases

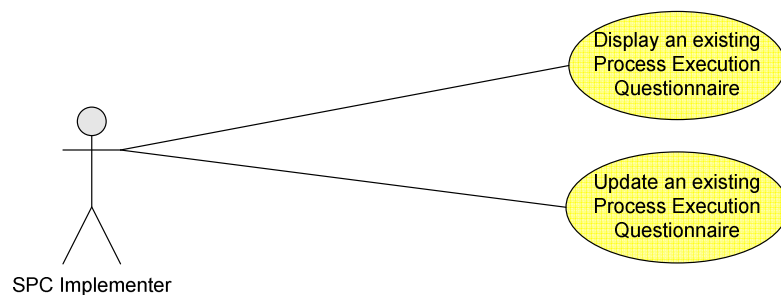


Figure 17 Process Execution Questionnaire Use case

Display an existing Process Execution Questionnaire

For each process execution defined, there is one process execution questionnaire attached to it. This questionnaire is used for detecting abnormalities in the execution of the process. In other words, this questionnaire helps User to find out external factors that affect the process. This questionnaire is filled for each Out-of-Control Point detected during retrospective analysis. For prospective analysis, it is recommended to fill for each process execution record created.

In the scope of this use case, User can see the process execution questionnaire of a selected process execution record on GUI. To be clearer, User can see the answers for the questions of the process execution questionnaire besides the name of the person who filled the questionnaire and the date of filling.

Update an existing Process Execution Questionnaire

For each process execution defined, there is one process execution questionnaire attached to it. This questionnaire is used for detecting abnormalities in the execution of the process. In other words, this questionnaire helps User to find out external factors that affect the process. This questionnaire is filled for each Out-of-Control Point detected during retrospective analysis. For prospective analysis, it is recommended to fill for each process execution record created.

In the scope of this use case, User can update the process execution questionnaire of a selected process execution record on GUI. To be clearer, User can change the answers for the questions of the process execution questionnaire besides the name of the person who filled the questionnaire and the date of filling.

3.3.5 Process Similarity Matrix Use cases

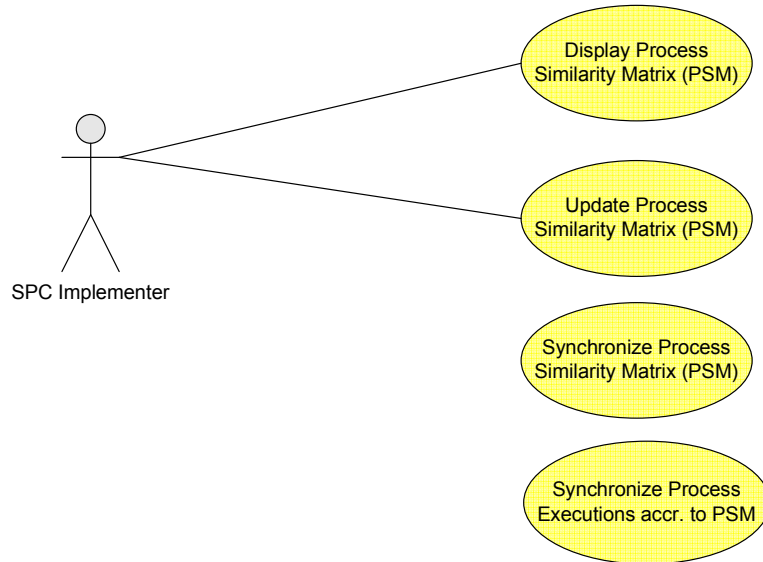


Figure 18 Process Similarity Matrix Use cases

Synchronize Process Similarity Matrix (PSM)

In the scope of this use case, process similarity matrix should be updated automatically according to the changes done on process execution records. In other words, process attributes (inputs, outputs, activities, roles, tools & techniques) defined for process execution records should be consistent with the information resides on PSM. Execution of the following use cases will trigger this use case:

- *Import Metric Data to Workspace*
- *Create a Process Execution Record*
- *Update an existing Process Execution Record*
- *Delete an existing Process Execution Record*

Display Process Similarity Matrix (PSM)

PSM is a grid which holds the process executions defined so far on one side and the set of process attribute values of all process executions in the other side. PSM helps User to see the differences among process executions in terms of process attribute values.

In the scope of this use case, User can see the inclusion or exclusion of process attribute values for existing process execution records defined so far.

Update Process Similarity Matrix (PSM)

PSM is a grid which holds the process executions defined so far on one side and the set of process attribute values of all process executions in the other side. PSM helps User to see the differences among process executions in terms of process attribute values.

In the scope of this use case, User can add/remove process attribute values to process executions by checking/un-checking the corresponding cells. User can also create a new process attribute value which is not defined for any of the existing process execution. Besides these, User can create a new process execution record or delete an existing process execution records on PSM.

Synchronize Process Executions accr. to PSM

In the scope of this use case, process execution records should be updated automatically according to the changes done on PSM. In other words, process attributes (inputs, outputs, activities, roles, tools & techniques) defined for process execution records should be consistent with the information resides on PSM. Execution of the following use cases will trigger this use case:

- *Update Process Similarity Matrix (PSM)*

3.3.6 Base Process Clusters Use cases

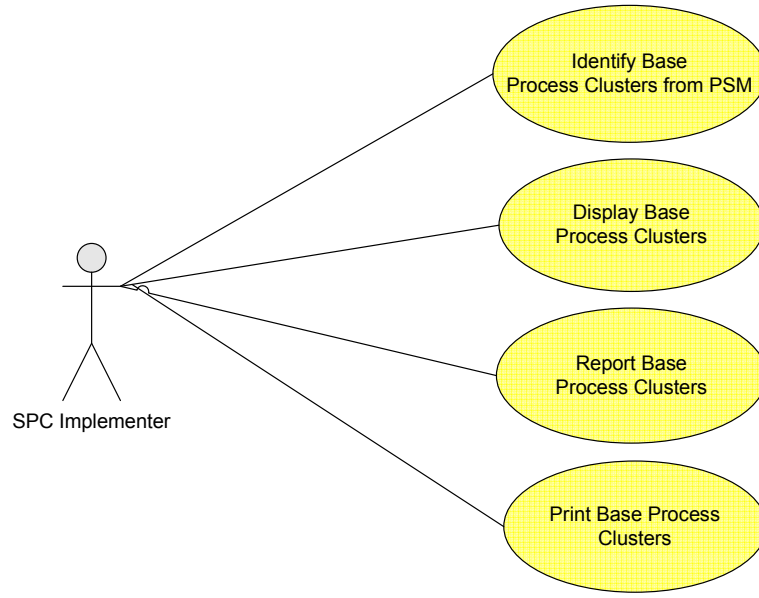


Figure 19 Base Process Clusters Use cases

Identify Base Process Clusters from PSM

In the scope of this use case, process execution records are clustered according to their process attribute values and one process cluster is created for each cluster found out. To be clustered in the same base process cluster, two process executions should have exactly the same process attribute values for inputs, outputs, activities, roles, tools & techniques.

Display Base Process Clusters

In the scope of this use case, User can see all the base process clusters identified according to the process attribute values of existing process execution records. User can also see inputs, outputs, activities, roles, tools & techniques of a selected base process cluster.

Report Base Process Clusters

In the scope of this use case, a report which shows all the base process clusters identified and the number of process executions clustered for each base cluster should be generated and displayed to User.

Print Base Process Clusters

User can print the report generated in the *Report Base Process Cluster* use case.

3.3.7 Process Attributes Description Use cases

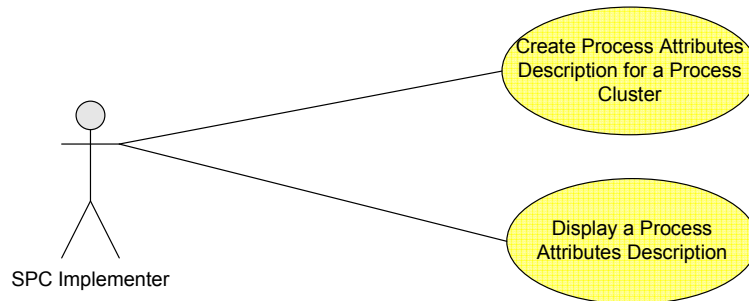


Figure 20 Process Attributes Description Use cases

Create Process Attributes Description for a Process Cluster

While clustering process execution records according to their process attribute values and creating clusters, process attributes description of process clusters should be created by using process attribute values (inputs, outputs, activities, roles, tools & techniques) of the process execution records clustered. Process attributes description contains information about inputs, outputs, activities, roles, tools & techniques of a process cluster.

Display a Process Attributes Description

In the scope of this use case, User can see the process attributes description of a selected process cluster. To be clearer, User should see information about inputs, outputs, activities, roles, tools & techniques of a process cluster on GUI.

3.3.8 Process Metrics Use cases

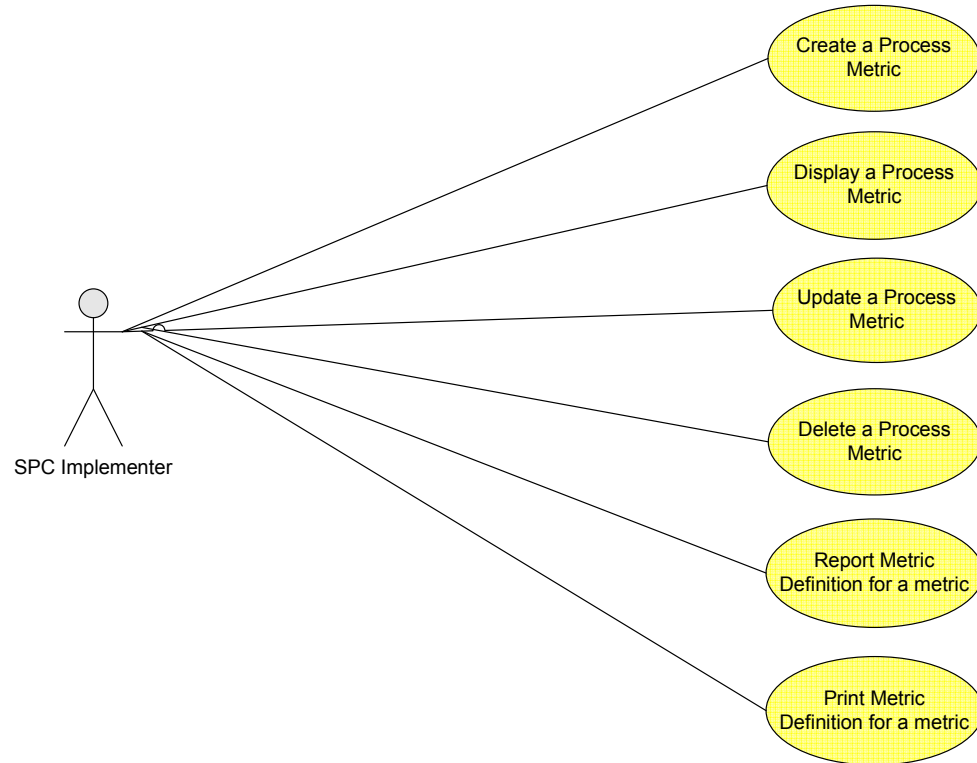


Figure 21 Process Metrics Use cases

Create a Process Metric

Besides creating process metrics during import of metric data from files, User could also create process metrics manually from the GUI. User can enter metric name, conceptual definition of the created process metric besides the name of the person who created the metric and date of creation. Base metrics and derived metrics should be differentiated. User can enter also metric formula for derived metrics.

Display a Process Metric

In the scope of this use case, User can see type (base or derived), metric name, conceptual definition of a selected process metric besides the name of the person who created the metric and date of creation. User can see also metric formula for derived metrics.

Update a Process Metric

In the scope of this use case, User can change metric name, conceptual definition of a selected process metric besides the name of the person who created the metric and date of creation. User can change also metric formula for derived metrics.

Delete a Process Metric

In the scope of this use case, User can remove a selected process metric. All data related with the removed process metric should be cleared: metric data points, metric name, and conceptual definition.

Report Metric Definition for a metric

In the scope of this use case, a report which shows name, definition, formula, scale, unit, type and range of a selected process metric should be generated and displayed to User.

Print Metric Definition for a metric

User can print the report generated in the *Report Metric Definition for a metric* use case.

3.3.9 Metric Usability Questionnaire Use cases

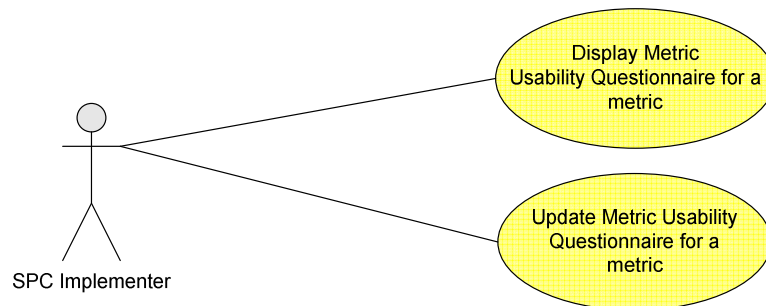


Figure 22 Metric Usability Questionnaire Use cases

Display Metric Usability Questionnaire for a metric

For each process metric defined, there is one metric usability questionnaire attached to it. This questionnaire is used for analyzing the usability of a process

metric for applying SPC tools on its values. In other words, this questionnaire helps User to decide on using this process metric in statistical analysis or not. This questionnaire is filled for each process metric defined.

In the scope of this use case, User can see the metric usability questionnaire of a selected process metric on GUI. To be clearer, User can see the answers for the questions of the metric usability questionnaire besides the name of the person who filled the questionnaire and the date of filling.

Update Metric Usability Questionnaire for a metric

For each process metric defined, there is one metric usability questionnaire attached to it. This questionnaire is used for analyzing the usability of a process metric for applying SPC tools on its values. In other words, this questionnaire helps User to decide on using this process metric in statistical analysis or not. This questionnaire is filled for each process metric defined.

In the scope of this use case, User can update the metric usability questionnaire of a selected process metric on GUI. To be clearer, User can change the answers for the questions of the metric usability questionnaire besides the name of the person who filled the questionnaire and the date of filling.

3.3.10 Metric Usability Rating Use cases

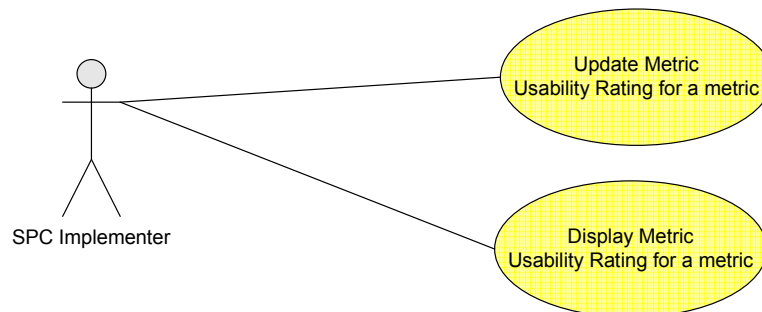


Figure 23 Metric Usability Rating Use cases

Update Metric Usability Rating for a metric

Metric usability attributes of metric usability questionnaires are rated within four ordinal values, based on the answers to the questions of the questionnaires: Fully satisfied (F: %86-100), Largely satisfied (L: %51-85), Partially satisfied (%16-50), and Not satisfied (N: %0-15).

In the scope of this use case, User can change the rating of Metric Identity, Data Existence, Data Verifiability and Data Dependability metric usability attributes. According to these ratings assigned, overall rating of the metric should be updated.

Display Metric Usability Rating for a metric

Metric usability attributes of metric usability questionnaires are rated within four ordinal values, based on the answers to the questions of the questionnaires: Fully satisfied (F: %86-100), Largely satisfied (L: %51-85), Partially satisfied (%16-50), and Not satisfied (N: %0-15).

In the scope of this use case, User can see the rating of Metric Identity, Data Existence, Data Verifiability and Data Dependability metric usability attributes besides the overall rating of the process metric.

3.3.11 Metric Usability Assessment Results Use cases

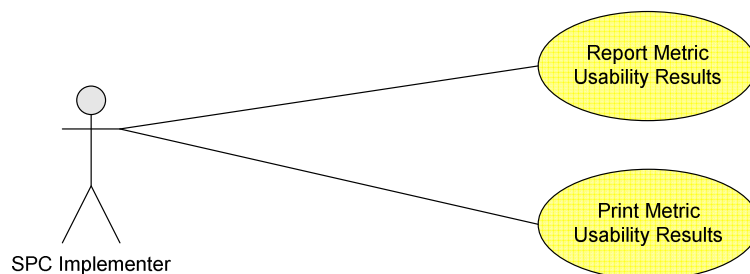


Figure 24 Metric Usability Assessment Results Use cases

Report Metric Usability Results

In the scope of this use case, a report which shows metric name, metric type (base or derived) and usability status (usable or not usable) of all existing process metrics should be generated and displayed to User.

Print Metric Usability Results

User can print the report generated in the *Report Metric Usability Results* use case.

3.3.12 Process Clusters Use cases

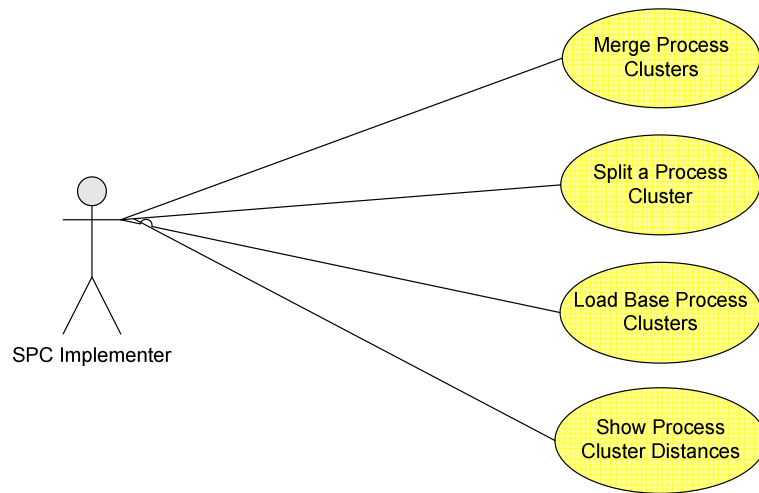


Figure 25 Process Clusters Use cases

Merge Process Clusters

In the scope of this use case, two process clusters can be merged to generate a new process cluster which contains all the process executions of both process clusters by preserving the order of process executions. Process attribute values of new process cluster is the union of the process attribute values of two process clusters merged. The purpose of merge operation is having process clusters which have enough data points for statistical analysis.

Split a Process Cluster

In the scope of this use case, User can split one process cluster into two process clusters which were merged before to create the process cluster being split. Process attribute values of two process clusters should be same as their process attribute values before merge operation. The purpose of split operation is being able to roll back changes done on process clusters. Base process clusters can not be split.

Load Base Process Clusters

In the scope of this use case, User can replace all the process clusters which are created by merge and split operations by base process clusters which are identified according to the current process attribute values of existing process executions.

Show Process Cluster Distances

The number of differing process attribute values between two clusters is called as “cluster distance”. Cluster distance is used to identify the mergable clusters since it is desired to merge process clusters whose differing attributes’ number is minimal.

In the scope of this use case, User can see process cluster distances between all existing process clusters to decide on the process clusters to be merged. Cluster distance triangle can be used for this purpose.

3.3.13 Use cases related with Using SPC Tools

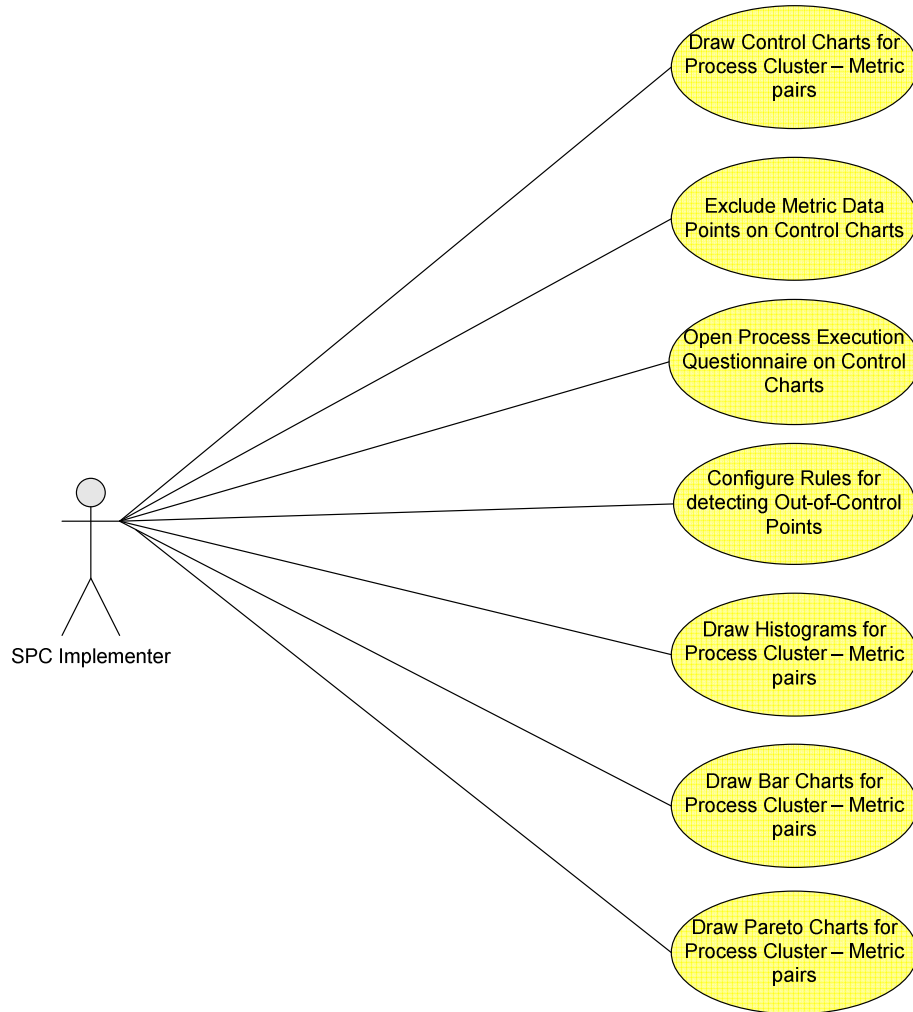


Figure 26 SPC Tools Use cases

Draw Control Charts for Process Cluster – Metric pairs

In the scope of this use case, User can draw control chart for each “process cluster – process metric” pair except ordinal and nominal type process metrics. Configuration done with *Configure Rules for detecting Out-of-Control Points* use case should be used to detect OCPs.

Draw Bar Charts for Process Cluster – Metric pairs

In the scope of this use case, User can draw bar chart for each “process cluster – process metric” pair except absolute and ratio type process metrics.

Draw Histograms for Process Cluster – Metric pairs

In the scope of this use case, User can draw histogram for each “process cluster – process metric” pair except ordinal and nominal type process metrics.

Draw Pareto Charts for Process Cluster – Metric pairs

In the scope of this use case, User can draw pareto chart for each “process cluster – process metric” pair except absolute and ratio type process metrics.

Exclude Metric Data Points on Control Charts

In the scope of this use case, User can exclude metric values of process executions which are detected as OCP on a control chart shown. The purpose of this functionality is not to include metric values of Out-of-Control points in the statistical analysis.

Open Process Execution Questionnaire on Control Charts

In the scope of this use case, User can directly reach (from a control chart shown) and update process execution questionnaire of a process execution which is detected as OCP. The purpose of this use case is to help User for detecting abnormalities in the execution of the process. Process execution questionnaire is filled for each Out-of-Control Point detected during retrospective analysis.

Configure Rules for detecting Out-of-Control Points

When detecting OCPs on control charts, the following four tests are applied by default:

- 1 point > 3 standard deviations from center line
- 9 points in a row on same side of center line
- 2 out of 3 points > 2 standard deviations from center line (same side)
- 4 out of 5 points > 1 standard deviation from center line (same side)

In the scope of this use case, User can choose the tests to be applied when detecting OCPs. Therefore, these tests chosen should be used in any place where calculations about OCPs are involved.

3.3.14 Process Control Status Use cases

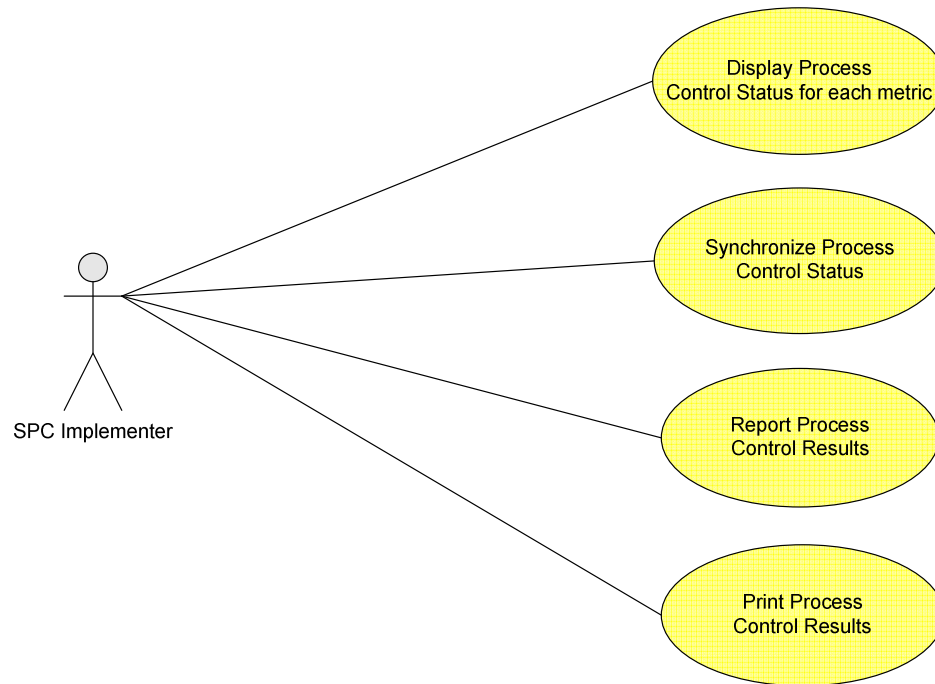


Figure 27 Process Control Status Use cases

Display Process Control Status for each metric

In the scope of this use case, User can see control status of process cluster – process metric (but only ratio and absolute type) pairs. Control status of process clusters for ordinal and nominal process metrics is displayed as N/A.

Synchronize Process Control Status

In the scope of this use case, control status of process cluster – process metric pairs should be updated automatically according to the changes done on process

clusters, metric data values and configuration rules for OCPs. Execution of the following use cases will trigger this use case:

- *Configure Rules for detecting Out-of-Control Points*
- *Exclude Metric Data Points on Control Charts*
- *Load Base Process Clusters*
- *Split a Process Cluster*
- *Merge Process Clusters*
- *Exclude Process Metric Data Points*
- *Update Process Metric Data*
- *Import Metric Data to Workspace*

Report Process Control Results

In the scope of this use case, a report which shows control status of all existing “process cluster – process metric” pairs (Under Control or Out of Control), process metric names and process clusters’ names should be generated and displayed to User.

Print Process Control Results

User can print the report generated in the *Report Process Control Results* use case.

3.3.15 Use cases related with Out-of-Control Points

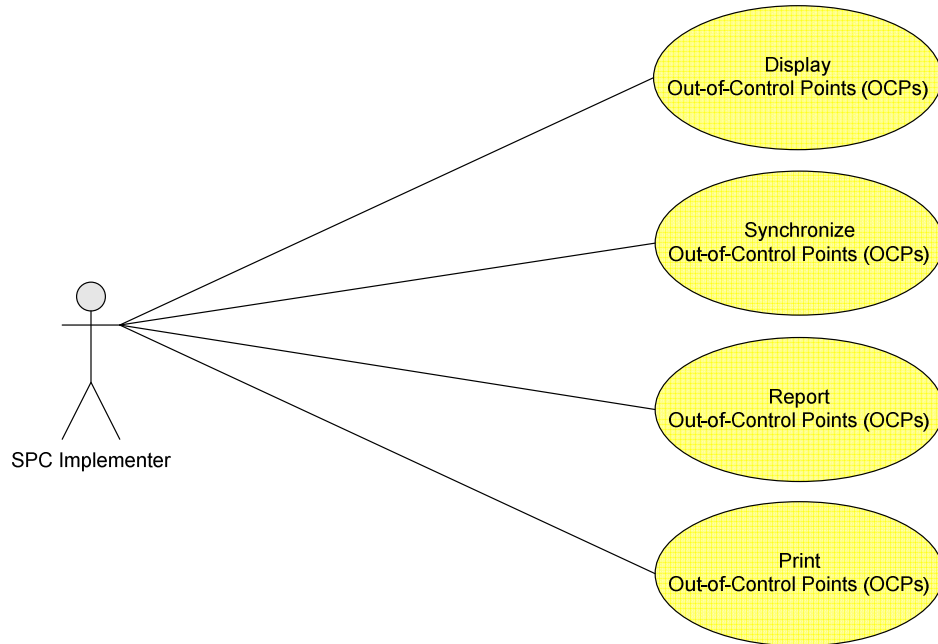


Figure 28 Out-of-Control Points Use cases

Display Out-of-Control Points (OCPs)

In the scope of this use case, it is aimed to see more details about “process cluster – process metric” (but only ratio and absolute type) pairs whose control status is Out of Control. User can see the number of OCPs and reasons for OCPs for all “process cluster – process metric” (but only ratio and absolute type) pairs.

Synchronize Out-of-Control Points (OCPs)

In the scope of this use case, information about OCPs of process “cluster – process metric” pairs should be updated automatically according to the changes done on process clusters, metric data values, process execution questionnaires and configuration rules for OCPs. Execution of the following use cases will trigger this use case:

- *Configure Rules for detecting Out-of-Control Points*
- *Exclude Metric Data Points on Control Charts*
- *Load Base Process Clusters*

- *Split a Process Cluster*
- *Merge Process Clusters*
- *Exclude Process Metric Data Points*
- *Update Process Metric Data*
- *Import Metric Data to Workspace*
- *Update an existing Process Execution Questionnaire*

Report Out-of-Control Points (OCPs)

In the scope of this use case, a report which shows number of OCPs and reasons of OCPs for all existing “process cluster – process metric” pairs besides process metric names and process clusters’ names should be generated and displayed to User.

Print Out-of-Control Points (OCPs)

User can print the report generated in the *Report Out-of-Control Points (OCPs)* use case.

3.3.16 Help Use cases

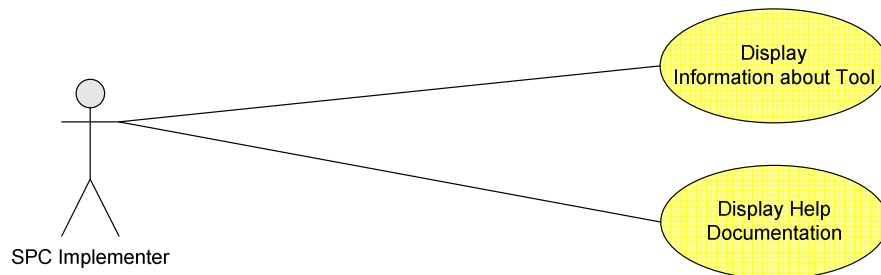


Figure 29 Help Use cases

Display Information about Tool

In the scope of this use case, User can see general information about the tool. Functionality should be similar to “About” dialogs of commercial applications.

Display Help Documentation

In the scope of this use case, User can open help documentation when he needs some information about the usage of the tool.

3.4. Design of SPC-AAT

3.4.1 Architecture

There are nine logical components that constitute SPC-AAT application. They are shown as packages in the figure below.

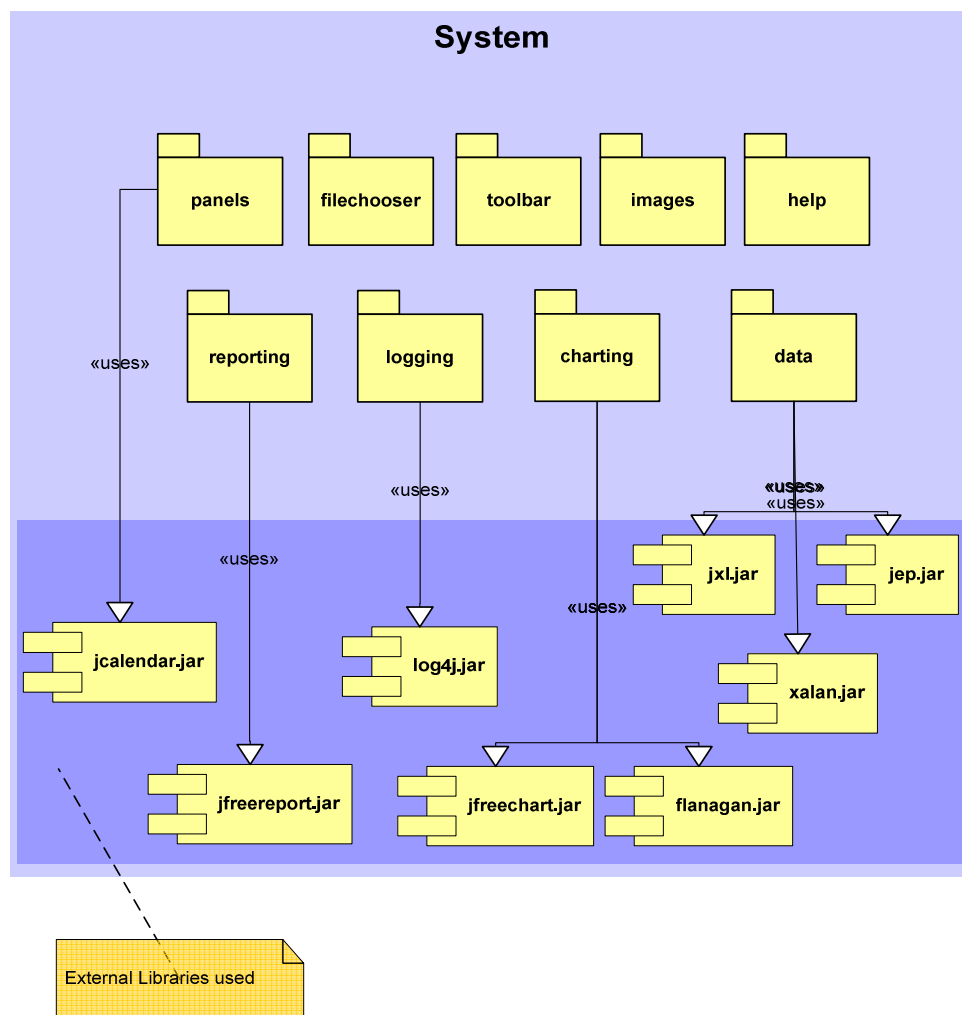


Figure 30 Component Diagram

Namely, these logical components are panels, filechooser, toolbar, images, help, reporting, logging, charting and data.

3.4.2 Detailed Descriptions

This section presents brief overviews of the basic classes. These classes and the relations between them are shown in the class diagram below.

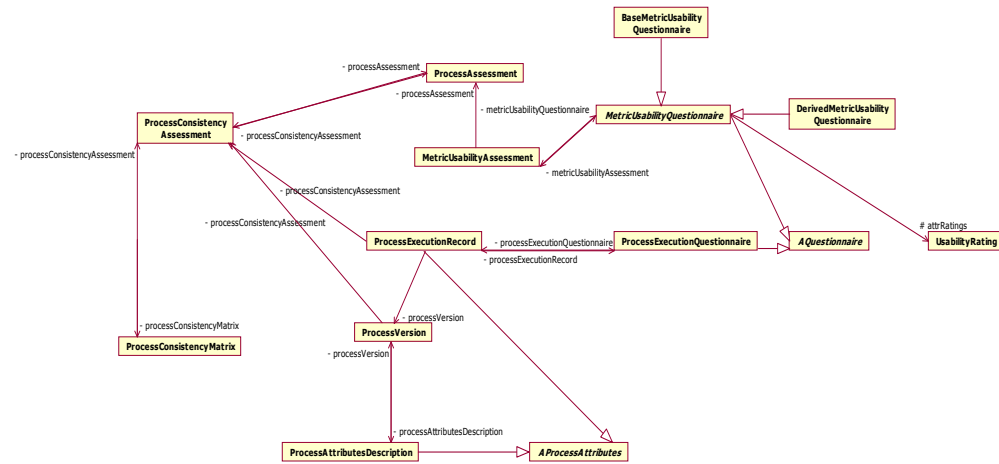


Figure 31 Class Diagram

The responsibility of each class, the collaborations with other classes is also explained below:

ProcessAssessment Class

The *ProcessAssessment* class represents a process which is to be analyzed statistically by using SPC-AAT. All classes related with a process are accessed over this class. ProcessConsistencyAssessment and MetricUsabilityAssessment classes are instantiated by ProcessAssessment Class. A ProcessAssessment object has the following attributes: processName, isRetrospective, metricData,

nonNumericMetricData, processConsistencyAssessment, metricUsabilityAssessments and rulesForDetectingOCPs.

This class contains methods for handling process metric data, process consistency assessment and metric usability assessment.

ProcessConsistencyAssessment Class

The *ProcessConsistencyAssessment* class represents a consistency assessment of a process which is to be analyzed statistically by using SPC-AAT. All classes related with a process consistency assessment are accessed over this class. ProcessSimilarityMatrix, ProcessExecutionRecord and ProcessVersion classes are instantiated by ProcessConsistencyAssessment class. A ProcessConsistencyAssessment object has the following attributes: processAssessment, processConsistencyMatrix, processExecutionRecords and processVersions.

This class contains methods for adding, removing, updating process execution records; adding, removing, updating process versions; synchronization between PSM and process execution records; creating and updating PSM; calculation of cluster distances; merging, splitting process versions, finding Out-of-Control points.

MetricUsabilityAssessment Class

The *MetricUsabilityAssessment* class represents a metric usability assessment of a process which is to be analyzed statistically by using SPC-AAT. All classes related with a metric usability assessment are accessed over this class. MetricUsabilityQuestionnaire classes are instantiated by MetricUsabilityAssessment class. A MetricUsabilityAssessment object has the following attributes: processAssessment and metricUsabilityQuestionnaire.

This class contains methods for handling metric usability questionnaires, saving a MetricUsabilityAssessment object to XML and creating a MetricUsabilityAssessment object from XML.

AProcessAttributes Class

The *AProcessAttributes* class is an abstract class. *ProcessExecutionRecord* and *ProcessAttributesDescription* classes are inherited from *AProcessAttributes* class. All common attributes and methods of a process cluster and a process execution reside in this class. *AProcessAttributes* class has the following attributes: inputs, outputs, activities, roles, toolsAndTechniques, processName, recordedOn and recordedBy.

This class contains methods for handling all process attributes (inputs, outputs, activities, roles, toolsAndTechniques), saving an *AProcessAttributes* object to XML and creating an *AProcessAttributes* object from XML.

AQuestionnaire Class

The *AQuestionnaire* class is an abstract class. *MetricUsabilityQuestionnaire* and *ProcessExecutionQuestionnaire* classes are inherited from *AQuestionnaire* class. All common attributes and methods of a metric usability questionnaire and a process execution questionnaire reside in this class. *AQuestionnaire* class has no attributes defined.

This class contains abstract methods for handling questions, answers, status and attributes of a questionnaire. Besides abstract methods, this class has concrete methods for saving an *AQuestionnaire* object to XML and creating an *AQuestionnaire* object from XML.

ProcessSimilarityMatrix Class

The *ProcessSimilarityMatrix* class represents the process similarity matrix which is created in the scope of process consistency assessment. A *ProcessSimilarityMatrix* object has the following attributes: processConsistencyAssessment.

This class contains methods for creating, updating and displaying PSM.

ProcessVersion Class

The *ProcessVersion* class represents a process cluster of a process assessment, which is created in the scope of process consistency assessment. All data and operations related with a process version are accessed over this class.

ProcessAttributesDescription classes are instantiated by ProcessVersion class. A ProcessVersion object has the following attributes: mergedProcessVersions, numberOfPEs, processAttributesDescription, processConsistencyAssessment and oldPERsOfMergedProcessVersions.

This class contains methods for handling process attributes descriptions, calculating cluster distance to another process version, handling merged process versions, checking split support of a process version, handling process executions of a process version, saving a ProcessVersion object to XML and creating a ProcessVersion object from XML.

ProcessAttrRowData Class

The *ProcessAttrRowData* class represents a process attribute value of a process execution or a process version. Inputs, outputs, activities, roles and toolsAndTechniques of a process execution or a process version are stored at ProcessAttrRowData objects. A ProcessAttrRowData object has the following attributes: no, name, activityNo, and description.

This class contains methods for handling no, name, activityNo, and description of process attribute values besides methods for saving a ProcessAttrRowData object to XML and creating a ProcessAttrRowData object from XML.

ProcessAttributesDescription Class

The *ProcessAttributesDescription* class represents process attribute descriptions (inputs, outputs, activities, roles, toolsAndTechniques) of a process version, which is created in the scope of process consistency assessment. All data and operations related with process attribute descriptions are accessed over this class. ProcessAttributesDescription is inherited from AProcessAttributes abstract class. A ProcessAttributesDescription object has the following attributes: descriptionVersion and processVersion.

This class contains methods for handling descriptionVersion and processVersion attributes, saving a ProcessAttributesDescription object to XML and creating a ProcessAttributesDescription object from XML.

ProcessExecutionRecord Class

The *ProcessExecutionRecord* class represents a process execution record of a process assessment, which is an instance of the assessed process. All data and operations related with a process execution record are accessed over this class. *ProcessExecutionQuestionnaire* classes are instantiated by *ProcessExecutionRecord* class. A *ProcessExecutionRecord* object has the following attributes: *metricInclusion*, *processConsistencyAssessment*, *processExecutionNo*, *processExecutionQuestionnaire* and *processVersion*.

This class contains methods for handling process execution questionnaires, handling process execution record identifier, handling process execution metric data, saving a *ProcessExecutionRecord* object to XML and creating a *ProcessExecutionRecord* object from XML.

ProcessExecutionQuestionnaire Class

The *ProcessExecutionQuestionnaire* class represents a process execution questionnaire of a process execution which is an instance of the assessed process. All data and operations related with a process execution questionnaires are accessed over this class. *ProcessExecutionQuestionnaire* is inherited from *AQuestionnaire* class. A *ProcessExecutionQuestionnaire* object has the following attributes: *questions*, *attributes*, *answers*, *attrCounts*, *statusArr*, *recordedOn*, *recordedBy*, *emptyList*, *emptyListCellEditor* and *processExecutionRecord*.

This class contains methods for handling questions, attributes, answers, status, cell editor, *recordedOn* and *recordedBy* attributes besides saving a *ProcessExecutionQuestionnaire* object to XML and creating a *ProcessExecutionQuestionnaire* object from XML.

MetricUsabilityQuestionnaire Class

The *MetricUsabilityQuestionnaire* class represents a base or derived process metric usability questionnaire of a process assessment, which is created in the scope of metric usability assessment. All data and operations related with a metric usability questionnaire are accessed over this class. *MetricUsabilityQuestionnaire*

is inherited from AQuestionnaire class. A MetricUsabilityQuestionnaire object has the following attributes: questions, answers, attrCounts, conceptualDefinition, assessedOn, assessedBy, attrRatings, metricName, choicesForAnswersItems, reportItems and metricUsabilityAssessment.

This class contains methods for handling questions, attributes, answers, cell editor, assessedOn, attrRatings, metricName, choicesForAnswersItems, reportItems, conceptualDefinition and assessedBy attributes of metric usability questionnaires besides saving a MetricUsabilityQuestionnaire object to XML and creating a MetricUsabilityQuestionnaire object from XML.

BaseMetricUsabilityQuestionnaire Class

The *BaseMetricUsabilityQuestionnaire* class represents a base metric usability questionnaire of a process assessment, which is created in the scope of metric usability assessment. All data and operations related with a base metric usability questionnaire are accessed over this class. BaseMetricUsabilityQuestionnaire is inherited from MetricUsabilityQuestionnaire class. A BaseMetricUsabilityQuestionnaire object has the following attributes inherited from MetricUsabilityQuestionnaire and instantiated: questions, answers, attrCounts, conceptualDefinition, assessedOn, assessedBy, attrRatings, metricName, choicesForAnswersItems, reportItems and metricUsabilityAssessment.

This class contains methods for creating table model for reporting of metric usability results.

DerivedMetricUsabilityQuestionnaire Class

The *DerivedMetricUsabilityQuestionnaire* class represents a derived metric usability questionnaire of a process assessment, which is created in the scope of metric usability assessment. All data and operations related with a derived metric usability questionnaire are accessed over this class. DerivedMetricUsabilityQuestionnaire is inherited from MetricUsabilityQuestionnaire class. A DerivedMetricUsabilityQuestionnaire object has the following attributes inherited from MetricUsabilityQuestionnaire

and instantiated: questions, answers, attrCounts, conceptualDefinition, assessedOn, assessedBy, attrRatings, metricName, choicesForAnswersItems, reportItems and metricUsabilityAssessment.

This class contains methods for creating table model for reporting of metric usability results besides getting and setting metric formula.

UsabilityRating Class

The *UsabilityRating* class represents rating of a metric usability questionnaire attribute, which is handled in the scope of metric usability assessment. All data and operations related with a usability rating are accessed over this class. A UsabilityRating object has the following attributes: ratingValue, expectedRating and rating.

This class contains methods for handling rating, expected rating and rating value of a metric usability rating besides saving a UsabilityRating object to XML and creating a UsabilityRating object from XML.

3.5. Example Scenario

In this section, we will go over one scenario to show the basic steps taken to use SPC-AAT for statistical analysis. In the example scenario, we will import the tasks planned and tracked by MS Project to SPC-AAT and we will show how this information can be processed and analyzed by our tool. All the steps to be taken are explained in the following sections:

3.5.1 Creating a New Workspace

Choose “**Create Workspace**” menu item.

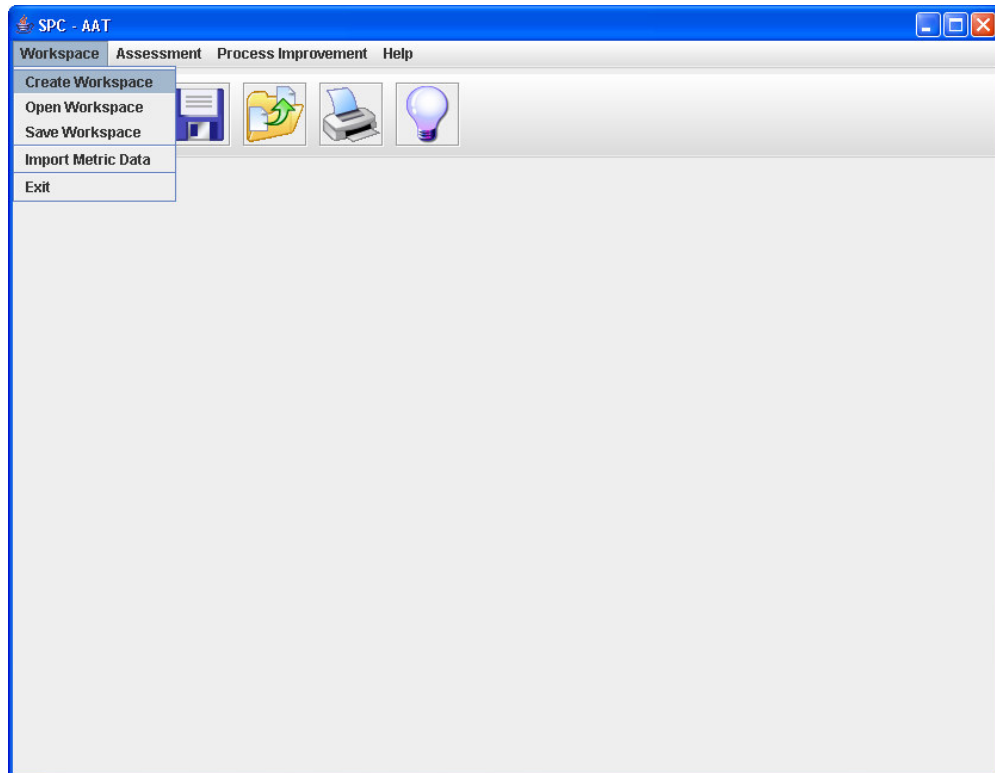


Figure 32 Create Workspace

Enter process name and assessment type and press **OK**.

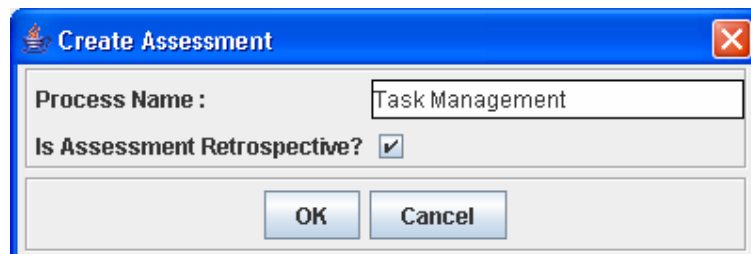


Figure 33 Create Assessment

Workspace is created and by default the context is "**PROCESS DATA**".

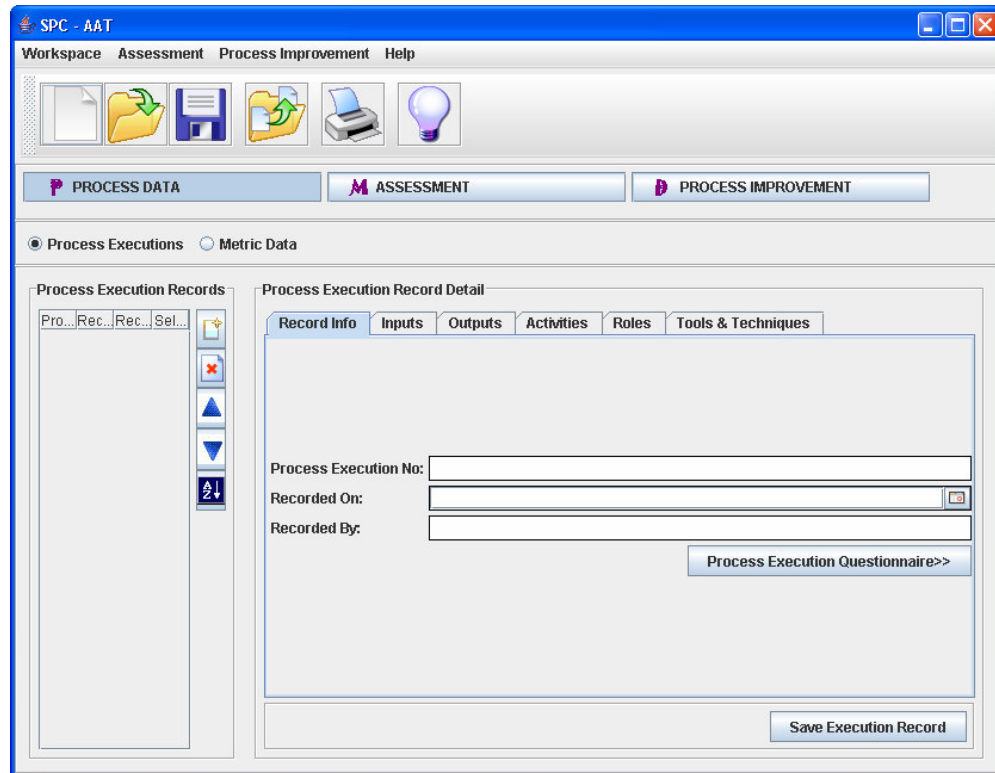


Figure 34 Newly created Workspace

3.5.2 Importing Metric Data

Choose “**Import Metric Data**” menu item.

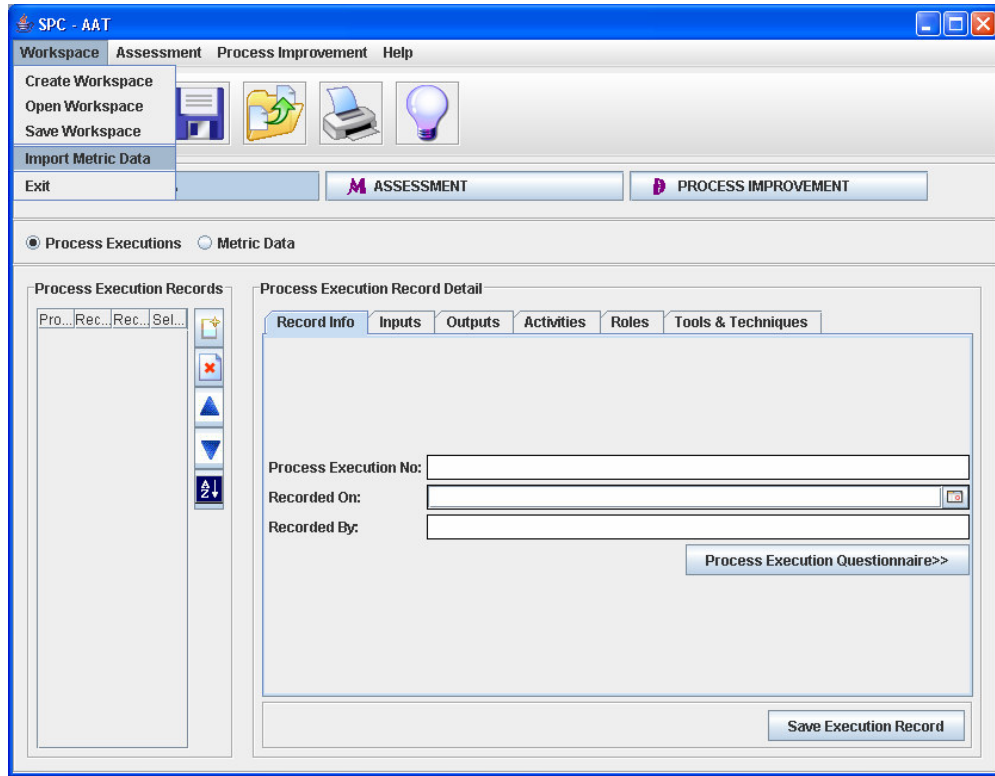


Figure 35 Import Metric Data

Choose the metric data file to be used for export:

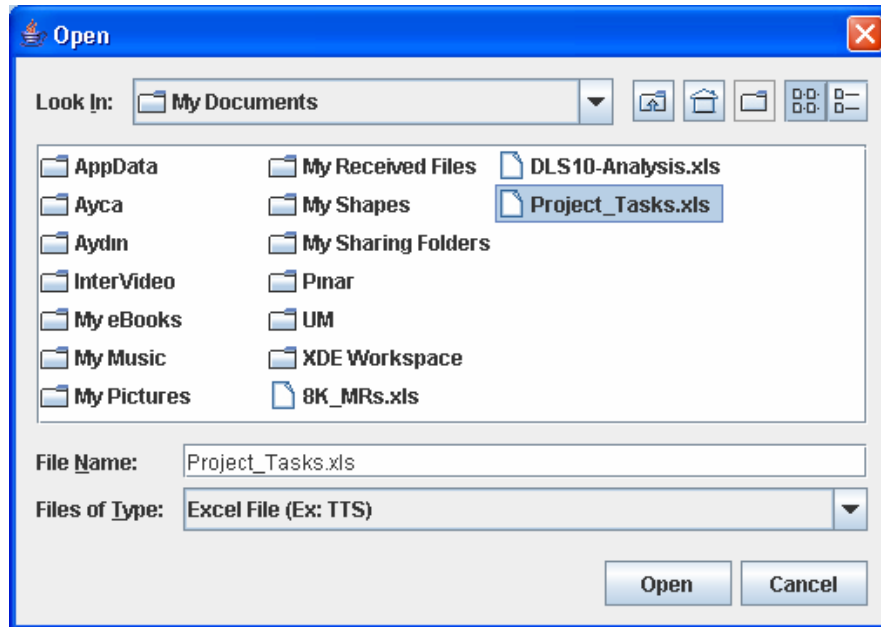


Figure 36 Open Metric Data File

All imported metrics and their values will be shown in a chart:

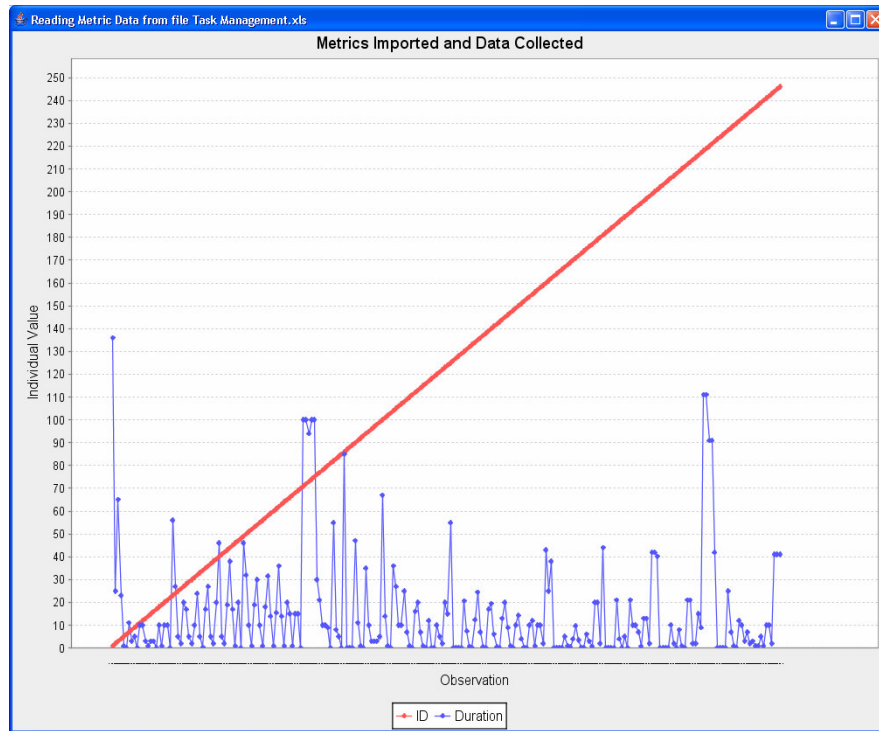


Figure 37 Imported Metrics

Here are the values of imported metrics:

Process Executions	ID	Duration	Name	Start	Finish	Aging	Process ...
1	1	136	Teambase V5.0	Jul 12, 2004	Jan 31, 2005	203	Version A
2	2	25	TB Preliminary ...	Jul 12, 2004	Aug 13, 2004	32	Version A
3	3	65	TB Main	Aug 9, 2004	Nov 10, 2004	93	Version A
4	4	23	TB Main SRS	Aug 9, 2004	Sep 10, 2004	32	Version A
5	5	1	TB Main SRS R...	Sep 13, 2004	Sep 13, 2004	0	Version A
6	6	0	End of Definition...	Sep 13, 2004	Sep 13, 2004	0	Version A
7	7	11	Prototype Desig...	Sep 14, 2004	Sep 28, 2004	14	Version A
8	8	3	DB Design	Sep 29, 2004	Oct 1, 2004	2	Version A
9	9	5	TB Main Design	Oct 4, 2004	Oct 8, 2004	4	Version A
10	10	0	End of Design P...	Oct 8, 2004	Oct 8, 2004	0	Version A
11	11	10	Implementation	Oct 11, 2004	Oct 22, 2004	11	Version A
12	12	10	Perperation of T...	Oct 11, 2004	Oct 22, 2004	11	Version A
13	13	3	Roles and optio...	Oct 11, 2004	Oct 13, 2004	2	Version A
14	14	1	Call Statistics	Oct 14, 2004	Oct 14, 2004	0	Version A
15	15	3	TT Edit,Save,Ma...	Oct 15, 2004	Oct 19, 2004	4	Version A
16	16	3	KP operations	Oct 20, 2004	Oct 22, 2004	2	Version A
17	17	0	End of Impleme...	Oct 22, 2004	Oct 22, 2004	0	Version A
18	18	10	Test Specification	Oct 11, 2004	Oct 22, 2004	11	Version A
19	19	1	Test Spec Review	Oct 25, 2004	Oct 25, 2004	0	Version A
20	20	10	Tests	Oct 26, 2004	Nov 10, 2004	15	Version A
21	21	10	Bug Fixes	Oct 26, 2004	Nov 10, 2004	15	Version A
22	22	0	FIRST RELEASE	Nov 10, 2004	Nov 10, 2004	0	Version A
23	23	56	TB Supplementa...	Aug 31, 2004	Nov 22, 2004	83	Version A
24	24	27	Multi value comp...	Sep 14, 2004	Oct 20, 2004	36	Version A
25	25	5	MVC SRS	Sep 14, 2004	Sep 20, 2004	6	Version A
26	26	2	MVC SRS Review	Sep 21, 2004	Sep 22, 2004	1	Version A
27	27	20	MVC Design & I...	Sep 23, 2004	Oct 20, 2004	27	Version A
28	28	17	Messaging (+Ale...	Sep 14, 2004	Oct 6, 2004	22	Version A
29	29	5	Messaging SRS	Sep 14, 2004	Sep 20, 2004	6	Version A
30	30	2	Messaging SRS ...	Sep 21, 2004	Sep 22, 2004	1	Version A
31	31	10	Messaging Desi...	Sep 23, 2004	Oct 6, 2004	13	Version A

Figure 38 Metric Data

And all **Process Execution Records** and **Metric Usability Assessments** will be generated automatically. Here is **Process Execution Records**:

The screenshot displays the SPC - AAT software interface. The main workspace is divided into two sections: a table of Process Execution Records and a detailed view of a selected record.

Process Execution Records Table:

Process Execution	Recorded On	Recorded By	Selected
1	Jan 18, 2007		<input checked="" type="checkbox"/>
2	Jan 18, 2007		<input checked="" type="checkbox"/>
3	Jan 18, 2007		<input checked="" type="checkbox"/>
4	Jan 18, 2007		<input checked="" type="checkbox"/>
5	Jan 18, 2007		<input checked="" type="checkbox"/>
6	Jan 18, 2007		<input checked="" type="checkbox"/>
7	Jan 18, 2007		<input checked="" type="checkbox"/>
8	Jan 18, 2007		<input checked="" type="checkbox"/>
9	Jan 18, 2007		<input checked="" type="checkbox"/>
10	Jan 18, 2007		<input checked="" type="checkbox"/>
11	Jan 18, 2007		<input checked="" type="checkbox"/>
12	Jan 18, 2007		<input checked="" type="checkbox"/>
13	Jan 18, 2007		<input checked="" type="checkbox"/>
14	Jan 18, 2007		<input checked="" type="checkbox"/>
15	Jan 18, 2007		<input checked="" type="checkbox"/>
16	Jan 18, 2007		<input checked="" type="checkbox"/>
17	Jan 18, 2007		<input checked="" type="checkbox"/>
18	Jan 18, 2007		<input checked="" type="checkbox"/>
19	Jan 18, 2007		<input checked="" type="checkbox"/>
20	Jan 18, 2007		<input checked="" type="checkbox"/>
21	Jan 18, 2007		<input checked="" type="checkbox"/>
22	Jan 18, 2007		<input checked="" type="checkbox"/>
23	Jan 18, 2007		<input checked="" type="checkbox"/>
24	Jan 18, 2007		<input checked="" type="checkbox"/>
25	Jan 18, 2007		<input checked="" type="checkbox"/>
26	Jan 18, 2007		<input checked="" type="checkbox"/>
27	Jan 18, 2007		<input checked="" type="checkbox"/>
28	Jan 18, 2007		<input checked="" type="checkbox"/>
29	Jan 18, 2007		<input checked="" type="checkbox"/>
30	Jan 18, 2007		<input checked="" type="checkbox"/>
31	Jan 18, 2007		<input checked="" type="checkbox"/>
32	Jan 18, 2007		<input checked="" type="checkbox"/>
33	Jan 18, 2007		<input checked="" type="checkbox"/>
34	Jan 18, 2007		<input checked="" type="checkbox"/>
35	Jan 18, 2007		<input checked="" type="checkbox"/>
36	Jan 18, 2007		<input checked="" type="checkbox"/>
37	Jan 18, 2007		<input checked="" type="checkbox"/>
38	Jan 18, 2007		<input checked="" type="checkbox"/>

Process Execution Record Detail Panel:

The detail panel is titled "Process Execution Record Detail" and contains several tabs: Record Info, Inputs, Outputs, Activities, Roles, and Tools & Techniques. The "Record Info" tab is currently selected, showing the following fields:

- Process Execution No.: [Empty text box]
- Recorded On: Jan 18, 2007
- Recorded By: [Empty text box]

Below these fields is a button labeled "Process Execution Questionnaire>>". At the bottom of the panel is a button labeled "Save Execution Record".

Figure 39 Process Execution Records created

Here is **Metric Usability Assessments**:

SPC - AAT

Workspace Assessment Process Improvement Help

PROCESS DATA ASSESSMENT PROCESS IMPROVEMENT

Consistency Assessment Metrics Evaluation

Base Metrics Derived Metrics

Process Metrics

Metric Name	Metric Usability
ID	Not Usable
Duration	Not Usable
Name	Not Usable
Start	Not Usable
Finish	Not Usable

Metric Usability Assessment Detail

General Info Questionnaire Usability Rating

Metric Name:

Conceptual Definition:

Assessed On:

Assessed By:

Save Metric Usability Assessment

Figure 40 Metric Usability Assessments created

3.5.3 Update Process Execution Records

Enter the desired info for the newly created Process Execution Records (PERs).

Fill “Recorded On”, “Recorded By” and “Process Execution No” fields on “Record Info” tab:

SPC - AAT

Workspace Assessment Process Improvement Help

PROCESS DATA ASSESSMENT PROCESS IMPROVEMENT

Process Executions Metric Data

Process Execution Records

Process Execution	Recorded On	Recorded By	Selected
1	Jan 18, 2007		✓
2	Jan 18, 2007		✓
3	Jan 18, 2007		✓
4	Jan 18, 2007		✓
5	Jan 18, 2007		✓
6	Jan 18, 2007		✓
7	Jan 18, 2007		✓
8	Jan 18, 2007		✓
9	Jan 18, 2007		✓
10	Jan 18, 2007		✓
11	Jan 18, 2007		✓
12	Jan 18, 2007		✓
13	Jan 18, 2007		✓
14	Jan 18, 2007		✓
15	Jan 18, 2007		✓
16	Jan 18, 2007		✓
17	Jan 18, 2007		✓
18	Jan 18, 2007		✓
19	Jan 18, 2007		✓
20	Jan 18, 2007		✓
21	Jan 18, 2007		✓
22	Jan 18, 2007		✓
23	Jan 18, 2007		✓
24	Jan 18, 2007		✓
25	Jan 18, 2007		✓
26	Jan 18, 2007		✓
27	Jan 18, 2007		✓
28	Jan 18, 2007		✓
29	Jan 18, 2007		✓
30	Jan 18, 2007		✓
31	Jan 18, 2007		✓
32	Jan 18, 2007		✓
33	Jan 18, 2007		✓
34	Jan 18, 2007		✓
35	Jan 18, 2007		✓
36	Jan 18, 2007		✓
37	Jan 18, 2007		✓
38	Jan 18, 2007		✓
39	Jan 18, 2007		✓
40	Jan 18, 2007		✓
41	Jan 18, 2007		✓

Process Execution Record Detail

Record Info Inputs Outputs Activities Roles Tools & Techniques

Process Execution No:

Recorded On:

Recorded By:

Process Execution Questionnaire>>

Save Execution Record

Figure 41 Process Execution Record Details

Then add **Input** entries for the newly created PERs on “**Inputs**” tab. Create a new Input entry by **right clicking** on the table and choose “**Insert New Entry**” menu item or just click on the “**New**” button at the right side of the table:

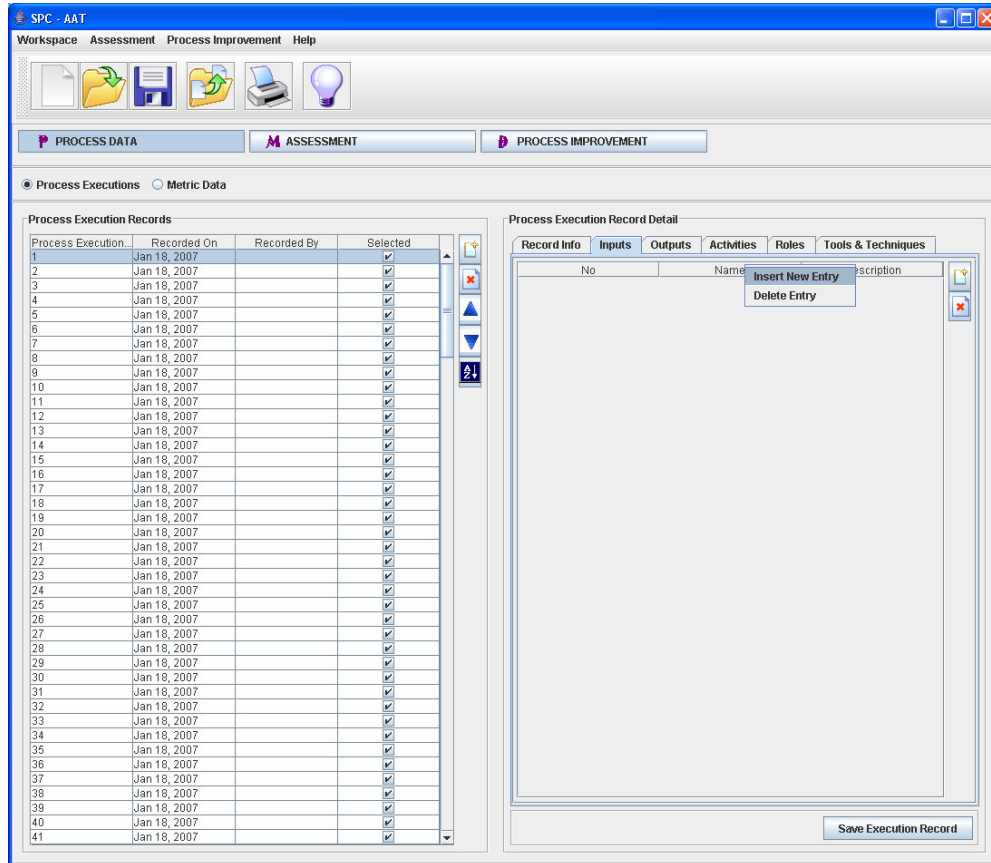


Figure 42 Inserting new Inputs

Enter the desired info for the newly created Input entry:

The screenshot displays the SPC - AAT software interface. The main window has a menu bar with 'Workspace', 'Assessment', 'Process Improvement', and 'Help'. Below the menu bar is a toolbar with icons for file operations and a lightbulb icon. The main area is divided into two tabs: 'PROCESS DATA' and 'ASSESSMENT'. Under 'PROCESS DATA', there are two sub-tabs: 'Process Executions' (selected) and 'Metric Data'.

The 'Process Executions' tab shows a table with the following columns: 'Process Execution', 'Recorded On', 'Recorded By', and 'Selected'. The table contains 41 rows, all with 'Jan 18, 2007' in the 'Recorded On' column and a checkmark in the 'Selected' column.

To the right of the table is the 'Process Execution Record Detail' form. It has a tabbed interface with 'Record Info', 'Inputs', 'Outputs', 'Activities', 'Roles', and 'Tools & Techniques'. The 'Record Info' tab is active, showing a table with the following columns: 'No', 'Name', and 'Description'. The table contains one row with '1' in the 'No' column, 'Task Definition' in the 'Name' column, and 'Definition of the task assigned' in the 'Description' column. Below the table is a 'Save Execution Record' button.

Figure 43 Input created

Follow the same steps of **Inputs** for **Outputs**, **Activities**, **Roles** and **Tools&Techniques**:

The screenshot displays the SPC - AAT software interface. The main window has a menu bar with 'Workspace', 'Assessment', 'Process Improvement', and 'Help'. Below the menu bar is a toolbar with icons for file operations and a lightbulb icon. The main area is divided into two sections: 'PROCESS DATA' and 'ASSESSMENT'. The 'ASSESSMENT' section is active, showing 'Process Executions' and 'Metric Data' tabs. The 'Process Executions' tab is selected, displaying a table of 'Process Execution Records'.

Process Execution	Recorded On	Recorded By	Selected
1	Jan 18, 2007		✓
2	Jan 18, 2007		✓
3	Jan 18, 2007		✓
4	Jan 18, 2007		✓
5	Jan 18, 2007		✓
6	Jan 18, 2007		✓
7	Jan 18, 2007		✓
8	Jan 18, 2007		✓
9	Jan 18, 2007		✓
10	Jan 18, 2007		✓
11	Jan 18, 2007		✓
12	Jan 18, 2007		✓
13	Jan 18, 2007		✓
14	Jan 18, 2007		✓
15	Jan 18, 2007		✓
16	Jan 18, 2007		✓
17	Jan 18, 2007		✓
18	Jan 18, 2007		✓
19	Jan 18, 2007		✓
20	Jan 18, 2007		✓
21	Jan 18, 2007		✓
22	Jan 18, 2007		✓
23	Jan 18, 2007		✓
24	Jan 18, 2007		✓
25	Jan 18, 2007		✓
26	Jan 18, 2007		✓
27	Jan 18, 2007		✓
28	Jan 18, 2007		✓
29	Jan 18, 2007		✓
30	Jan 18, 2007		✓
31	Jan 18, 2007		✓
32	Jan 18, 2007		✓
33	Jan 18, 2007		✓
34	Jan 18, 2007		✓
35	Jan 18, 2007		✓
36	Jan 18, 2007		✓
37	Jan 18, 2007		✓
38	Jan 18, 2007		✓
39	Jan 18, 2007		✓
40	Jan 18, 2007		✓
41	Jan 18, 2007		✓

The 'Process Execution Record Detail' panel is open, showing a table with columns: 'No', 'Name', 'Act_No', and 'Description'. The first row shows '1' in the 'No' column and 'Project Manager' in the 'Name' column. The 'Save Execution Record' button is visible at the bottom right of the panel.

Figure 44 Roles of a Process Execution Record

When all the related info is entered for the Process Execution Record, “**Save Execution Record**” button is pressed to save the changes. This button can also be used for updating PERs.

The screenshot displays the SPC - AAT software interface. The main window has a menu bar (Workspace, Assessment, Process Improvement, Help) and a toolbar with icons for file operations. Below the toolbar are three tabs: PROCESS DATA, ASSESSMENT, and PROCESS IMPROVEMENT. The PROCESS DATA tab is active, showing a sub-tab for Process Executions. On the left, a table titled 'Process Execution Records' lists 41 records. Each record has columns for 'Process Execution...', 'Recorded On', 'Recorded By', and 'Selected'. The 'Recorded On' column shows dates from Jan 18, 2007 to Jan 19, 2007. The 'Selected' column has checkboxes, all of which are checked. On the right, the 'Process Execution Record Detail' form is visible. It has a tabbed interface with 'Record Info' selected. The 'Record Info' tab contains fields for 'Process Execution No:', 'Recorded On:', and 'Recorded By:'. The 'Recorded On:' field is set to 'Jan 18, 2007' and the 'Recorded By:' field is set to 'Serkan Kirbaş'. Below these fields is a button labeled 'Process Execution Questionnaire>>'. At the bottom right of the form, there is a button labeled 'Save Execution Record', which is highlighted with a red circle.

Process Execution...	Recorded On	Recorded By	Selected
1	Jan 18, 2007		✓
2	Jan 18, 2007		✓
3	Jan 18, 2007		✓
4	Jan 18, 2007		✓
5	Jan 18, 2007		✓
6	Jan 18, 2007		✓
7	Jan 18, 2007		✓
8	Jan 18, 2007		✓
9	Jan 18, 2007		✓
10	Jan 18, 2007		✓
11	Jan 18, 2007		✓
12	Jan 18, 2007		✓
13	Jan 18, 2007		✓
14	Jan 18, 2007		✓
15	Jan 18, 2007		✓
16	Jan 18, 2007		✓
17	Jan 18, 2007		✓
18	Jan 18, 2007		✓
19	Jan 18, 2007		✓
20	Jan 18, 2007		✓
21	Jan 18, 2007		✓
22	Jan 18, 2007		✓
23	Jan 18, 2007		✓
24	Jan 18, 2007		✓
25	Jan 18, 2007		✓
26	Jan 18, 2007		✓
27	Jan 18, 2007		✓
28	Jan 18, 2007		✓
29	Jan 18, 2007		✓
30	Jan 18, 2007		✓
31	Jan 18, 2007		✓
32	Jan 18, 2007		✓
33	Jan 18, 2007		✓
34	Jan 18, 2007		✓
35	Jan 18, 2007		✓
36	Jan 18, 2007		✓
37	Jan 18, 2007		✓
38	Jan 18, 2007		✓
39	Jan 18, 2007		✓
40	Jan 18, 2007		✓
41	Jan 18, 2007		✓

Figure 45 Saving Process Execution Record

3.5.4 Operations on Process Similarity Matrix

Then Process Similarity Matrix can be seen by clicking on “ASSESSMENT” toggle button and then on “Consistency Assessment” radio button. So context is changed to “ASSESSMENT” from “PROCESS DATA”.

Below “Process Similarity Matrix” can be seen for **Inputs**:

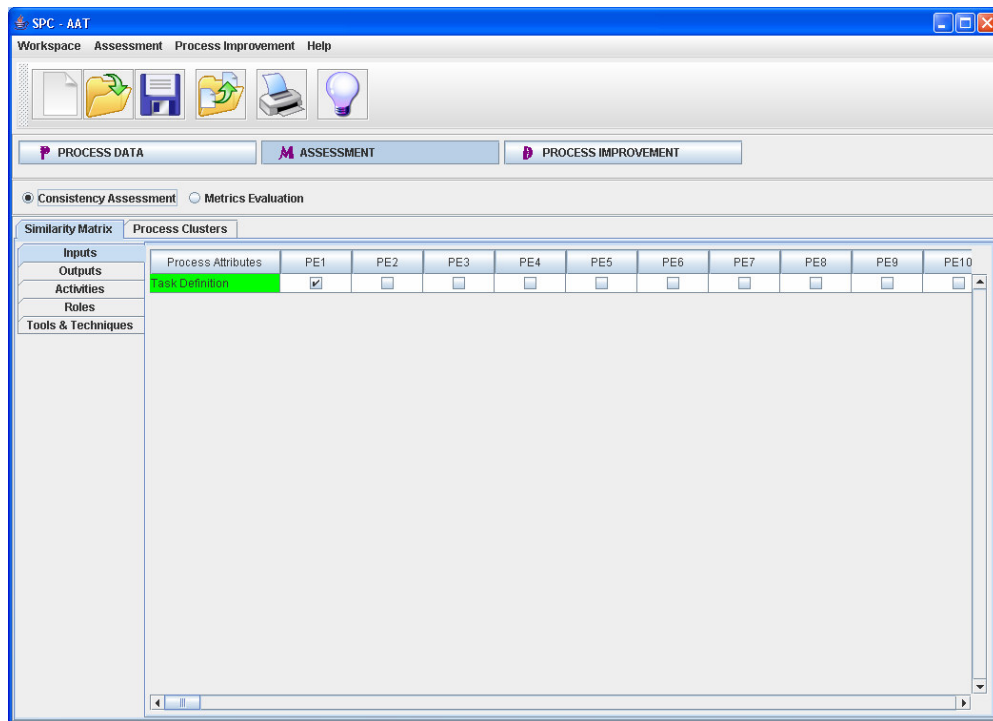


Figure 46 Process Similarity Matrix for Inputs

Adding and deleting “**Inputs**” entries to the Process Executions can be done here by updating the checkboxes accordingly.

Follow the same steps of **Inputs** for **Outputs**, **Activities**, **Roles** and **Tools&Techniques**.

3.5.5 Identifying Base Process Clusters

Base Process Clusters can be seen by clicking on “**ASSESSMENT**” toggle button and then on “**Consistency Assessment**” radio button and to “**Process Clusters**” tab-sheet. To re-identify the base process clusters, choose “**Re-identify Process Clusters**” menu item:

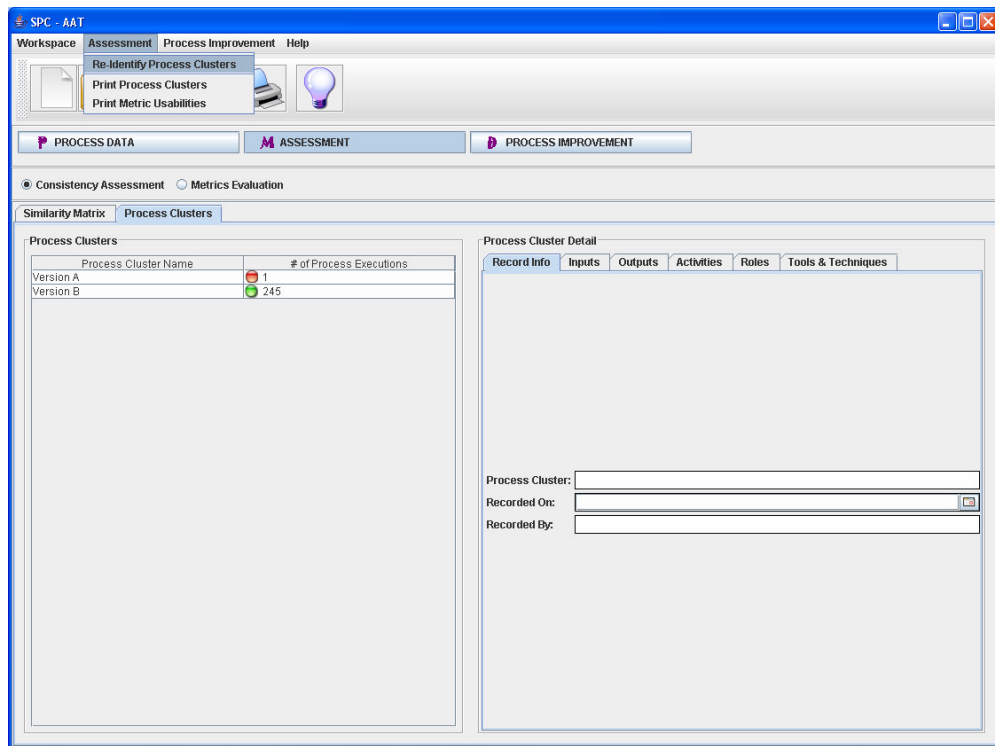


Figure 47 Re-identify Process Clusters

3.5.6 Handling Metrics

We can switch to “Metric Usability Assessment” by clicking on “ASSESSMENT” toggle button and then on “Metrics Evaluation” radio button.

So context is changed to “Metrics Evaluation”:

ID	Metric Name	Metric Usability
Duration		Not Usable
Name		Not Usable
Start		Not Usable
Finish		Not Usable

Metric Usability Assessment Detail

General Info | Questionnaire | Usability Rating

Metric Name:

Conceptual Definition:

Assessed On:

Assessed By:

Figure 48 Base Metrics

“Metric Name”, “Conceptual Definition”, “Assessed On” and “Assessed By” fields on “General Info” tab can be updated here.

Then click on the “**Questionnaire**” tab to answer the questions and rate the metrics:

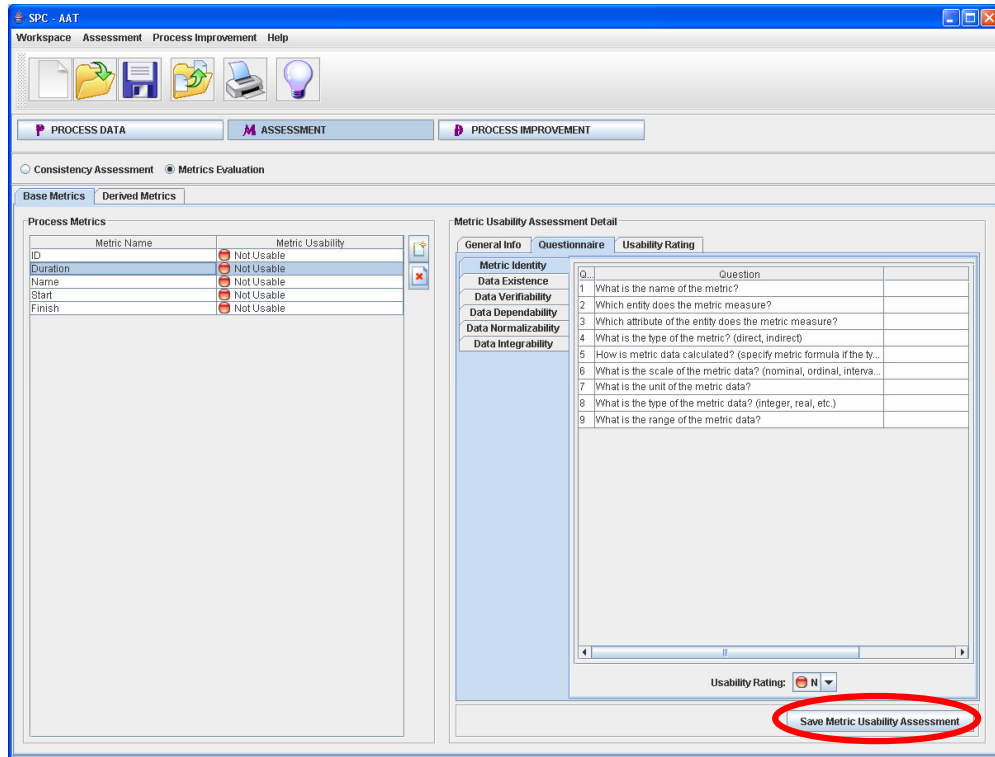


Figure 49 Base Metric Questionnaire

When all the related info is entered for the metrics, press “**Save Metric Usability Assessment**” to save the changes.

In this way, enter the info and fill the questionnaires for all base and derived metrics.

3.5.7 Displaying Usability Results for Metrics

Usability results of the process metrics for statistical analysis can be seen by clicking on the “**Print**” toolbar icon and choosing “**Metric Usability Evaluation Report**” check box in the panel shown:

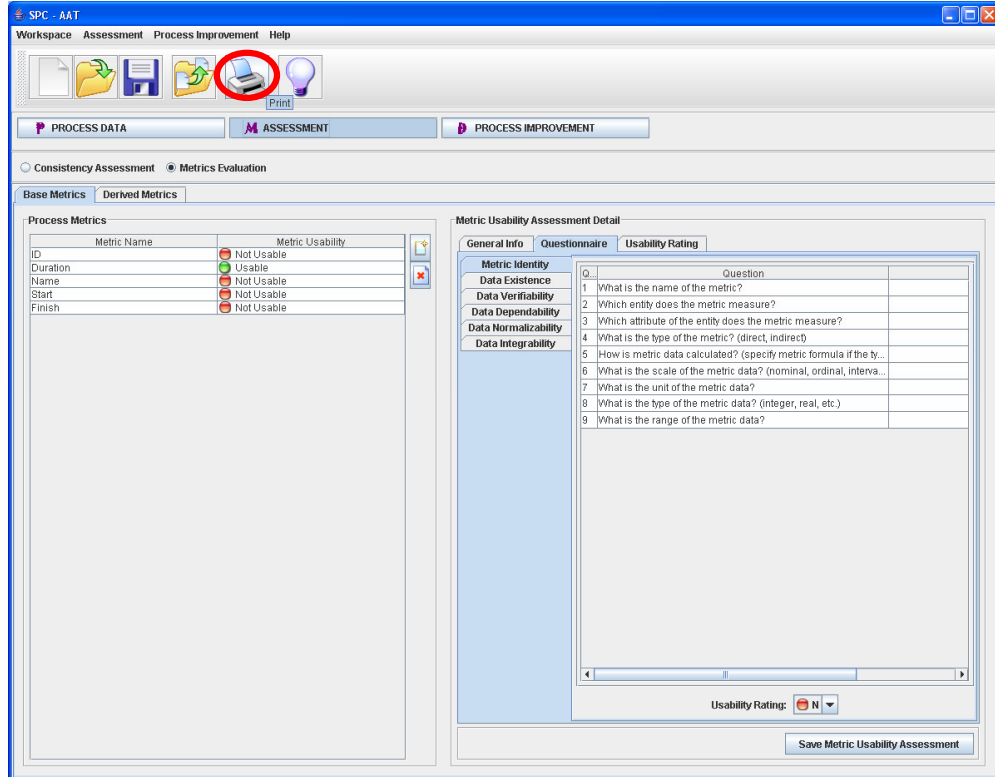


Figure 50 Reporting Metric Usability Evaluation

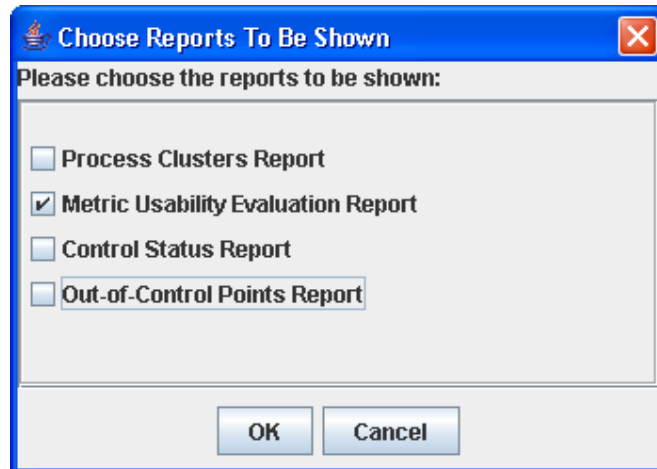


Figure 51 Four Kinds of Reports

Below is the report generated:

Metric Usability Evaluation Report - Print Preview

File Navigation Zoom Help

100 %

Metric Name	Type	Metric Usability
ID	Base	NOT Usable
Duration	Base	Usable
Name	Base	NOT Usable
Start	Base	NOT Usable
Finish	Base	NOT Usable
Aging	Derived	Usable

Page 1 of 1

Figure 52 Metric Usability Evaluation Report

3.5.8 Operations on Process Clusters

Process Clusters can be seen by clicking on “**PROCESS IMPROVEMENT**” toggle button and then on “**Process Clusters**” radio button. To load the base process clusters, choose “**Load Base Process Clusters**” menu item under Process Improvement menu.

When we right click on a Process Cluster, 6 menu items are shown:

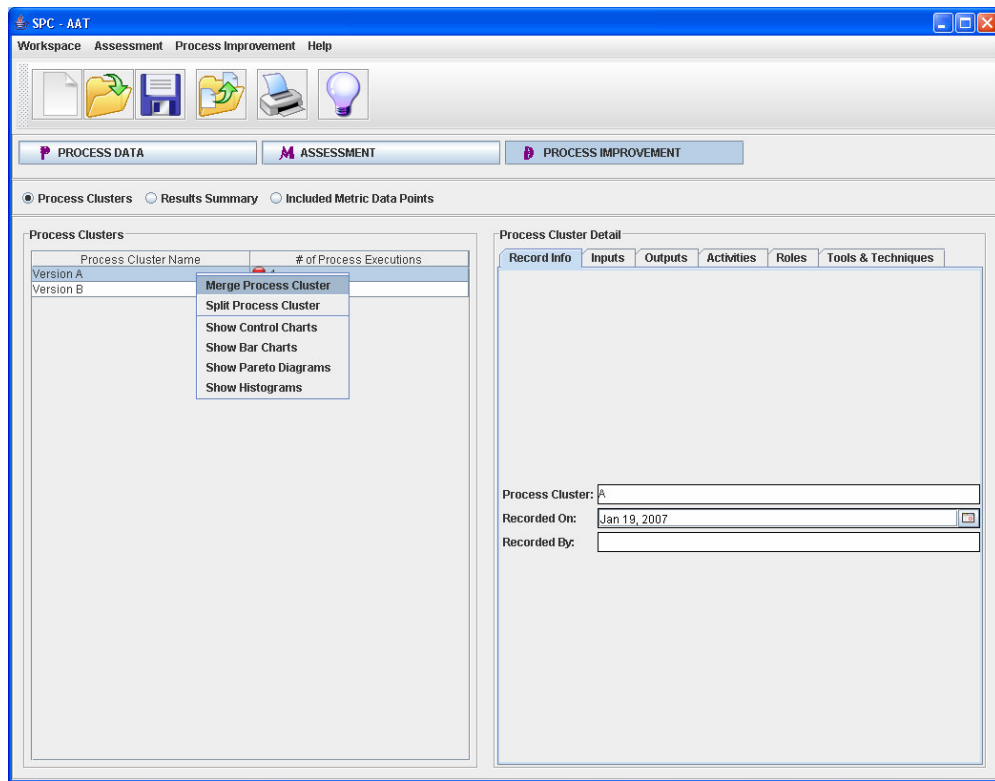


Figure 53 Menu Items on a Process Cluster

“**Merge Process Cluster**” is used to merge two different process clusters.

“**Split Process Cluster**” is used to split one process cluster into two process clusters which were merged before into one.

3.5.9 Using SPC Tools Supported

“Show Control Charts”, “Show Bar Charts”, “Show Pareto Diagrams”, “Show Histograms” are the menu items that are used to apply SPC tools on “process cluster – metric” pairs:

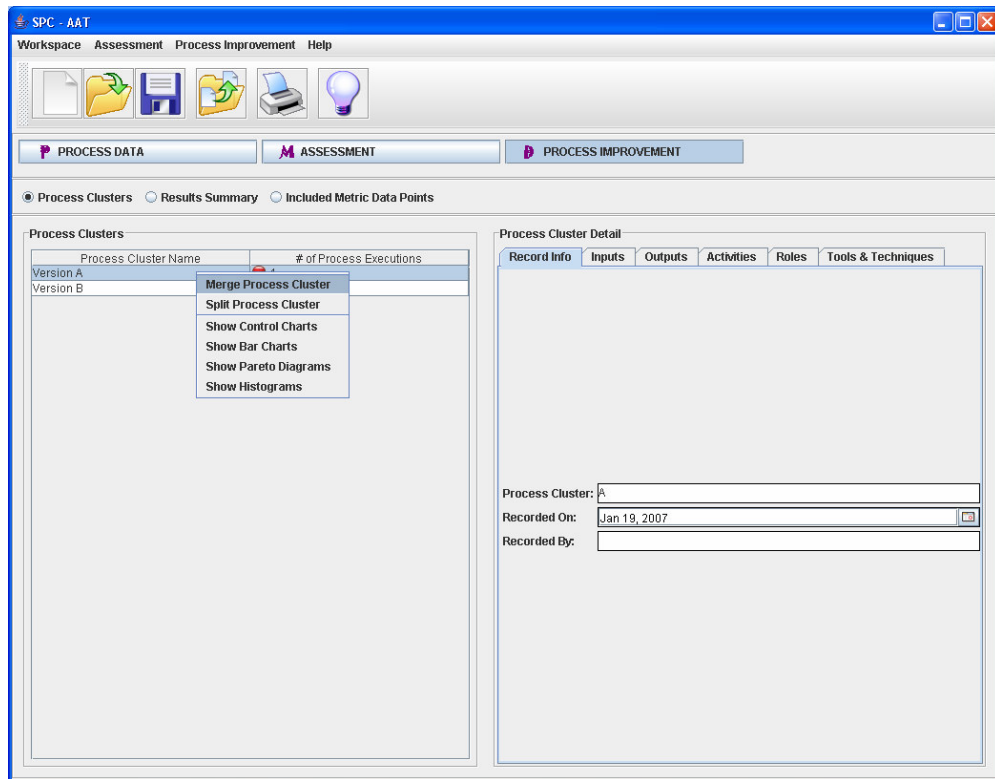


Figure 54 SPC Tools supported

When “Show Control Charts” is chosen, a panel is opened that asks for the metrics to draw control charts for the chosen Process Cluster:

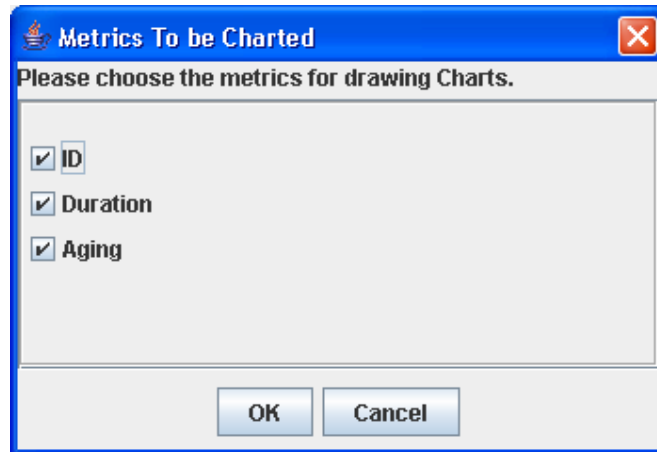


Figure 55 Choosing Metrics to be Charted

Then if OK is clicked, **Control Charts** are drawn:

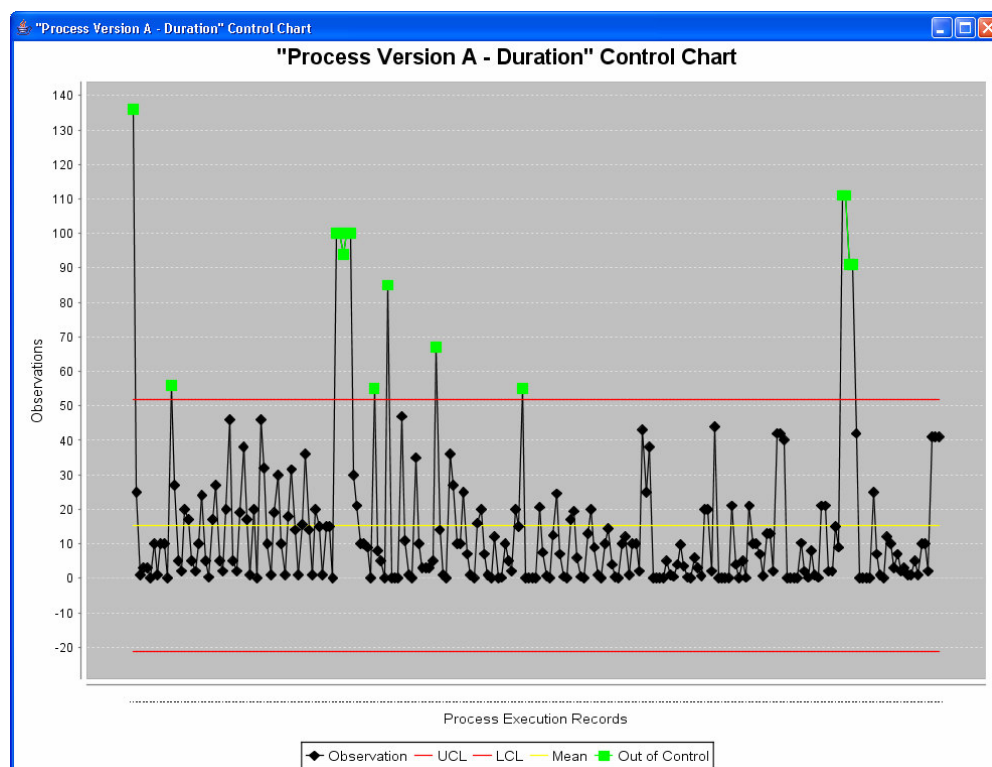


Figure 56 A Control Chart drawn

The green points are the out-of-control points detected for the process analyzed. When we click on an out-of-control point, a questionnaire (**Process Execution Questionnaire**) is shown to detect the reason for this point by answering the questions:

Q...	Question	Statu...	
4	Has there been a recent change in location?	<input type="checkbox"/>	
5	Has there been a recent change in support systems? (infrastr...	<input type="checkbox"/>	
6	Has there been a recent change in communication channels ...	<input type="checkbox"/>	
7	Has there been a recent change in funding and resources all...	<input type="checkbox"/>	
8	Has the process been tailored for this specific execution?	<input type="checkbox"/>	

Figure 57 Process Execution Questionnaire

Then it is asked whether to remove the point from the analysis or not:

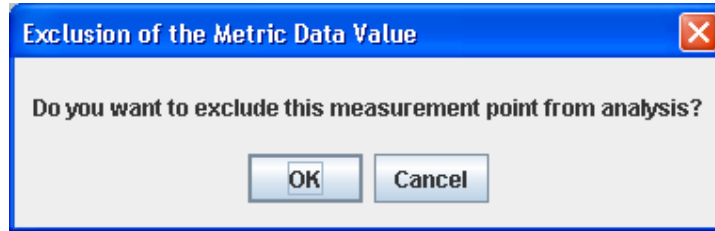


Figure 58 Exclusion of a Metric Value

3.5.10 Overall Process Control Results

Overall results for process control can be seen by clicking on “**PROCESS IMPROVEMENT**” toggle button and then on “**Results Summary**” radio button:

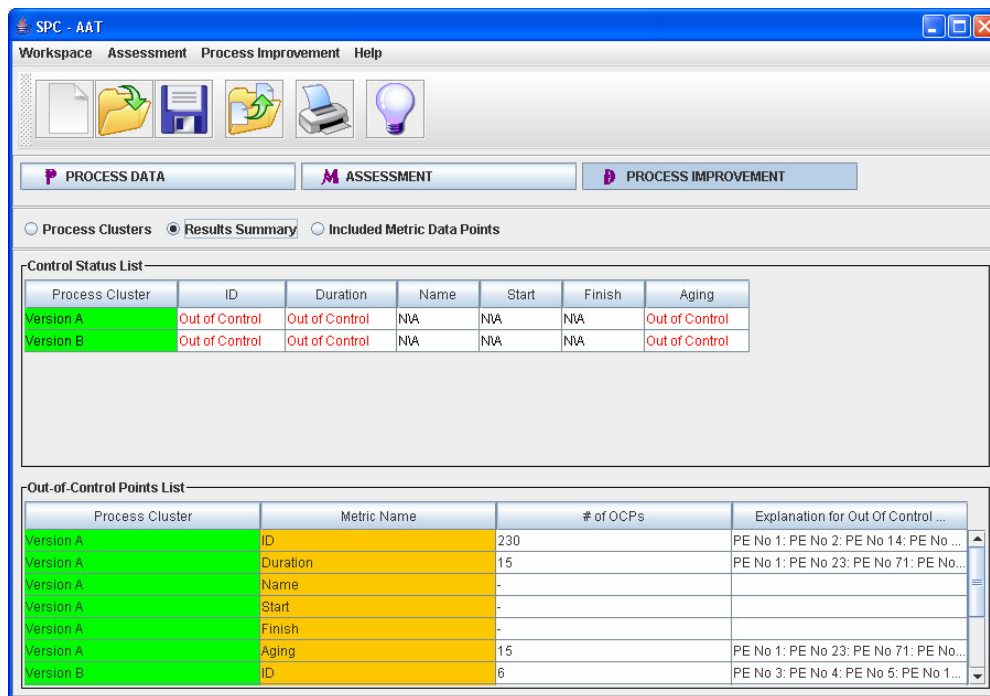


Figure 59 Assessment Results Summary

Reports can be generated to show the results of the assessment and the analysis:

Control Status Report - Print Preview

File Navigation Zoom Help

100 %

Process Cluster: Version A		6
Metric Name	ControlStatus	
ID	Out of Control	
Duration	Out of Control	
Name	N/A	
Start	N/A	
Finish	N/A	
Aging	Out of Control	

Process Cluster: Version B		6
Metric Name	ControlStatus	
ID	Out of Control	
Duration	Out of Control	
Name	N/A	
Start	N/A	
Finish	N/A	
Aging	Out of Control	

Page 1 of 1

Figure 60 Control Status Report

CHAPTER 4

CASE STUDY

4.1.Fundamentals of the Case

We designed our applications as a multiple-case study, and identified our unit of analysis as “process-metric” pair. In our multiple-case study design, we decided that every case would include more than one unit of analysis.

We used the following criteria while selecting the cases among nominations:

- Historical process execution: at least 20-25 metric data points are required;
- Accessibility of performers of historical process executions: performers will be interviewed during the assessment of process consistency;
- If there is no historical data, ability of the process to generate 20-25 metric data points in the near future;
- Availability of process performers to participate in the assessment.

To validate SPC-AAT, we implemented three case studies at a project-based working software organization (referred as organization X in the study) having CMMI L3. We worked on recruitment and bug fixing processes (for two different projects) of organization X and related metrics of these processes. These processes and the metrics used in the case studies can be seen in Table 2.

Table 2 Processes and Metrics used in the Case Studies

Process Name	Metric Name	Comments
Bug Fixing (Project A)	Creation Date	Base Metric
	Resolution Date	Base Metric
	Bug Aging	Derived Metric SPC Tools used: Control Chart - Histogram
	Person Hours (Effort)	Base Metric SPC Tools used: Control Chart - Histogram
	Status	Base Metric SPC Tools used: Bar Chart
Recruitment	Go Date	Base Metric
	Due date	Base Metric
	Start Date	Base Metric
	Planned Procurement Time	Derived Metric
	Actual Procurement Time	Derived Metric SPC Tools used: Control Chart - Histogram
	Procurement Time Variance	Derived Metric SPC Tools used: Control Chart - Histogram
	Position	Base Metric SPC Tools used: Bar Chart, Pareto Chart
Bug Fixing (Project B)	Creation Date	Base Metric
	Actual Finish Date	Base Metric
	Estimated Finish Date	Base Metric
	Bug Aging	Derived Metric SPC Tools: Control Chart - Histogram
	Estimated Bug Aging	Derived Metric
	Estimation Variance	Derived Metric
	Estimation Capability	Base Metric
	Error Reason	Base Metric SPC Tools: Bar Chart, Pareto Chart
	Problem Source	Base Metric SPC Tools: Bar Chart, Pareto Chart
	Should-be found	Base Metric SPC Tools: Bar Chart, Pareto Chart
	Status	Base Metric SPC Tools: Bar Chart

Detailed information about the organization and the projects is given below:

Context-1 (project A in organization X): Project A is a software project in the domain of communication technologies. Basically Java technologies are used for development. There are 8 technical personnel working for project A.

Context-2 (organization X): Organization X which was founded in 2001 is a project-based working software organization having CMMI L3. Basically projects in the area of communication technologies are held by the organization. There are about 80 technical personnel working for organization X. It is experiencing a period of improvement since its foundation and currently aiming to achieve CMMI L4.

Context-3 (project B in organization X): Project B is a software project in the domain of communication technologies. Basically Java technologies are used for development. There are 5 technical personnel working for project B.

4.2. Context-1 (Case Study A)

Within the first context, we worked on bug fixing (MR solving) process of a telecom project. We especially focused on one feature of the project. We chose the bug fixing sub-process because this part of the development process was being heavily used and we could find a lot of instances of this sub-process to analyze. Bugs had been reported during component integration tests, system tests or operation at customer side. Bugs found during unit tests are not included in the study. Bug fixing process had been performed by using a change management tool which is used organization-wide. Bug fixing for the selected feature is analyzed for 3 months period, from September 2006 to December 2006.

Bug fixing process starts when a bug is reported by a submitter via the change management tool or e-mail directly. Bugs found in the project are usually entered to the tool by a Submitter. The submitter should enter the following fields: subject, problem description, project name, responsible person, priority, status. Creation date and unique ID are automatically created by the tool for the bug reported. Then the responsible person (Developer or Feature Owner) gets an e-

mail notification automatically from the tool and starts the analysis. If bug found is not entered to the tool, the content of bug report is up to the Submitter. Developer sends an e-mail to the Submitter if there is missing information about the bug reported. When necessary information is received, analysis of the problem is started by Developer. At the end of analysis, there can be two results. In one of them, we decide that the bug reported is not a real bug, it is a problem related with the usage of the application. As the other possibility, we decide that it is a problem related with our application and start to solve it. After deciding on a solution, implementation starts and implemented solution is tested by the Developer. Then changes are checked-in to the configuration management tool and integrated by using CruiseControl. After this step, if bug is entered to the change management tool, the state of the bug is changed to “solved” in the change management tool and resolution test is entered to explain the solution implemented. After these steps taken by the developer, an e-mail notification is automatically sent to the submitter by the tool. If bug is not reported by the tool, Developer sends an e-mail to the Submitter manually. The whole process is defined as an eEPC diagram in Figure 61.

The analysis was performed together with the owner (Developer) of the related feature of the project. We spent 1 hour for collecting data and 4.5 hours for applying the approach, performing the analyses, and interpreting the results. In other words, we spent 5.5 hours for whole analysis. The set of assets produced during this case study with the control charts are provided in Appendix C.

The study was retrospective, and instead of identifying process attribute values to put on process similarity matrices by filling process execution records, we preferred drawing general process flows with the Developer. We depicted the executions as draft on a paper first together with the Developer, and then converted the flows into eEPC (Extended Event Driven Process Change) diagram by using MS Visio. The flow for bug fixing process is given in Figure 61.

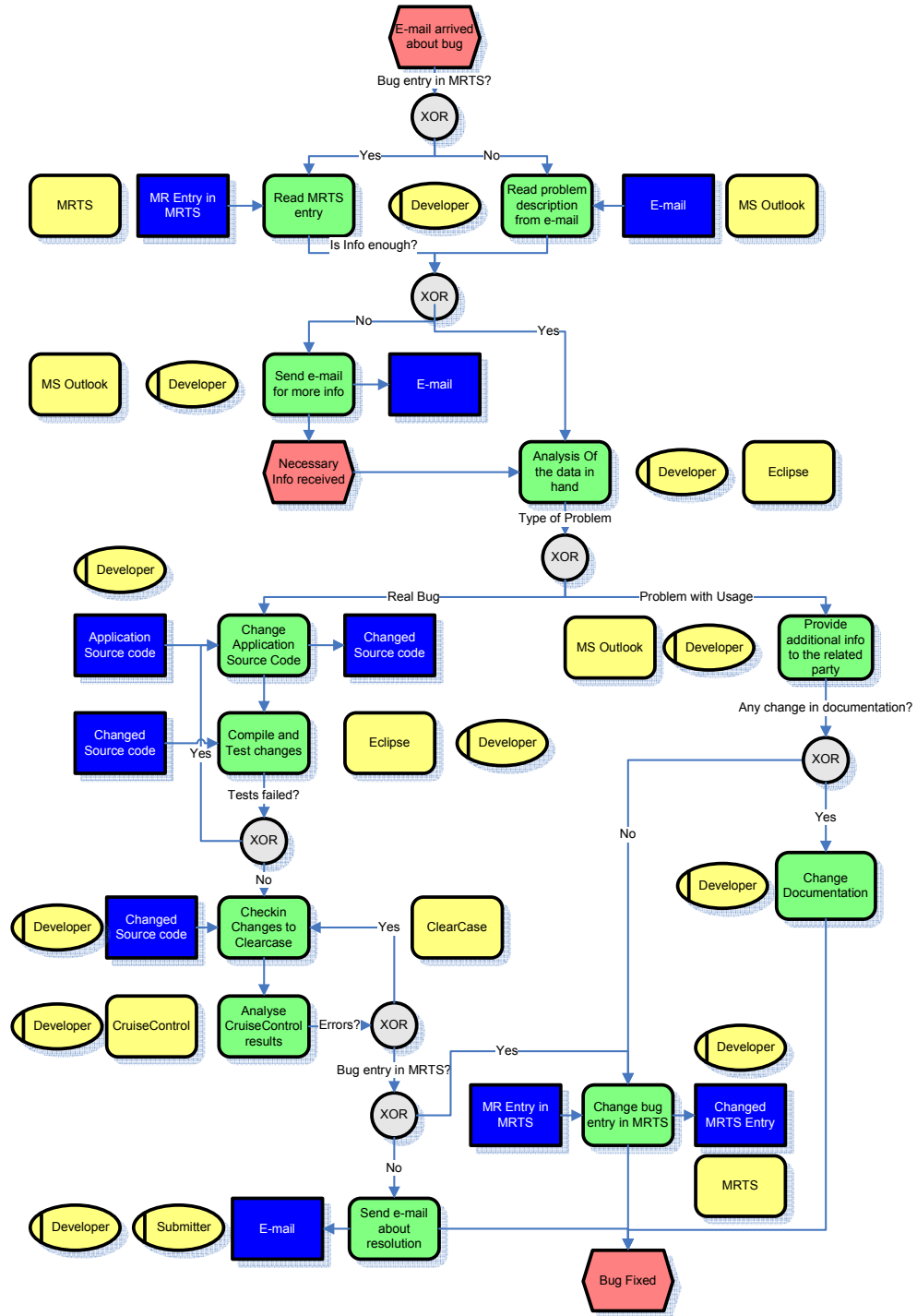


Figure 61 eEPC for Bug Fixing Process (Case A)

For the bugs which are reported via CM tool, the metric data had been exported in an MS Excel file. Metric data is exported in the time order from CM tool since this is very critical for statistical analysis. Then we added metric data for the bugs reported by e-mail into the file. For these bugs, we checked the e-mails sent to the Developer against the bugs reported for the analyzed feature. We found 9 bug reports notified via e-mail directly from the Submitter. We read the creation date of bug reports from the Received Date field of e-mails in MS Outlook. The Excel file which holds info about all bugs can be seen in Appendix C. We first imported this metric data to our tool. Therefore, one process execution record is automatically created for each bug report entry in the Excel file imported. There were 42 data points which were collected as bug reports. Then by using the process elements (inputs, outputs, activities, roles, and tools) on the eEPC diagram, we have added the values of process attributes for inputs, outputs, activities, roles, and tools & techniques on the process similarity matrix. Therefore we could add Process Attributes on the similarity matrix and checked against process executions. The process similarity matrix for bug fixing process executions is provided in Figure 62.

We had 33 process execution records for bugs reported via CM tool and 9 process execution records for bugs reported via e-mail and this yielded 42 process execution records totally. So that we completed process similarity matrices for Inputs, Outputs, Activities, Roles, and Tools & Techniques of all bug fixing process instances.

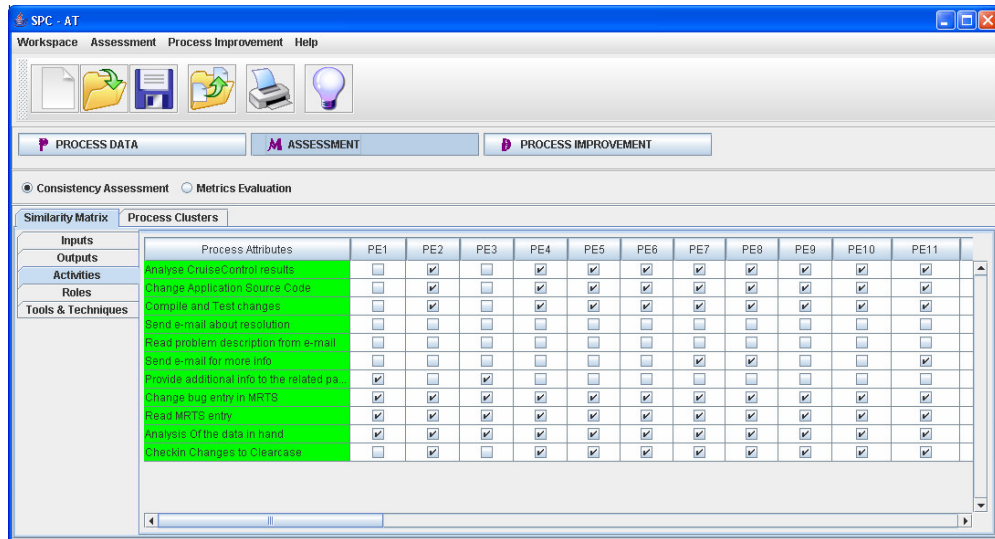


Figure 62 Process Similarity Matrix for Bug Fixing (Case A)

After finalizing the process similarity matrices, we checked under “Process Clusters” tab-sheet to see the automatically identified process clusters by our tool. Our tool identified 5 process clusters labeled as “Version A”, “Version B”, “Version C”, “Version D”, and “Version E” as shown in the Figure below, by observing the similarities between process executions. The number of data points was enough (at least 20) just for Version B, all other process clusters had data points less then 20.

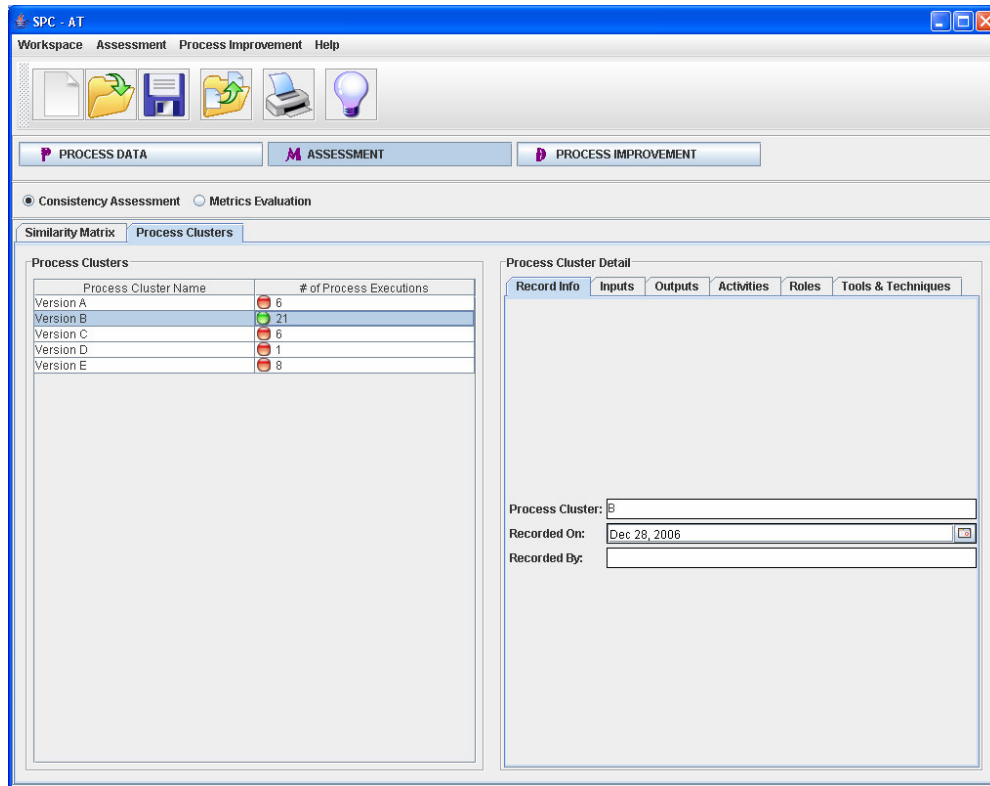


Figure 63 Base Process Clusters for Bug Fixing Process

After we identified initial process clusters, we have changed our view to Metric Evaluation (under ASSESSMENT view) to assess the usability of the process metrics. The metrics which are imported to our tool from Excel at the beginning of the case study are shown in the table below as Base metrics. We decided to derive one new metric with the imported base metrics: Bug Aging. The relationship between the base metrics and Bug Aging is shown visually in Figure 64. The metrics at upper side represent the base metrics.

Table 3 Process Metrics (Original and Derived) for Case A

Metric Name	Metric Type	Explanation
Creation Date	Base	The date bug is reported
Resolution Date	Base	The date bug is resolved

Bug Aging	Derived	Actual Finish Date - Creation Date + 1
Person Hours	Base	Effort spent to fix the bug
Status	Base	Current status of the bug

We created one entry under Derived Metrics tab-sheet for the new derived metric. After filling Metric Formula attribute for the Bug Aging derived metric, metric data was calculated automatically from the entered formula and stored by our tool.

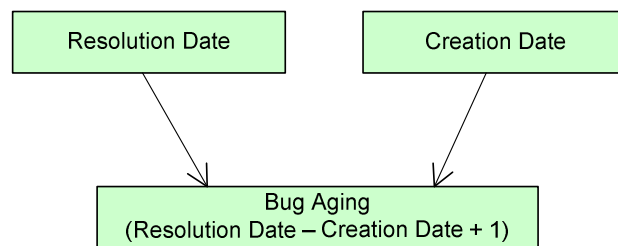


Figure 64 Derived Metrics Identified in Context-1

After deciding on the derived metrics, we filled Metric Usability Questionnaire for each base and derived metric from Questionnaire tab-sheet under Metric Evaluation view. Example questionnaire for “Bug Aging” derived metric and usability ratings given are shown in Figure 65 and Figure 66 (completed questionnaires for all metrics identified in Context-1 are provided in appendix C). The usability status of all base and derived metrics are listed in Figure 67. In the next step, only metrics which are evaluated as “usable” would be used for control charting.

SPC - AT

Workspace Assessment Process Improvement Help

PROCESS DATA ASSESSMENT PROCESS IMPROVEMENT

Consistency Assessment Metrics Evaluation

Base Metrics Derived Metrics

Process Metrics

Metric Name Bug Aging

Metric Usability Assessment Detail

General Info Questionnaire Usability Rating

Metric Identity

Data Existence

Data Verifiability

Data Dependability

Data Normalizability

Data Integrability

Q...	Question	
15	Is metric data stored precisely?	Yes
16	Is metric data stored for a specific purpose?	Yes
17	Is the purpose of metric data storage known by process performer...	Yes
18	Is metric data analyzed and reported?	Yes
19	Is metric data analysis results communicated to process perform...	Yes
20	Is metric data analysis results communicated to management?	Yes
21	Is metric data analysis results used as a basis for decision makin...	Yes

Usability Rating: ☒ F

Save Metric Usability Assessment

Figure 65 Metric Usability Questionnaire for “Bug Aging” Derived Metric of
“Bug Fixing” Process

SPC - AT

Workspace Assessment Process Improvement Help

PROCESS DATA ASSESSMENT PROCESS IMPROVEMENT

Consistency Assessment Metrics Evaluation

Base Metrics Derived Metrics

Process Metrics

Metric Name Bug Aging

Metric Usability Assessment Detail

General Info Questionnaire Usability Rating

Metric Usability Attributes	Rating	Expected Rating
Metric Identity	F	F
Data Existence	F	F
Data Verifiability	F	L
Data Dependability	F	L
MUF-3&4 for Create Date	L	L
MUF-3&4 for Res Date	L	L

Save Metric Usability Assessment

Figure 66 Metric Usability Ratings for “Bug Aging” Derived Metric of “Bug Fixing” Process

Metric Usability Evaluation Report - Print Preview

File Navigation Zoom Help

100 %

Metric Name	Type	Metric Usability
Person Hours	Base	Usable
Create Date	Base	Usable
Sta	Base	Usable
Res Date	Base	Usable
Bug Aging	Derived	Usable

Page 1 of 1

Figure 67 Metric Usability Report for Bug Fixing process

After identifying the base process clusters and assessing the usability of the metrics, we have changed our view to Process Clusters (under PROCESS IMPROVEMENT view) to finalize the process clusters. The number of data points was enough (at least 20) just for Version B, and we decided to identify possible merges between the clusters. We checked Process Cluster Distances Table in Figure 68 and saw that cluster distance value is 1 between Version B & Version C and between Version D & E. Therefore, Version B & Version C are merged into Version B_C and Version D & Version E are merged into Version D_E. Report generated for the new process clusters is shown in Figure 69. When we checked the process attributes, we noticed that Version B_C represents the bug fixing process where bugs are reported via CM tool and Version D & E via e-mail. After these merges, again we checked Process Cluster Distances Table in Figure 70 and we saw that distances between process clusters are very high. Therefore we decided to apply control charting on these process clusters and also the one which is produced by merging these three process clusters.

Process Cluster Distances & Process Attributes							
Cluster Distances							
Cluster Pairs	Distance	Cluster Pairs	Distance	Cluster Pairs	Distance	Cluster Pairs	Distance
Version A-Version B	8						
Version A-Version C	10	Version B-Version C	1				
Version A-Version D	17	Version B-Version D	12	Version C-Version D	11		
Version A-Version E	14	Version B-Version E	11	Version C-Version E	12	Version D-Version E	

Process Attributes					
Inputs	Outputs	Activities	Roles	Tools & Techniques	
Process Attributes	Version A	Version B	Version C	Version D	Version E
Analyse CruiseControl results	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Change Application Source Code	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Compile and Test changes	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Send e-mail about resolution	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Read problem description from e-mail	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Send e-mail for more info	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Provide additional info to the relate	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Change bug entry in MRTS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Read MRTS entry	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Analysis Of the data in hand	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 68 Cluster Distances for base Process Clusters

Process Clusters Report - Print Preview

File Navigation Zoom Help

100 %

Process Cluster Name	# of Process Executions
Version A	6
Version B_C	27
Version D_E	9

Page 1 of 1

Figure 69 Process Clusters after first merge

Process Cluster Distances & Process Attributes

Cluster Distances

Cluster Pairs	Distance	Cluster Pairs	Distance
Version A-Version B_C	10		
Version A-Version D_E	17	Version B_C-Version D_E	11

Process Attributes

Inputs Outputs Activities Roles Tools & Techniques

Process Attributes	Version A	Version B_C	Version D_E
Analyse CruiseControl results	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Change Application Source Code	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Compile and Test changes	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Send e-mail about resolution	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Read problem description from e-...	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Send e-mail for more info	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Provide additional info to the relate...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Change bug entry in MRTS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Read MRTS entry	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Analysis Of the data in hand	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 70 Cluster Distances after first merge

SPC tools could only be applied to the qualified process cluster – metric pairs. In this case, these were “Version B_C – Bug Aging” and “Version B_C – Person Hours” for control charting since just process cluster Version B_C had enough number of metric data points and Bug Aging, Person Hours metrics were assessed as usable and were the type of ratio/absolute. Control charts drawn for these two process cluster – metric pairs are shown in the following figures.

When we checked the control chart drawn for “Version B_C – Bug Aging” pair, we observed that no OCP was detected and process is under control. The mean was about 4 days and Upper Control Limit (UCL) was about 16. According to the control chart results, we conclude that the process is not only under control but also capable since mean and UCL were consistent with the service level agreement of the company.

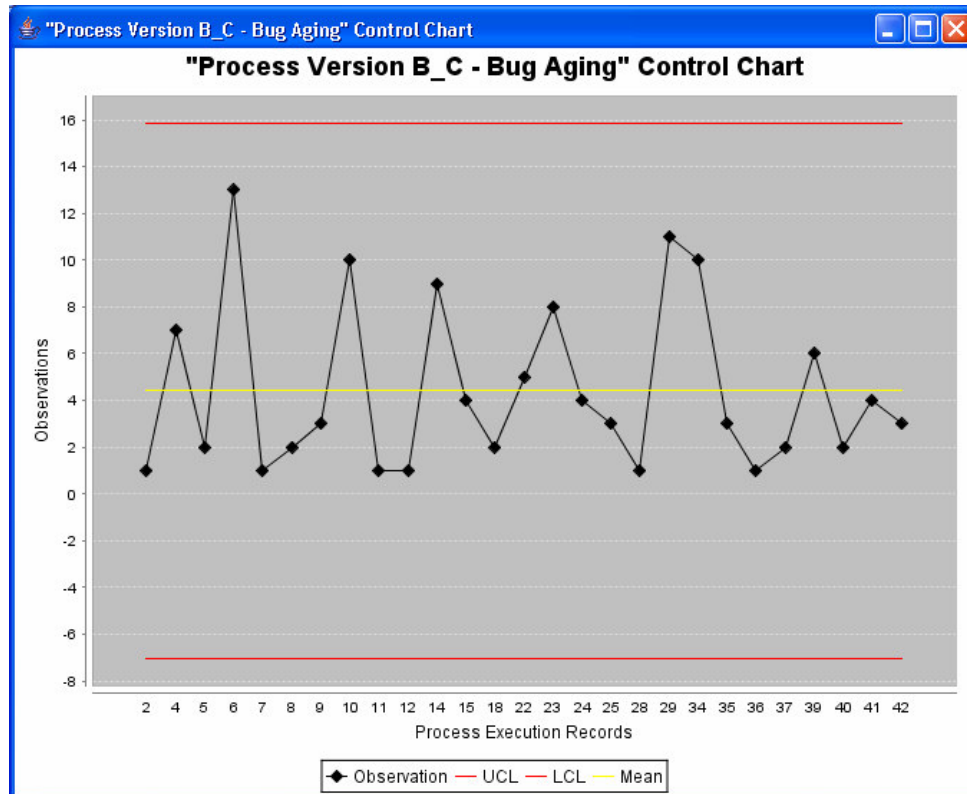


Figure 71 Process Version B_C – Bug Aging Control Chart

When we checked the control chart drawn for “Version B_C – Person Hours” pair, we observed one Out-of-Control Point (OCP). When we checked this OCP detected, we found out that corresponding process instance is not for an error request but for a development request. Therefore we excluded this metric data point from analysis and re-drew the control chart. This time, we detected again one OCP. When we filled Process Execution Questionnaire for the detected OCP, we have found out that for this bug fixing process instance, Developer had not had enough knowledge about the related component and the support from other experienced team members had been weak. We have reported this as a possible improvement point for the project. For example, such bugs can be discussed in more details at the weekly project meetings and one experienced team member can be assigned for support.

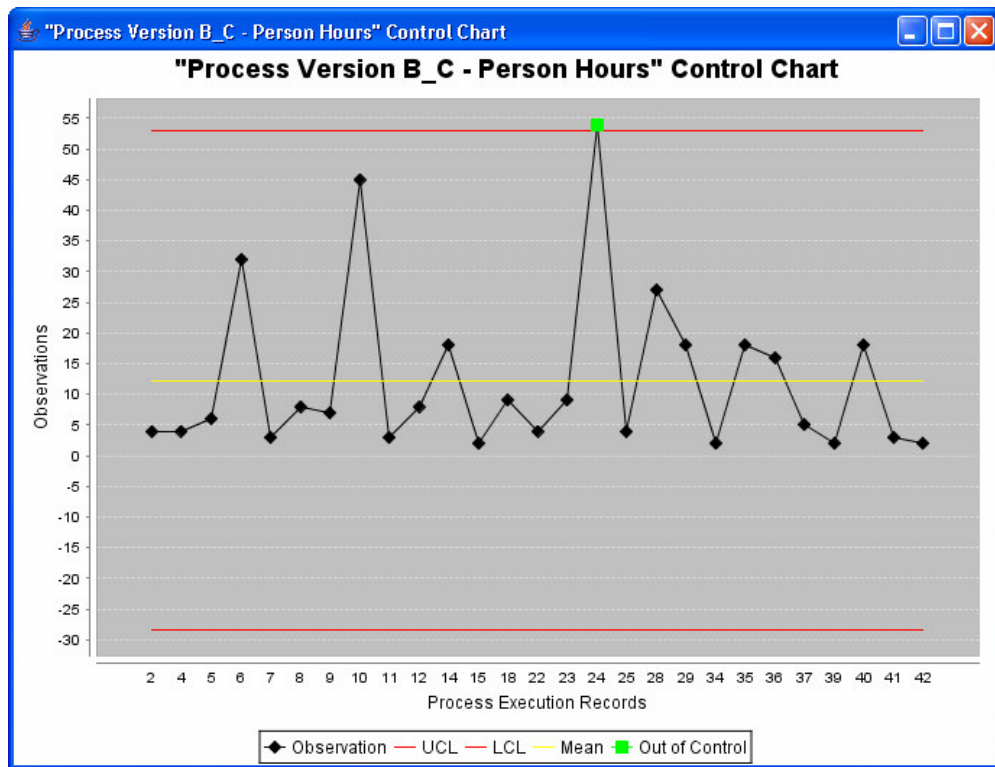


Figure 72 Control Chart drawn for “Version B_C – Person Hours” pair

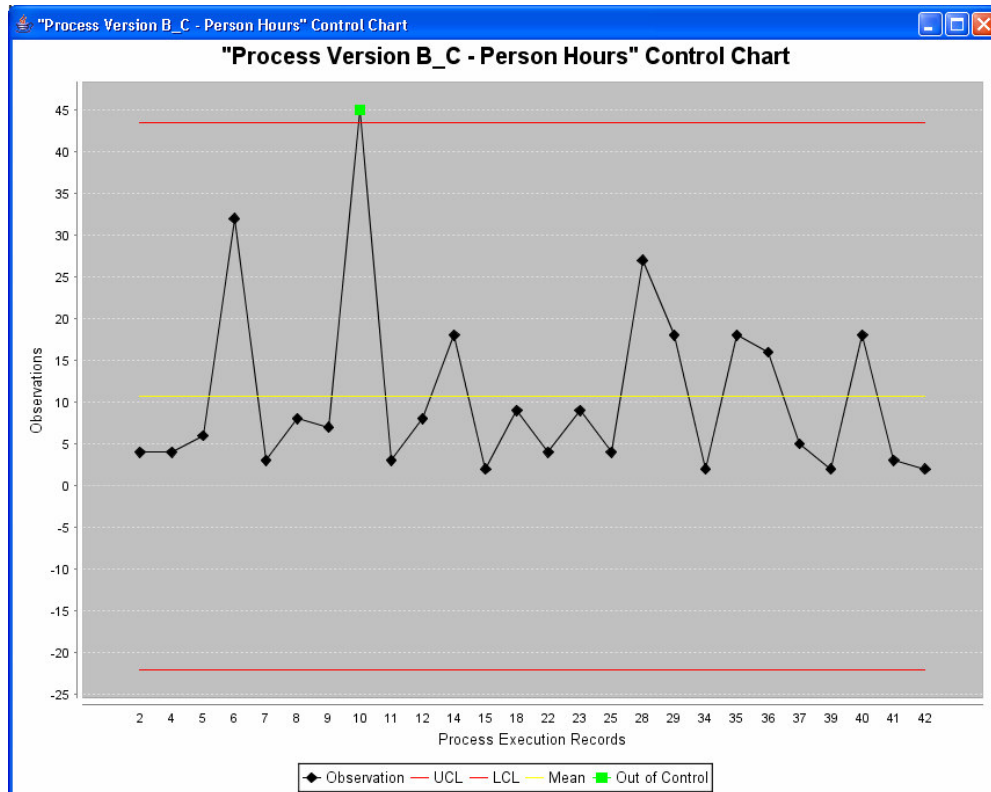


Figure 73 Control Chart drawn for “Version B_C – Person Hours” pair

When we excluded this OCP also and re-drew the control chart in Figure 74, we saw that process was under control. The mean was about 9 person-hours and Upper Control Limit (UCL) was about 35. According to the control chart results, we conclude that the process is not only under control but also capable since mean and UCL were consistent with the service level agreement of the company.

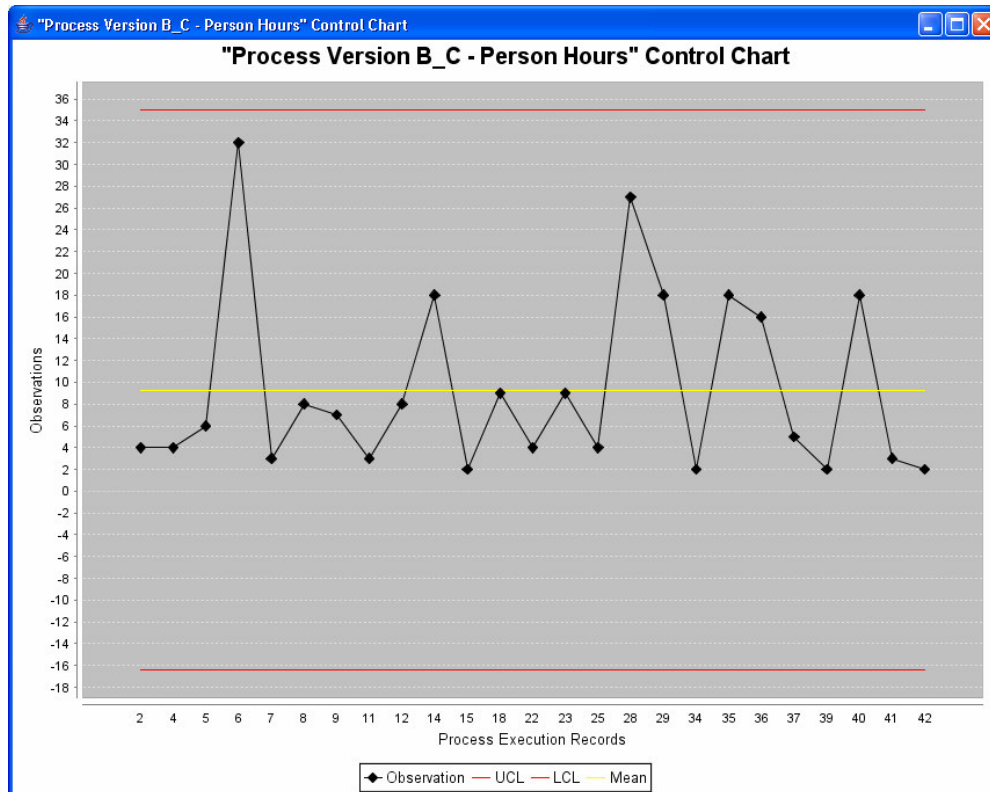


Figure 74 Final Control Chart drawn for “Version B_C – Person Hours” pair

As the second step, we merged the current three process clusters and drew control charts for the combined data. When we checked the control charts drawn for “Version D_E_B_C_A” process cluster, we observed one Out-of-Control Point (OCP) on both control charts and these were for the same process execution (PE no 6). When we checked this process execution, we found out that corresponding process instance is not for an error request but for a development request. Therefore we excluded this metric data point from analysis and re-drew the control charts. This time, we detected again one OCP for “Version D_E_B_C_A - Person Hours” pair while “Version D_E_B_C_A – Bug Aging” pair was under control. When we checked the detected OCP, we found out again that corresponding process instance is not for an error request but for a development

request. After excluding this OCP from the analysis, then process was also under control for “Version D_E_B_C_A - Person Hours” pair (see Figure 77).

When we have analyzed the final control chart for “Version D_E_B_C_A - Person Hours” pair, the mean observed was about 7 person-hours and Upper Control Limit (UCL) was about 22. When these results are compared to the results of Version B_C, it is observed that mean and UCL are smaller. This is an expected result because Version D_E_B_C_A includes the bugs which do not need change in the source code; therefore they could be resolved with less effort.

When we have analyzed the final control chart for “Version D_E_B_C_A – Bug Aging” pair, the mean was about 3.75 days and Upper Control Limit (UCL) was about 11 days. When these results are compared to the results of Version B_C, it is observed that mean and UCL are smaller again. This is an expected result because Version D_E_B_C_A includes the bugs which do not need change in the source code; therefore they could be resolved in less time.

A summary of analysis done with control charts can be found in the following tables.

Table 4 Initial Results from Charted Data in Context-1

Process	Metric	Cluster	Status
Bug Fixing	Bug Aging	Overall	1 OCPs
		Version B_C	Under Control
	Person Hours	Overall	4 OCPs
		Version B_C	2 OCPs

* OCP: Out-of-Control Point

Table 5 Assignable Causes for Out-of-Control Points in Context-1

Metric	Cluster	OCPs	Assignable Cause
Bug Aging	Version D_E_B_C_A	1	A development request was handled as a bug fix, so bug aging was high
	Version B_C	None	Not applicable
Person	Version D_E_B_C_A	4	3 of OCPs were due to development

Hours			requests handled as a bug fix. The other was due to missing knowledge of the Developer about the feature where bug was found and missing support from the team.
	Version B_C	2	One OCP was due to development request handled as a bug fix. The other was due to missing knowledge of the Developer about the feature where bug was found and missing support from the team.

* OCP: Out-of-Control Point

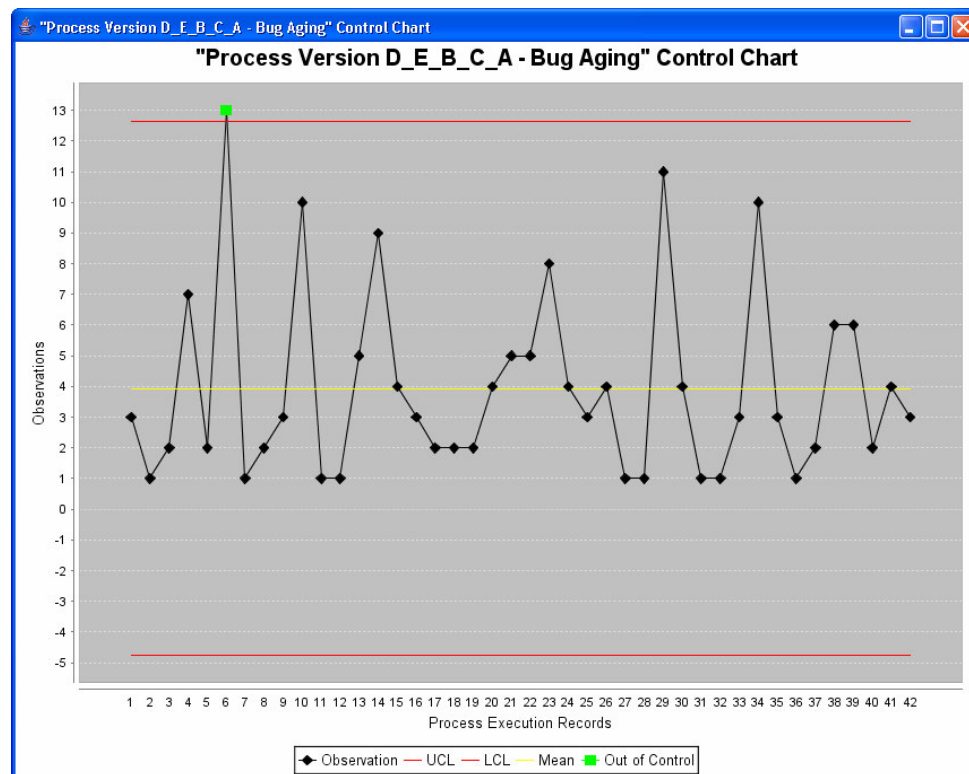


Figure 75 Control Chart for Combined Data of Bug Aging

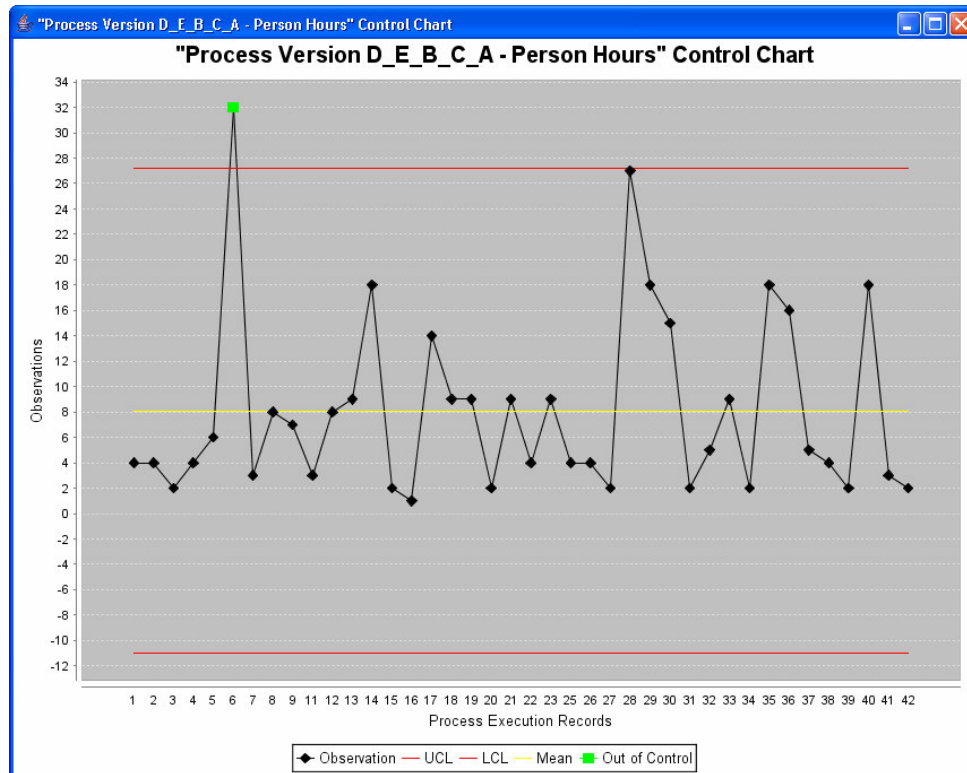


Figure 76 Control Chart for Combined Data of Person Hours

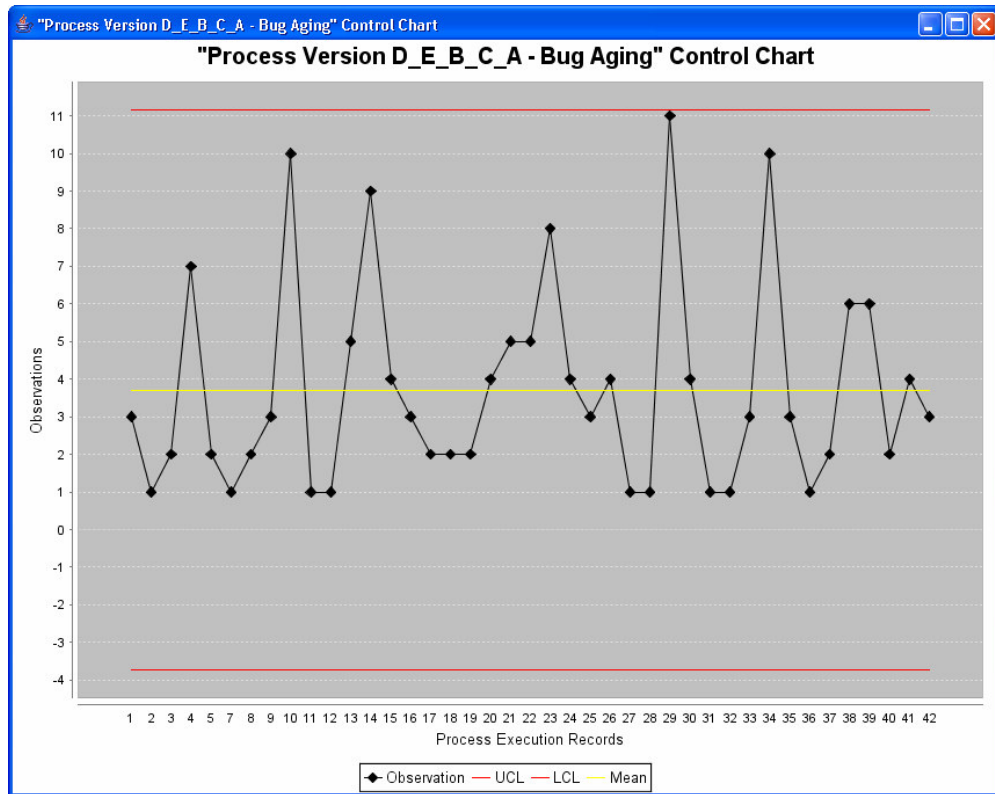


Figure 77 Final Control Chart for Combined Data of Bug Fixing

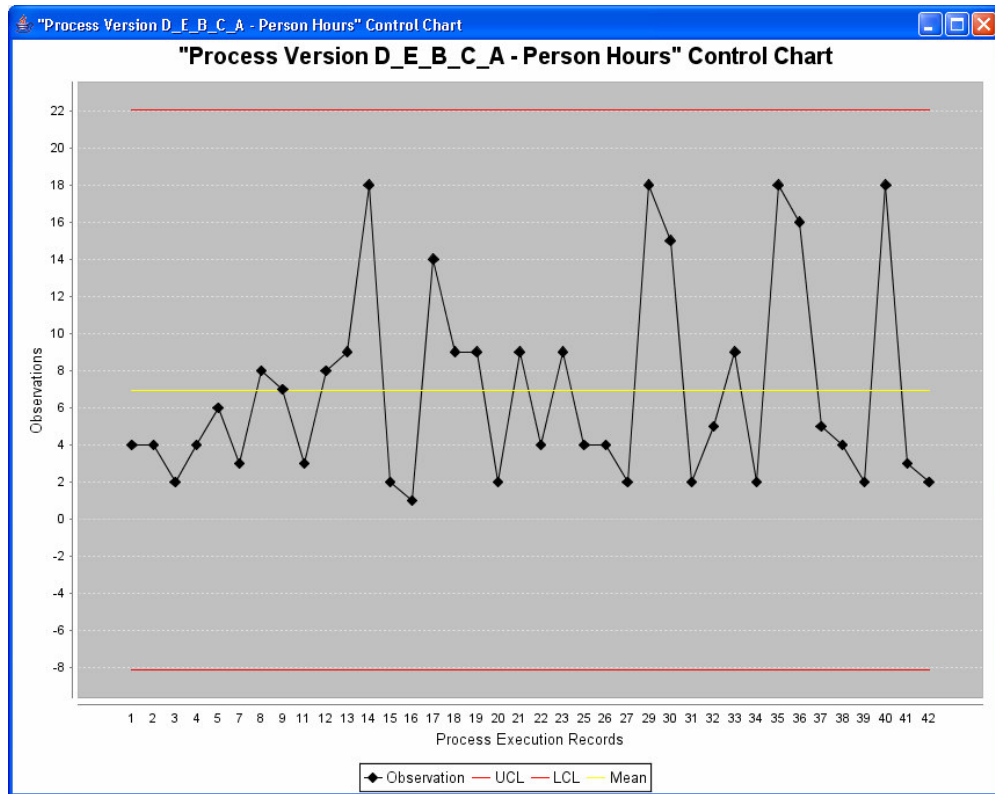


Figure 78 Final Control Chart for Combined Data of Person Hours

By using our tool, we have also drawn histograms for the metric data which are used at final control charts of Version B_C and Version D_E_B_C_A. We have used histograms for under control processes to visualize the frequency distribution of metric data. These can be seen below:

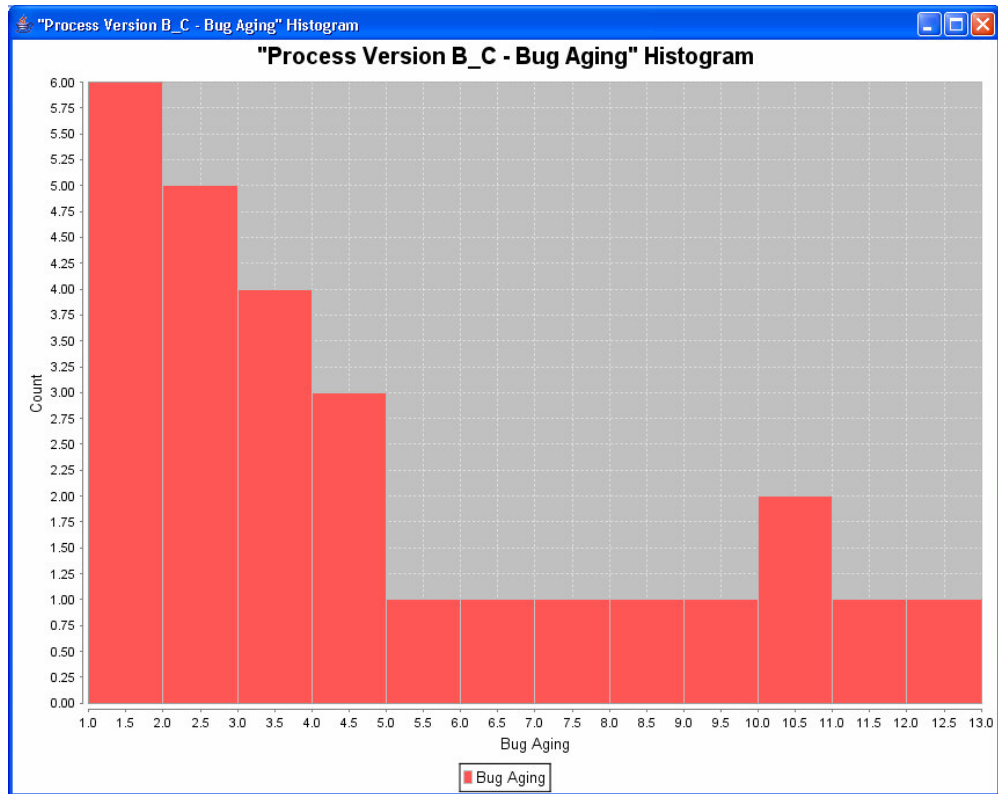


Figure 79 Process Version B_C – Bug Aging Histogram

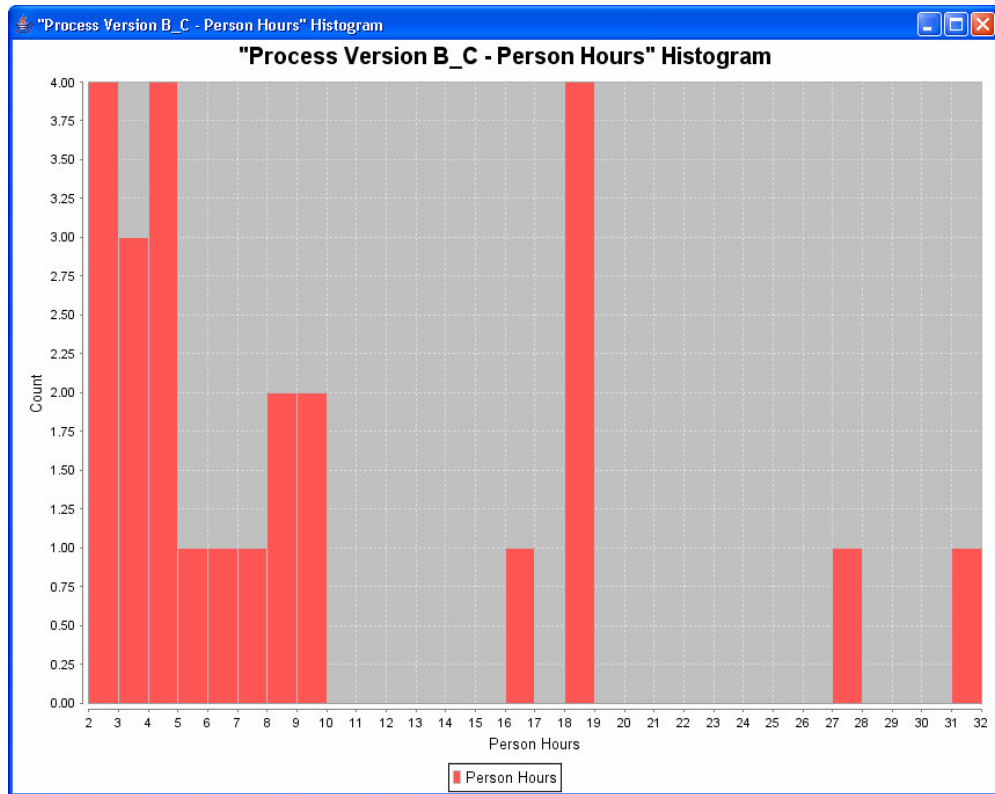


Figure 80 Process Version B_C – Person Hours Histogram

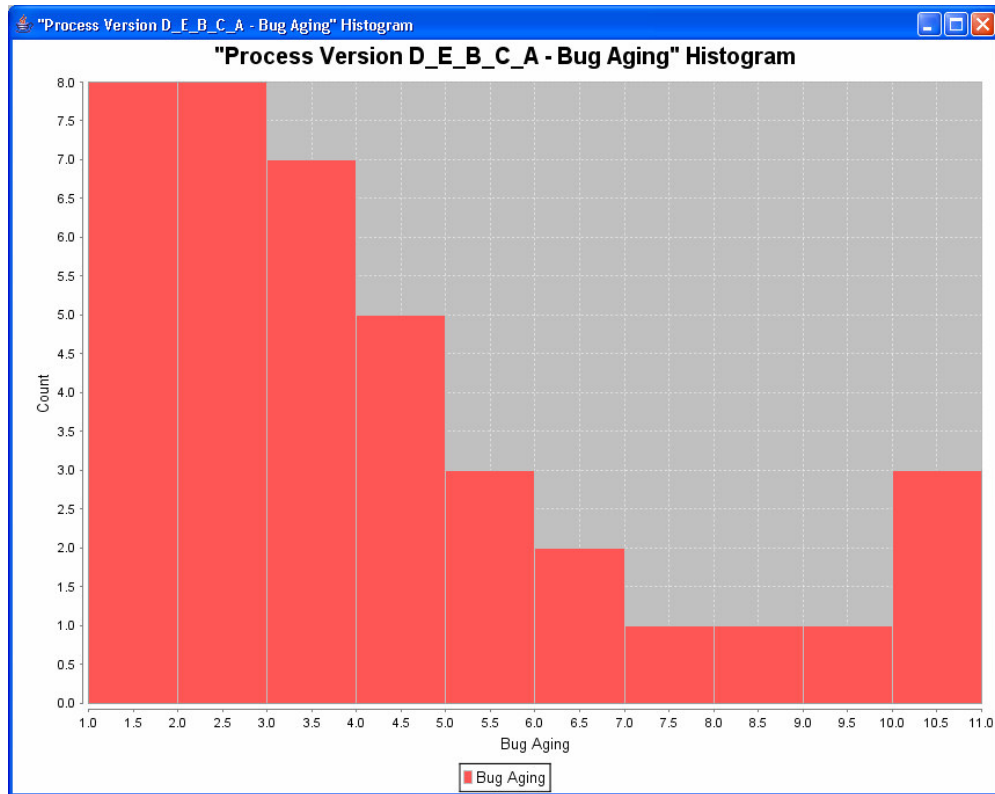


Figure 81 Process Version D_E_B_C_A – Bug Aging Histogram

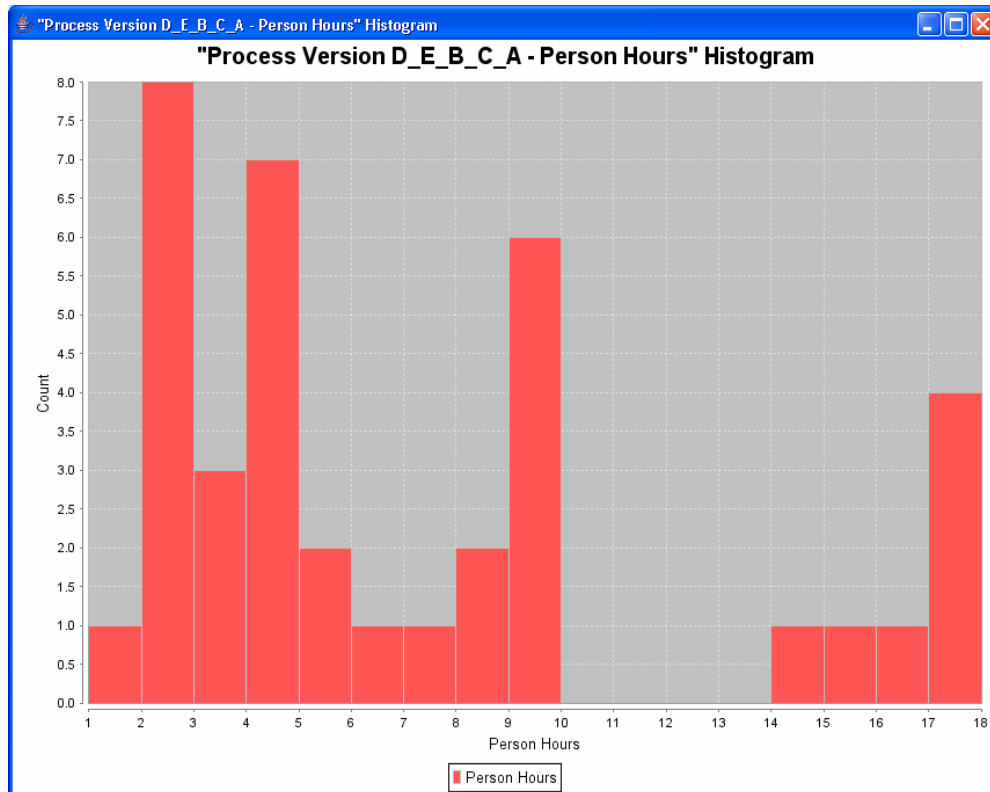


Figure 82 Process Version D_E_B_C_A – Person Hours Histogram

Until now we have shown usage of SPC tools for metrics type of ratio or absolute with our tool, but we have also supported analysis of nominal and ordinal metric types. We have drawn bar chart for status of bugs, shown in the Figure below. This bar chart revealed another problem about the project. Normally, when a bug is resolved it is status is changed to RES. After that tests are performed to check whether the problem reported is really resolved and status is changed to FIN. When we checked the bar chart, there were lots of bugs which were in state RES. In other words, we had lots of bugs which are resolved but not tested. This was also another point for improvement of the project.

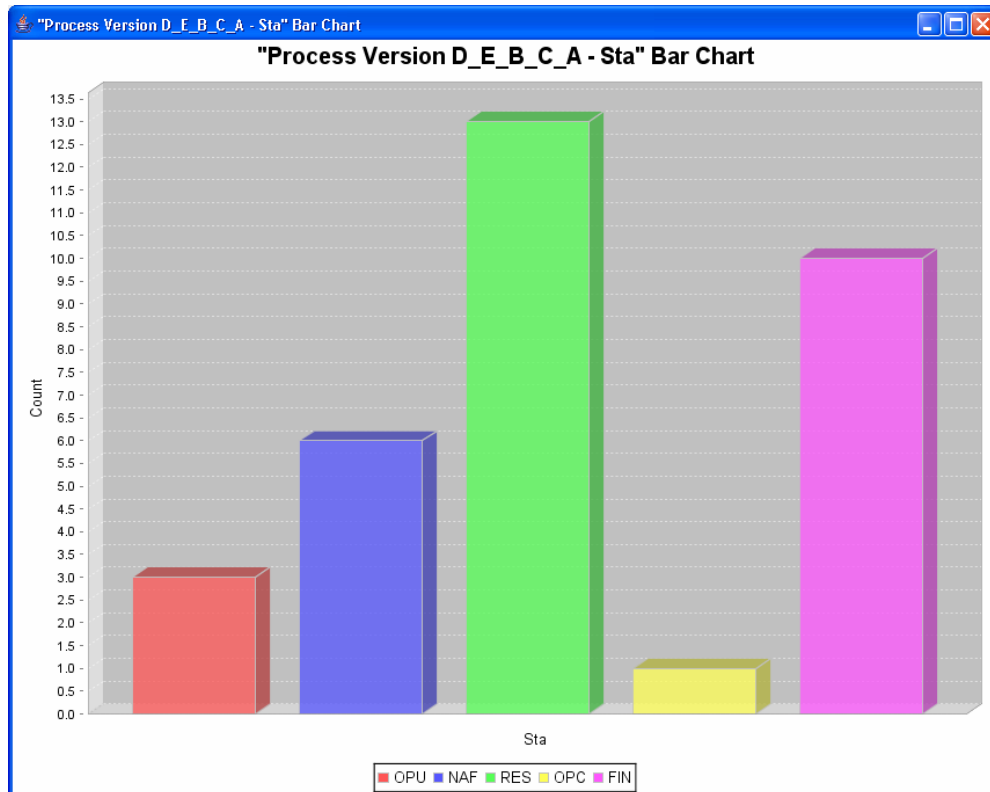


Figure 83 Bar Chart for Status

Findings from the study:

During the implementation of the case study, we have detected improvements about the process analyzed, bugs and improvements for our tool and improvements about the SPC assessment model we have used (SPC-AM) besides gaining inside about the analyzed process. These will be described in detail below:

Inside about the process: We have learned mean values and control limits of process metric data. This information is very precious especially for planning and tracking. Mean values and control limits of process metrics for each cluster are given in the table below.

Table 6 Results of Case Study A

Process	Metric	Cluster	Mean	UCL
Bug Fixing	Bug Aging	Overall	3.707	11.154
		Version B_C	4.407	15.864
	Person Hours	Overall	6.947	22.042
		Version B_C	9.28	34.989

* UCL: Upper Control Limit

Improvements about the process: Two possible improvement points have been detected for the analyzed project:

- Communication among Developers can be improved. Especially when a Developer has problems about identifying the solution for a bug. These problems can be discussed in more detail in the project meetings.
- Problems about testing of resolved bugs are detected. Most of the resolved bugs are not tested. For each bug resolved, a tester can be assigned and testing can be tracked.

Improvements for our tool: The following improvements are detected during the study:

- Reading process definition from a file or providing a GUI to define process visually and generating process attributes automatically from this
- Adding functionality for copying process attributes of a Process Execution Record to another Process Execution Record.
- Disabling showing input dialogs when a Process Attribute is checked on Process Similarity Matrix (PSM).
- Changing table cells of Process Attributes (PAs) to Combo boxes so that existing PAs can be chosen
- Adding Paste menu item to table cells

- At PSM, working with the row or column where mouse is on instead of the selected one
- At PSM, showing Process Attributes always at left side to ease identifying the correct rows for PAs

Bugs for our tool: Six minor bugs had been detected during the study and they were corrected later.

Improvements for SPC-AM: The following improvements are detected during the study:

- Questions in “Process Performers” part of Process Execution Questionnaire should be asked in negative form to be consistent with other questions.
- Questions in “Metric Definition” part of Metric Usability Questionnaire should be enhanced to consider the metrics in Date type.
- Two questions in “Data Existence” part of Metric Usability Questionnaire can be changed to increase the understandability. Instead of using “What is the amount of...” we can use “How many process instances are there...”

4.3. Context-2 (Case Study B)

Within the second case, we worked on recruitment process of a software department of a company. We decided to analyze the recruitment process at the initial meetings with the Managers. The aim was to have more inside about the process and detect the possible problems. Recruitment process is analyzed for the last 3 months period, from October 2006 to December 2006. The company had a separate Human Resources (HR) department which supports Managers from other departments during recruitment process.

Recruitment process starts when there is a free position in one of the existing projects or a new project which is not started yet. This request usually comes from a Requester who works for a partner department since most of the projects are

performed for some other departments in the company. First of all, Manager evaluates existing available resources to fill the requested position. If there are some available qualified resources then these are selected for further evaluation. If there are no available resources, Manager fills an employee request form (Eleman Talep Formu, ETF) and sends to HR department. Then HR creates an advertisement on the web site (Kariyer.net) to announce the free position. When there is enough number of applications, HR specialist performs the first selection and grouping on the web site. After this first selection, Manager performs the second selection and grouping on the web site. Then HR specialist arranges meetings with the selected applicants. These meetings are face to face meetings where both Manager and HR specialist attend. After the meetings, if both Manager and HR specialist agree on some applicants, these applicants are evaluated by the Requester. Then final decision is given and job is offered to the agreed applicants. When one applicant accepts the offer, formal tasks are performed for the recruitment. The whole process is defined as an eEPC diagram in Figure 84.

The analysis was performed together with the related Managers of the organization. We spent 1 hour 20 minutes for collecting data and 3.5 hours for applying the approach, performing the analyses, and interpreting the results. In other words, we spent about 5 hours for whole analysis. The set of assets produced during this case study with the control charts are provided in Appendix D.

The study was retrospective, and instead of identifying process attribute values to put on process similarity matrices by filling process execution records, we preferred drawing general process flows with one Manager. We depicted the executions as draft on a paper first together with the Manager, and then converted the flows into eEPC (Extended Event Driven Process Change) diagram by using MS Visio. The flow for recruitment process is given in Figure 84.

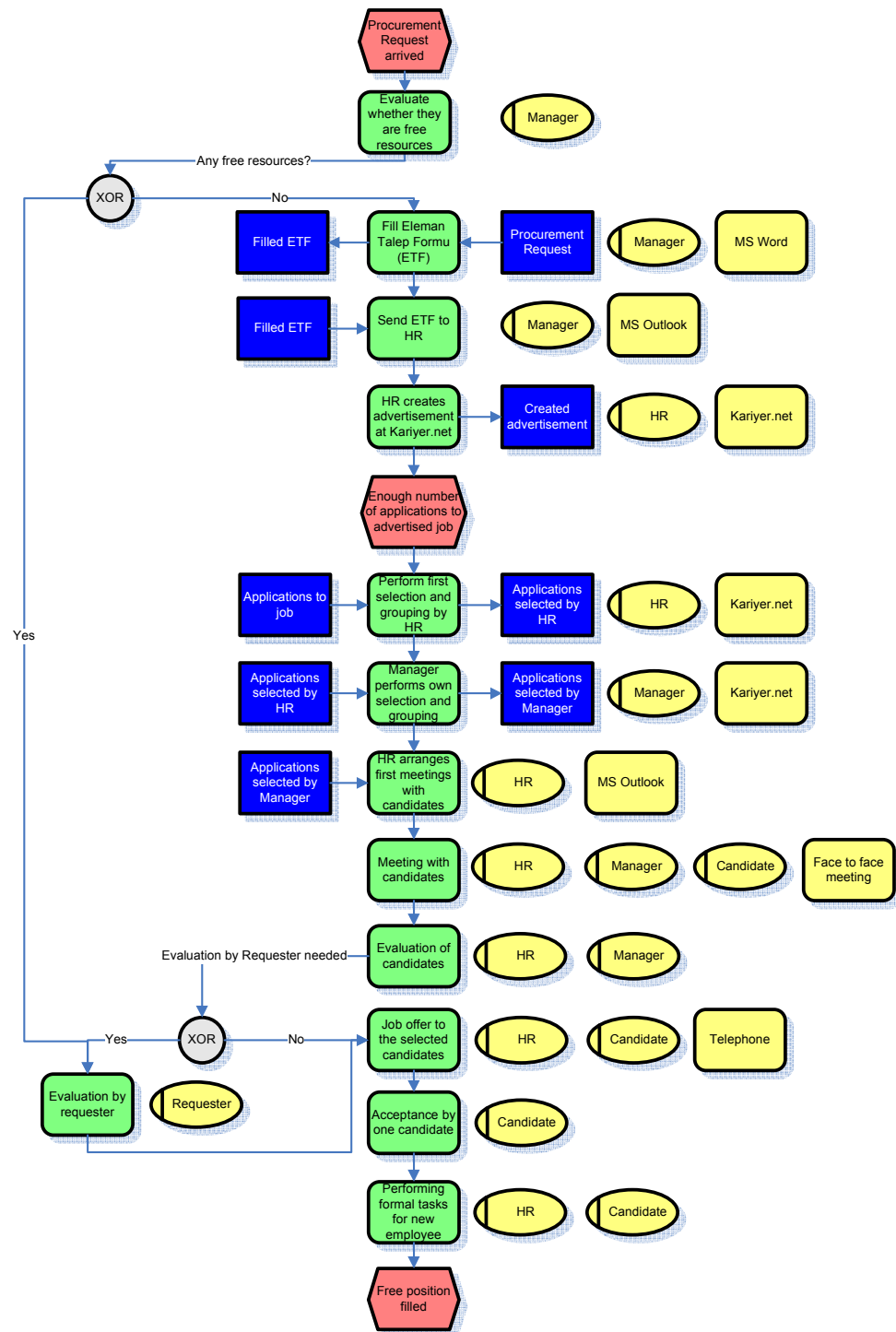


Figure 84 eEPC for Recruitment Process (Case B)

Managers were planning and tracking most of the recruitment information on one Excel sheet including procurement ID, position, project name, due date, onsite date, start date. But not all information is up-to-date and complete. We have also realized that go date (the date recruitment request arrived) was missing. Therefore, we checked the e-mails sent to the Managers to fill the missing information about recruitments. We were successful in finding out missing information and the file was complete. First of all, we imported this metric data to our tool. Therefore, one process execution record is automatically created for each recruitment process instance in the Excel file imported. There were 25 data points. Then by using the process elements (inputs, outputs, activities, roles, and tools) on the eEPC diagram, we have added the values of process attributes for inputs, outputs, activities, roles, and tools & techniques on the process similarity matrix. Therefore we could add Process Attributes on the similarity matrix and checked against process executions. The process similarity matrix for recruitment process executions is provided in Figure 85.

We had 25 process execution records totally and we completed process similarity matrices for Inputs, Outputs, Activities, Roles, and Tools & Techniques of all process execution records.

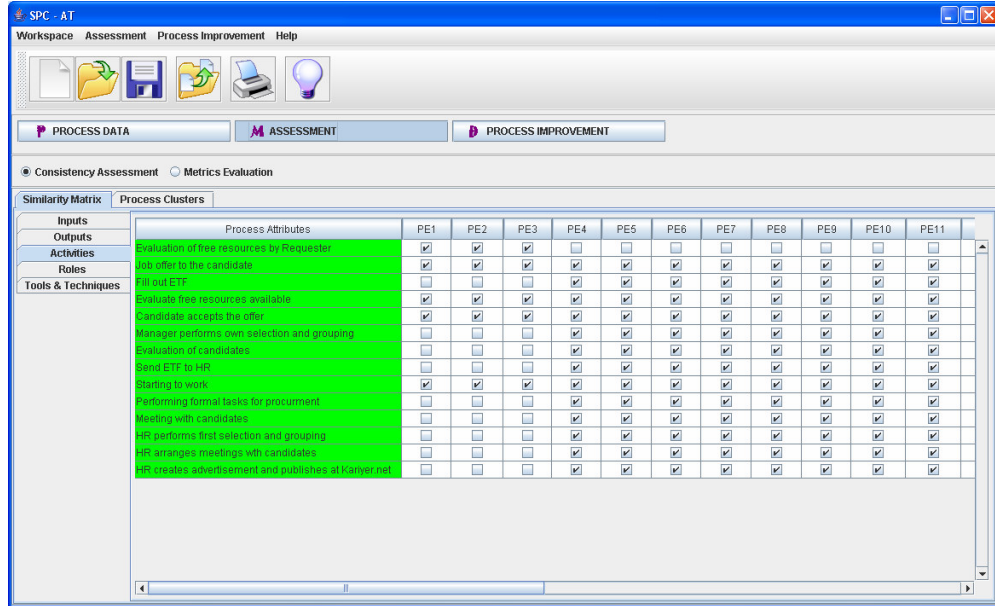


Figure 85 Process Similarity Matrix for Recruitment (Case B)

After finalizing the process similarity matrices, we checked under “Process Clusters” tab-sheet to see the automatically identified process clusters by our tool. Our tool identified 2 process clusters labeled as “Version A” and “Version B” as shown in Figure 86, by observing the similarities between process executions. The number of data points was enough (at least 20) for Version B, but not for Version A. We realized that process cluster Version B represents the process where new employee is found for the open position in the project while Version A represents existing employee is moved to the open position.

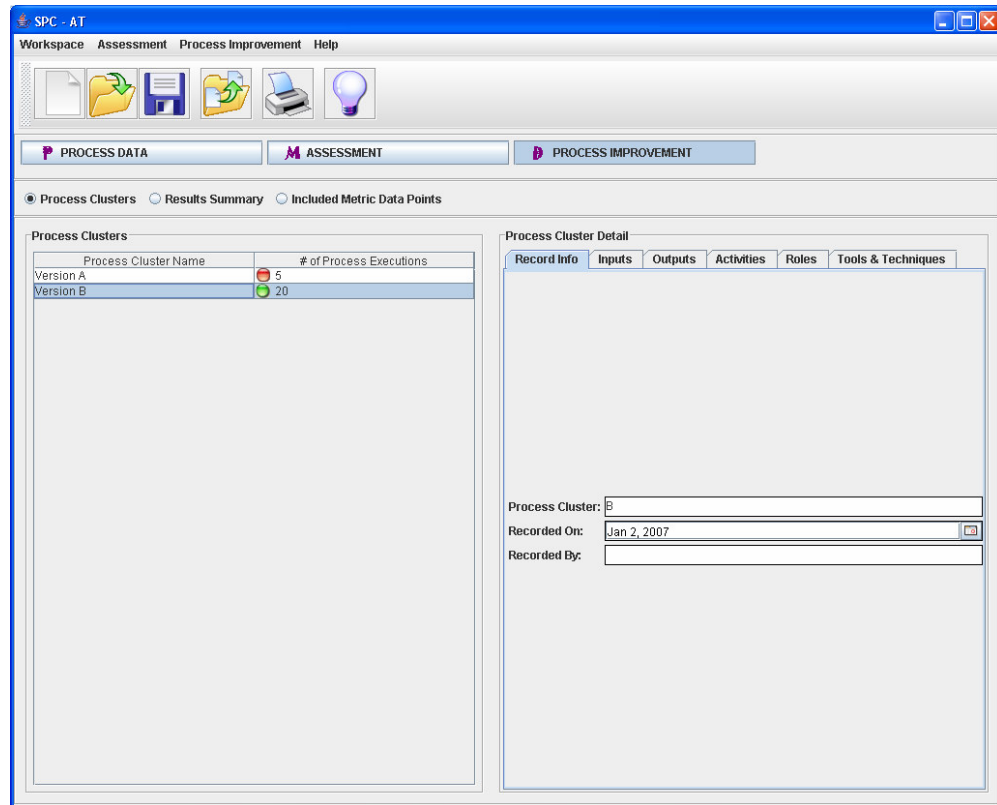


Figure 86 Base Process Clusters for Bug Fixing Process

After we identified initial process clusters, we have changed our view to Metric Evaluation (under ASSESSMENT view) to assess the usability of the process metrics. The metrics which are imported to our tool from Excel at the beginning of the case study are shown in the table below as Base metrics. We decided to derive three new metrics with the imported base metrics: Actual Procurement Time, Planned Procurement Time, and Procurement Time Variance. Explanation and formulas of derived metrics are shown in the table below. Furthermore, relationships between the base metrics and derived metrics are shown visually in Figure 87. The metrics at upper side represent the base metrics.

Table 7 Process Metrics (Original and Derived) for Case B

Metric Name	Metric Type	Explanation
Go Date	Base	Start of Recruitment process
Due date	Base	Planned Start Date for new project member
Start Date	Base	Start Date for joining of new project member
Planned Procurement Time	Derived	$Due\ date - Go\ Date + 1$ Planned time spent for finding the right person for the position
Actual Procurement Time	Derived	$Start\ Date - Go\ Date + 1$ Realized time spent for finding the right person for the position
Procurement Time Variance	Derived	$Actual\ Procurement\ Time - Planned\ Procurement\ Time$ The time difference between realized and planned procurement time
Position	Base	The position for the new project member

We created one entry under Derived Metrics tab-sheet for each new derived metric. After filling Metric Formula attribute for the derived metrics, metric data was calculated automatically from the entered formula and stored by our tool.

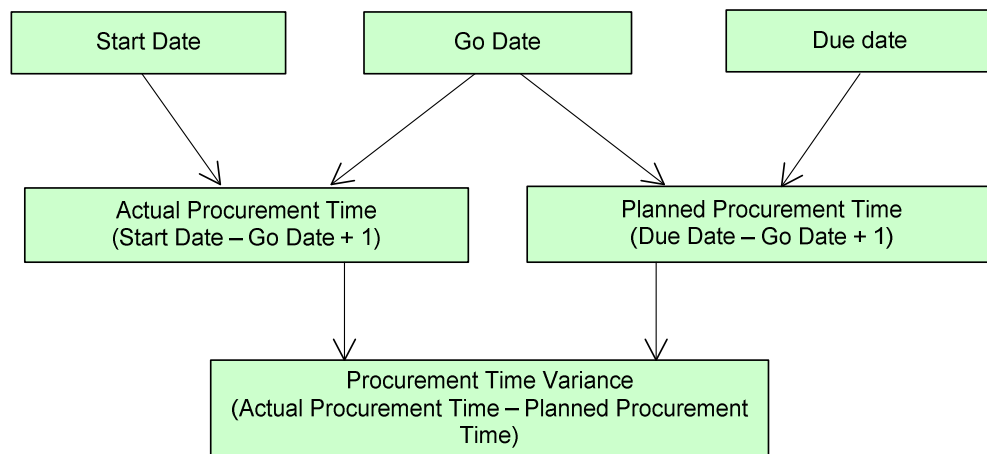


Figure 87 Derived Metrics Identified in Context-2

After creating the derived metrics, we filled Metric Usability Questionnaire for each base and derived metric from Questionnaire tab-sheet under Metric

Evaluation view. Example questionnaire for “Procurement Time Variance” derived metric and usability ratings given are shown in Figure 89 (completed questionnaires for all metrics identified in Context-2 are provided in appendix D). The usability status of all base and derived metrics are listed in Figure 67. In the next step, only metrics which are evaluated as “usable” would be used for control charting.

The screenshot shows the SPC - AT software interface. The main window is titled 'SPC - AT' and has a menu bar with 'Workspace', 'Assessment', 'Process Improvement', and 'Help'. Below the menu bar is a toolbar with icons for file operations and a lightbulb. The 'Assessment' tab is selected, and the 'Metrics Evaluation' sub-tab is active. The 'Process Metrics' list on the left includes 'Metric Name', 'Planned Procure...', 'Actual Procureme...', and 'Procurement Tim...'. The 'Metric Usability Assessment Detail' window is open, showing the 'Questionnaire' tab. It contains a table with 5 questions and their answers. The 'Usability Rating' is set to 'F' (Failed). A 'Save Metric Usability Assessment' button is at the bottom right.

Q.	Question	Answer
1	What is the metric formula? (please refer to related base metr...	Actual Procurement Time-Planned Procureme
2	What is the scale of the metric data? (nominal, ordinal, interva...	Ratio
3	What is the unit of the metric data?	days
4	What is the type of the metric data? (integer, real, etc.)	integer
5	What is the range of the metric data?	-inf-inf

Usability Rating: F

Save Metric Usability Assessment

Figure 88 Metric Usability Questionnaire for “Procurement Time Variance”
Derived Metric of “Procurement” Process

SPC - AT

Workspace Assessment Process Improvement Help

PROCESS DATA ASSESSMENT PROCESS IMPROVEMENT

Consistency Assessment Metrics Evaluation

Base Metrics Derived Metrics

Process Metrics

Metric Name	M
Planned Procur...	●
Actual Procure...	●
Procurement TI...	●

Metric Usability Assessment Detail

General Info Questionnaire Usability Rating

Metric Usability Attributes	Rating	Expected Rating
Metric Identity	● F	F
Data Existence	● F	F
Data Verifiability	● F	L
Data Dependability	● L	L
MUF-3&4 for Planned Procurement Time	● L	L
MUF-3&4 for Actual Procurement Time	● L	L

Save Metric Usability Assessment

Figure 89 Metric Usability Ratings for “Procurement Time Variance” Derived Metric of “Procurement” Process

Metric Usability Evaluation Report - Print Preview

File Navigation Zoom Help

100 %

Metric Name	Type	Metric Usability
No	Base	NOT Usable
Position	Base	Usable
Go Date	Base	Usable
Due date	Base	Usable
Start Date	Base	Usable
Planned Procureme..	Derived	Usable
Actual Procuremen..	Derived	Usable
Procurement Time ..	Derived	Usable

Page 1 of 1

Figure 90 Metric Usability Report for Procurement process

After identifying the base process clusters and assessing the usability of the metrics, we have changed our view to Process Clusters (under PROCESS IMPROVEMENT view) to finalize the process clusters. But we realized that the order of process executions should be changed to be able to apply control charting. Therefore, we have decided to add a new use case for our tool: Sorting Process Execution Records. After the implementation of the use case we continued to analyze. Process execution records were sorted according to “Due Date” metric values. The number of data points was enough (20) just for Version B, and we decided to include data points from Version A by merging both process clusters. Therefore we decided to apply control charting on Version B process cluster and also the one which is produced by merging these two process clusters.

Process Cluster Name	# of Process Executions
Version A_B	25

Figure 91 Process Clusters after the merge

SPC tools could only be applied to the qualified process cluster – metric pairs. In this case, these were “Version B – Actual Procurement Time”, “Version B – Procurement Time Variance” and “Version B – Planned Procurement Time” for control charting. We ignored “Version B – Planned Procurement Time” because Managers had not been interested in the results. Control charts drawn for these two process cluster – metric pairs are shown in Figure 92 and Figure 94.

When we checked the control chart drawn for “Version B – Procurement Time Variance” pair, we observed that no OCP was detected and process is under control. The mean was about 11 days, Lower Control Limit (LCL) was about -56 and Upper Control Limit (UCL) was about 78. According to the control chart results, we conclude that the process is under control. But we concluded that the process is not capable. Managers expected that the mean is near to 0 and the upper limit is less. After this analysis, process improvement was initiated to enhance the recruitment process.

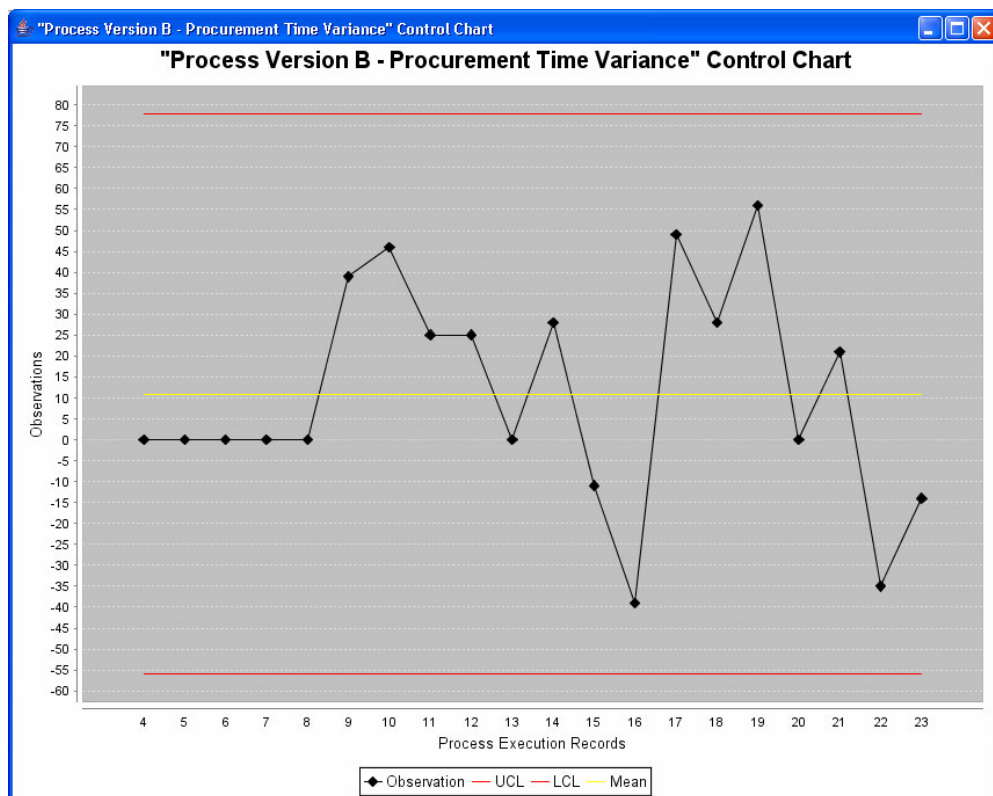


Figure 92 Control Chart for Version B – Procurement Time Variance pair

When we checked the control chart drawn for “Version B – Actual Procurement Time” pair, we observed two Out-of-Control Points (OCPs). When we checked these OCPs detected, we observed that for 5 positions same Due Date had been

given and all new employees had started to work at the same time. Therefore, we decided not to count such situations as OCPs and changed the related configuration accordingly (see Figure 93). After changing the configuration for OCP Rules we re-drew the control chart in Figure 95, we saw that process was under control. The mean was about 70 days, Lower Control Limit (LCL) was about 11 and Upper Control Limit (UCL) was about 130. According to the control chart results, we conclude that the process is under control. But we concluded that the process is not capable. 70 days as the average to procure one person and 130 days as maximum value were found so high by the Managers. We decided to initiate a study to reduce average recruitment time and the natural upper limit.

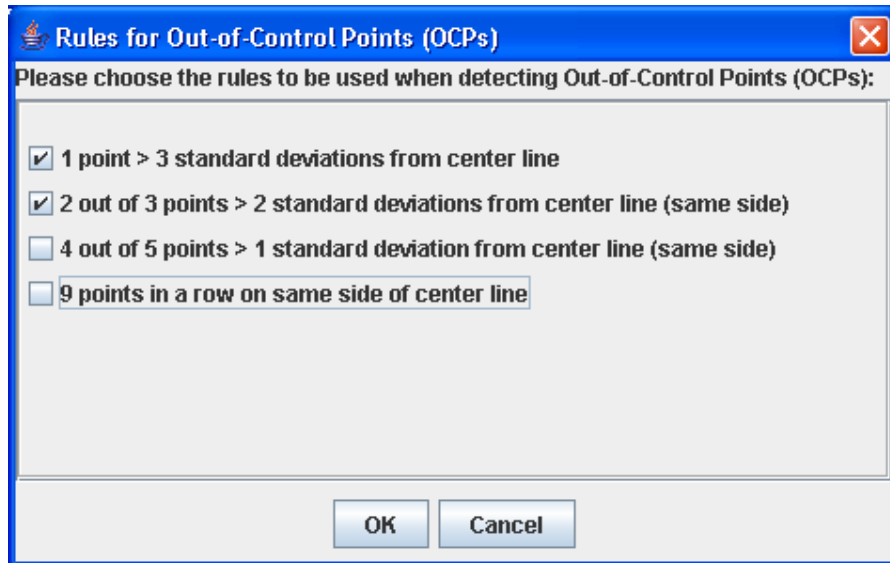


Figure 93 Rules for OCPs (Case B)

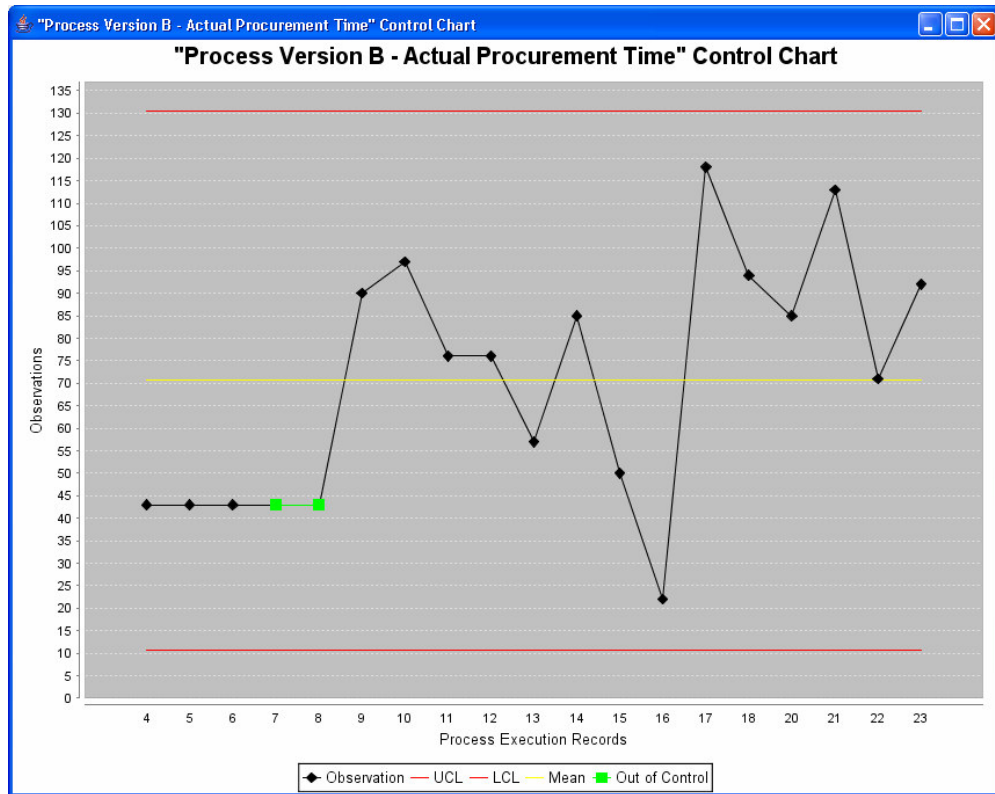


Figure 94 Control Chart for Version B – Actual Procurement Time pair

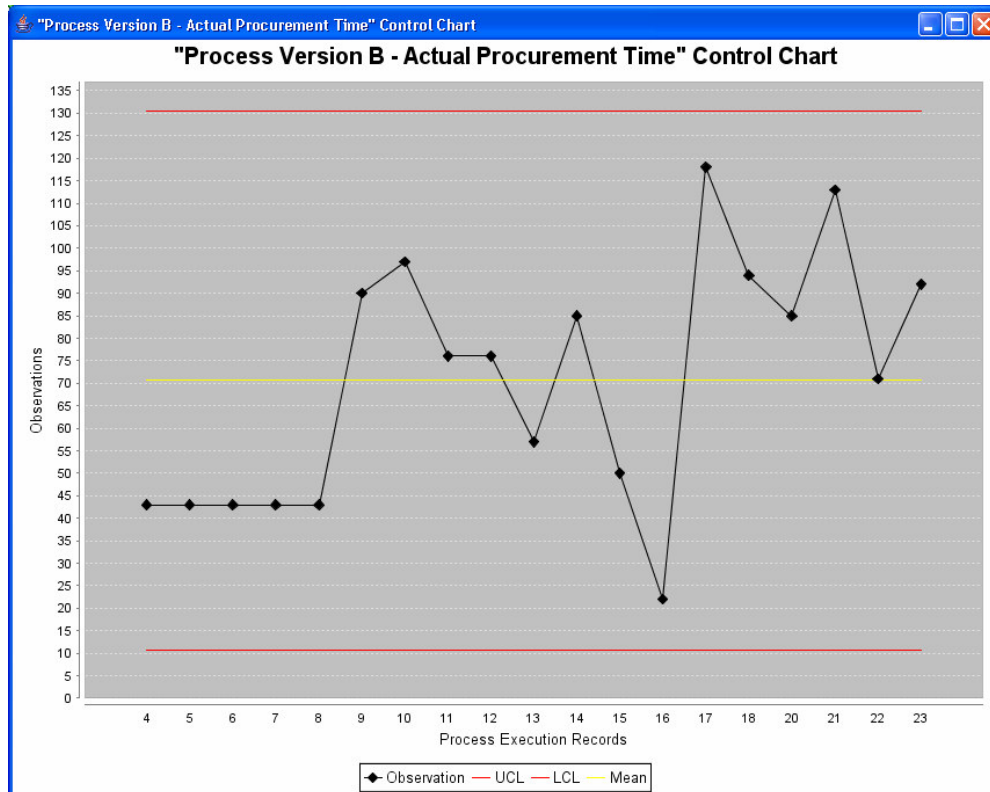


Figure 95 Control Chart drawn for “Version B – Actual Procurement Time” pair

As the second step, we merged the current two process clusters and drew control charts for the combined data. When we checked the control charts drawn for “Version A_B” process cluster (see Figure 97), we observed one Out-of-Control Point (OCP) for “Version A_B – Procurement Time Variance” pair and three OCPs for “Version A_B – Actual Procurement Time” pair. Process execution with the number 19 was detected as OCP on both control charts. When we checked this process execution, we found out that this new employee was currently working at abroad and the procedure to leave the company takes more than two months. The Manager who wanted to procure this candidate had decided not to miss this candidate and to wait for him since he had been very appropriate for the position requested. This was really an extraordinary situation and therefore we excluded this metric data point from analysis and re-drew both control charts.

This time, we detected one OCP for “Version A_B – Actual Procurement Time” pair while “Version A_B – Procurement Time Variance” pair was under control. When we checked the detected OCP (Process execution with no 1), we found out that an existing employee had just left his/her project and a request for a new team member of a project had been arrived. The Manager had been lucky since this employee had had enough skills for the new position and he/she had immediately been moved to the new project. This was also an extraordinary situation and therefore we excluded this metric data point from analysis. After excluding this OCP from the analysis, then process was also under control for “Version A_B – Actual Procurement Time” pair (see Figure 99).

When we have analyzed the final control chart for “Version A_B – Procurement Time Variance” pair, the mean observed was about 6 days, Lower Control Limit (LCL) was about -47 and Upper Control Limit (UCL) was about 59. When these results are compared to the results of Version B, it is observed that mean and UCL are smaller. This is an expected result because Version A_B includes recruitment instances with moving existing employees of the department to the requested position; therefore recruitment process takes less time. We have also observed that distance between UCL and LCL is less for Version A_B. This is also related with excluding one extreme metric data point from the analysis. This has reduced the expected range for metric data values. In spite of the enhancements of mean and control limits, we were again not satisfied with the capability of the process as it had been for Version B.

When we have analyzed the final control chart for “Version A_B – Actual Procurement Time” pair, the mean was about 69 days, Lower Control Limit (LCL) was about 18 days and Upper Control Limit (UCL) was about 120 days. When these results are compared to the results of Version B, it is observed that mean and UCL are smaller again. This is an expected result because Version A_B includes recruitment instances with moving existing employees of the department to the requested position. The change for mean is not much as UCL since we have also excluded one smallest metric data point besides one highest. We have also

observed that distance between UCL and LCL is less for Version A_B. This is also related with excluding two extreme metric data points from the analysis. This has reduced the expected range for metric data values. We were also not satisfied with the capability of the process as it had been for Version B.

A summary of analysis done with control charts can be found in the following tables.

Table 8 Initial Results from Charted Data in Context-2

Process	Metric	Cluster	Status
Recruitment	Actual Procurement Time	Overall	2 OCPs
		Version B	Under Control
	Procurement Time Variance	Overall	1 OCP
		Version B	Under Control

* OCP: Out-of-Control Point

Table 9 Assignable Causes for Out-of-Control Points in Context-2

Metric	Cluster	OCPs	Assignable Cause
Actual Procurement Time	Version B_A	2	One OCP was due to recruitment of a new employee working at abroad. The other was due to moving of an existing employee internally to the requested position.
	Version B	None	Not applicable
Procurement Time Variance	Version B_A	1	The OCP was due to Recruitment of a new employee working at abroad.
	Version B	None	Not applicable

* OCP: Out-of-Control Point

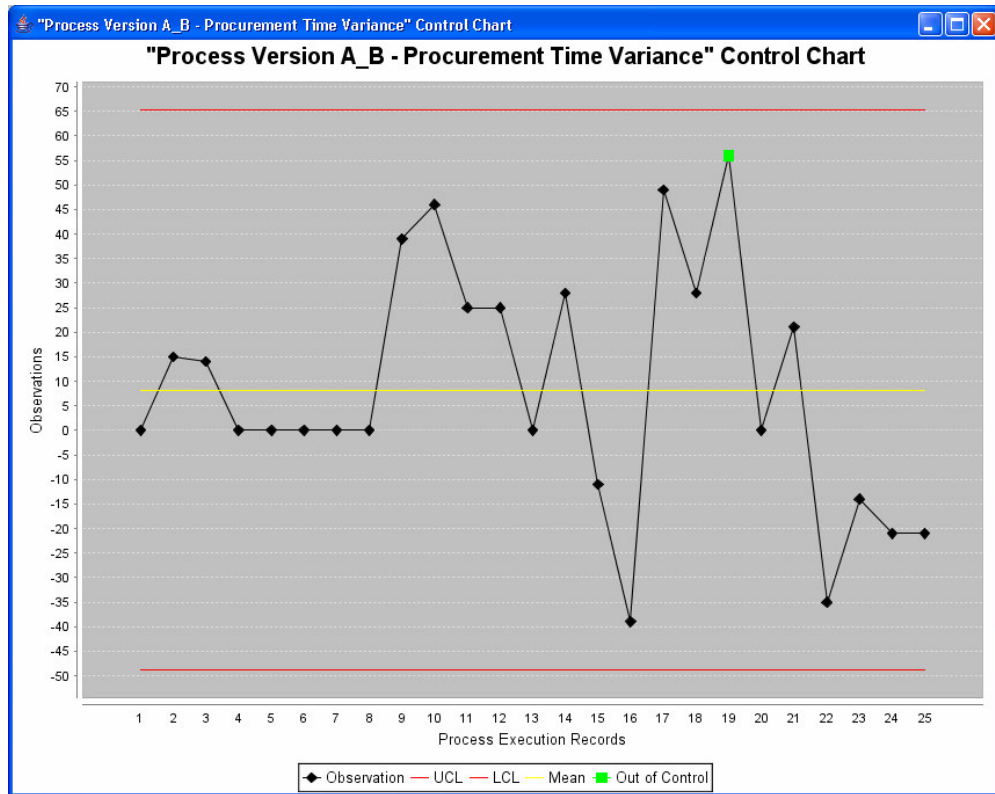


Figure 96 Control Chart for Version A_B – Procurement Time Variance pair

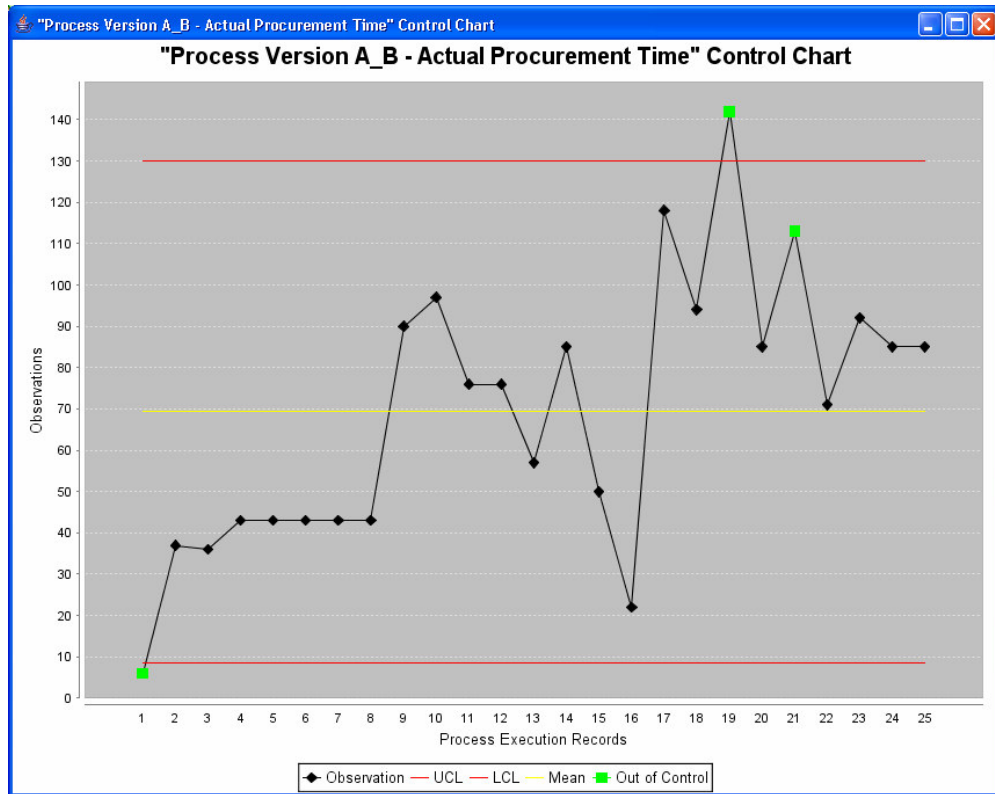


Figure 97 Control Chart for Combined Data of Recruitment

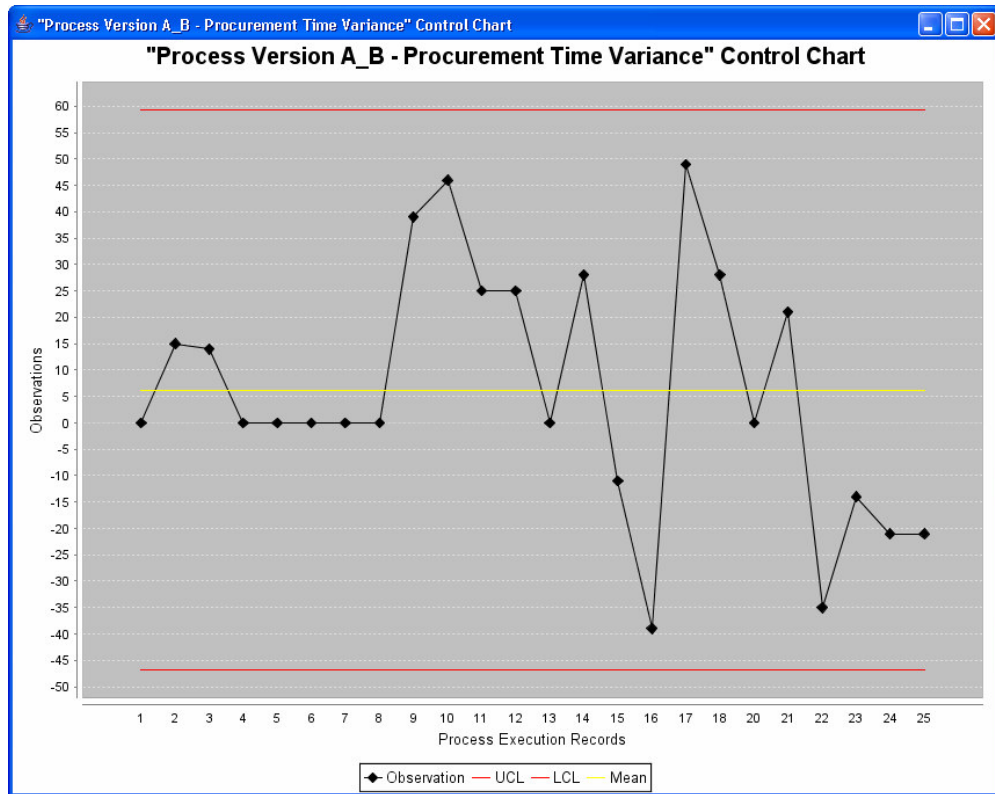


Figure 98 Final Control Chart for Combined Data of Recruitment

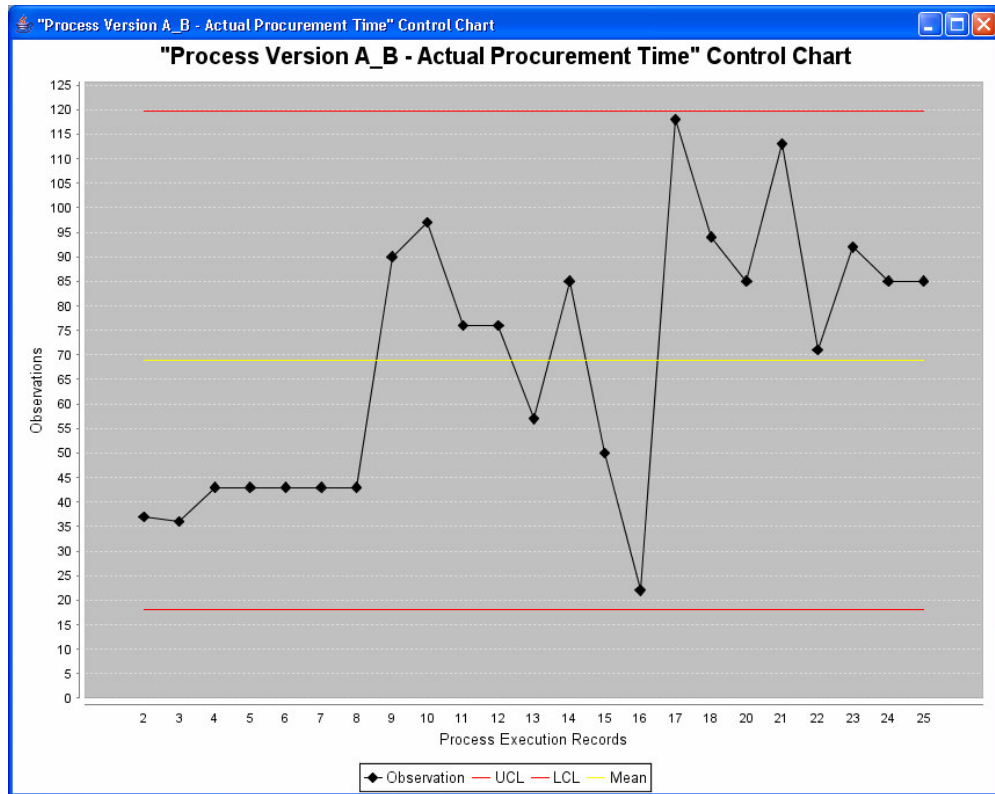


Figure 99 Final Control Chart for Combined Data of Recruitment

By using our tool, we have also drawn histograms for the metric data which are used at final control charts of Version B and Version A_B. We have used histograms for under control processes to visualize the frequency distribution of metric data. These can be seen below:

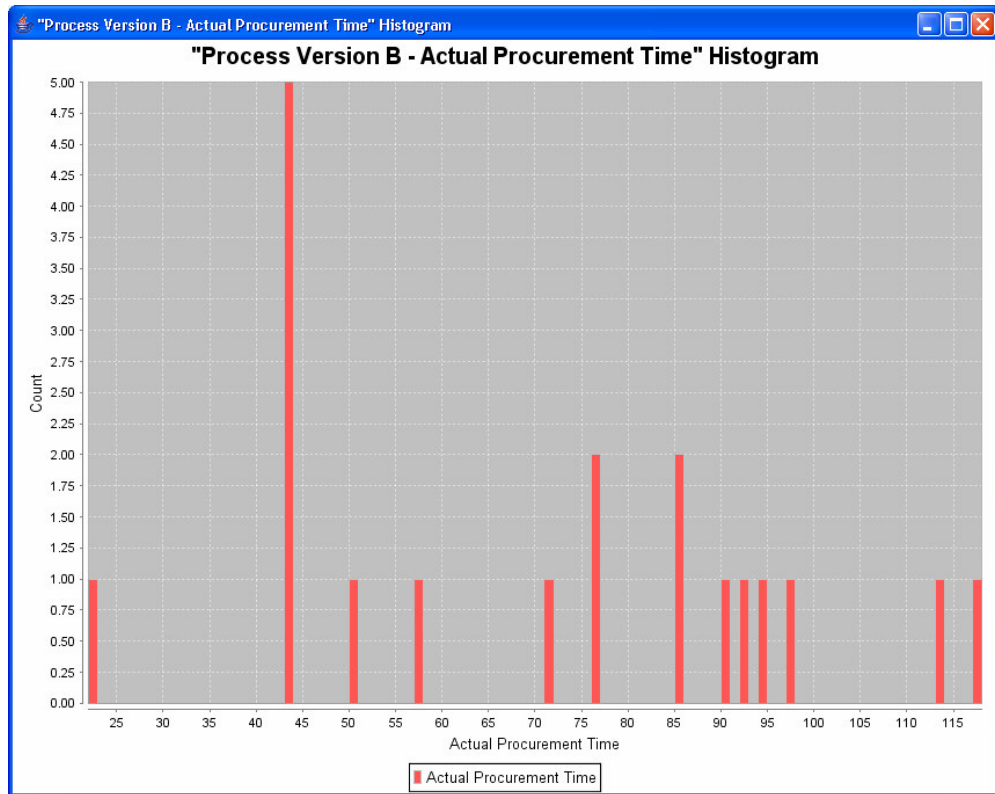


Figure 100 Version B – Actual Procurement Time Histogram

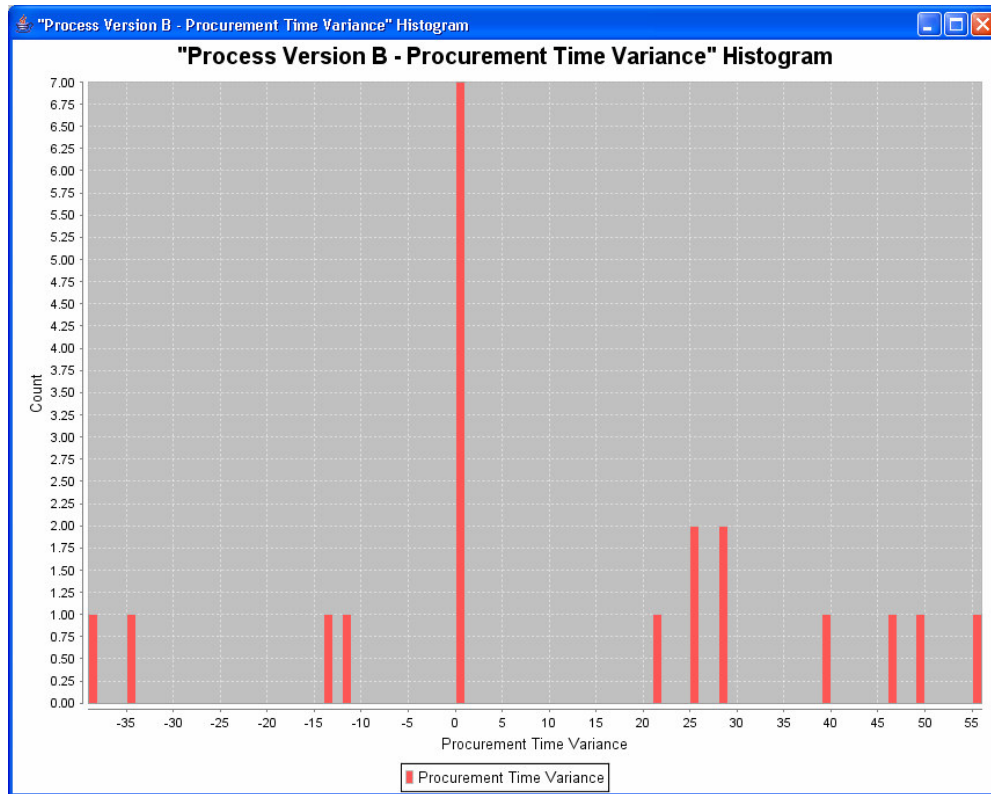


Figure 101 Version B –Procurement Time Variance Histogram

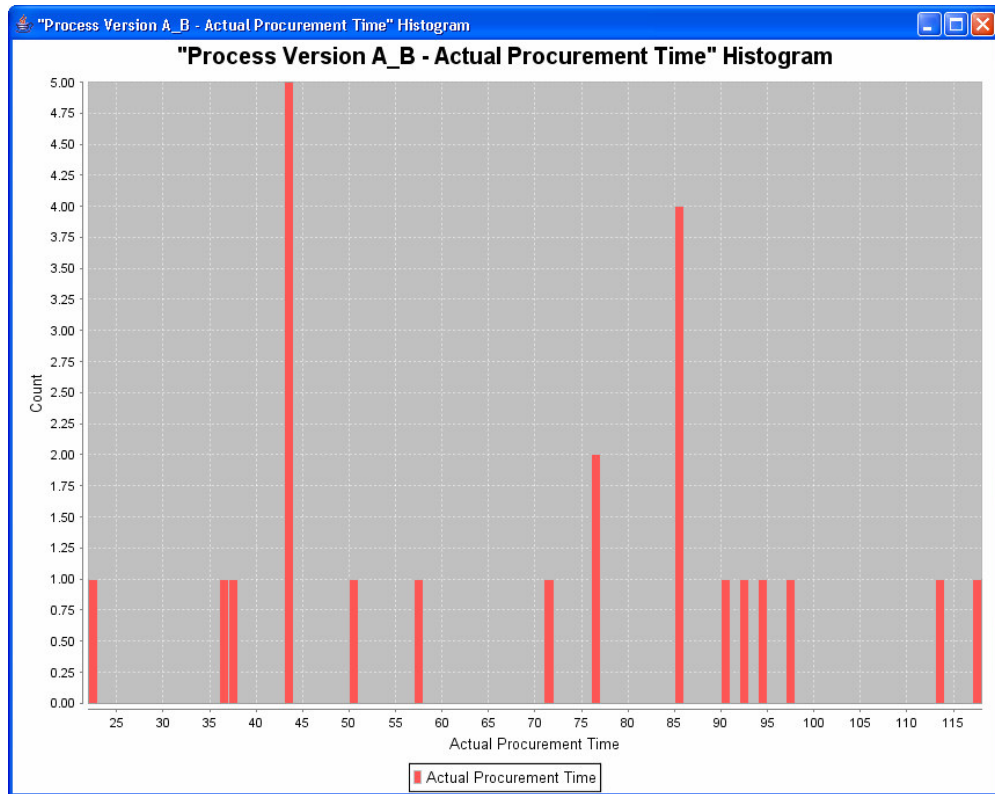


Figure 102 Version A_B – Actual Procurement Time Histogram

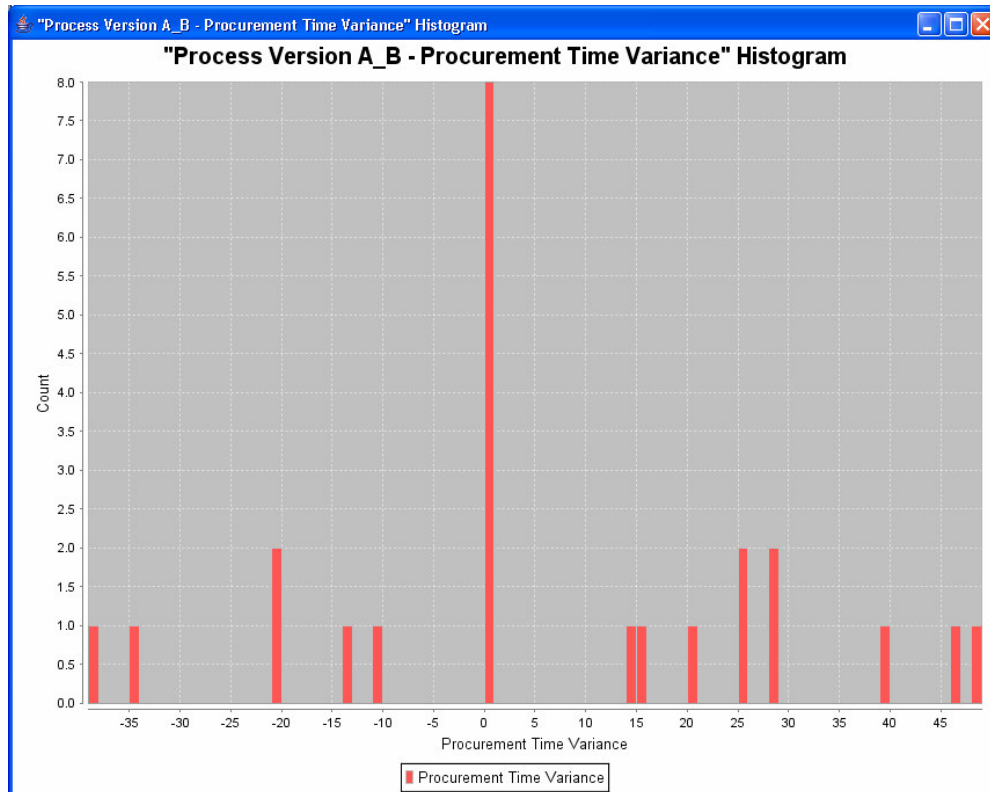


Figure 103 Version A_B –Procurement Time Variance Histogram

We have also drawn bar chart for the requested positions at recruitment, shown in Figure 104. This bar chart revealed that the most requested position at recruitment was Software Engineer (SE). The others were Test Engineer (STE) and Vendor Support (GVS) with 4 instances.

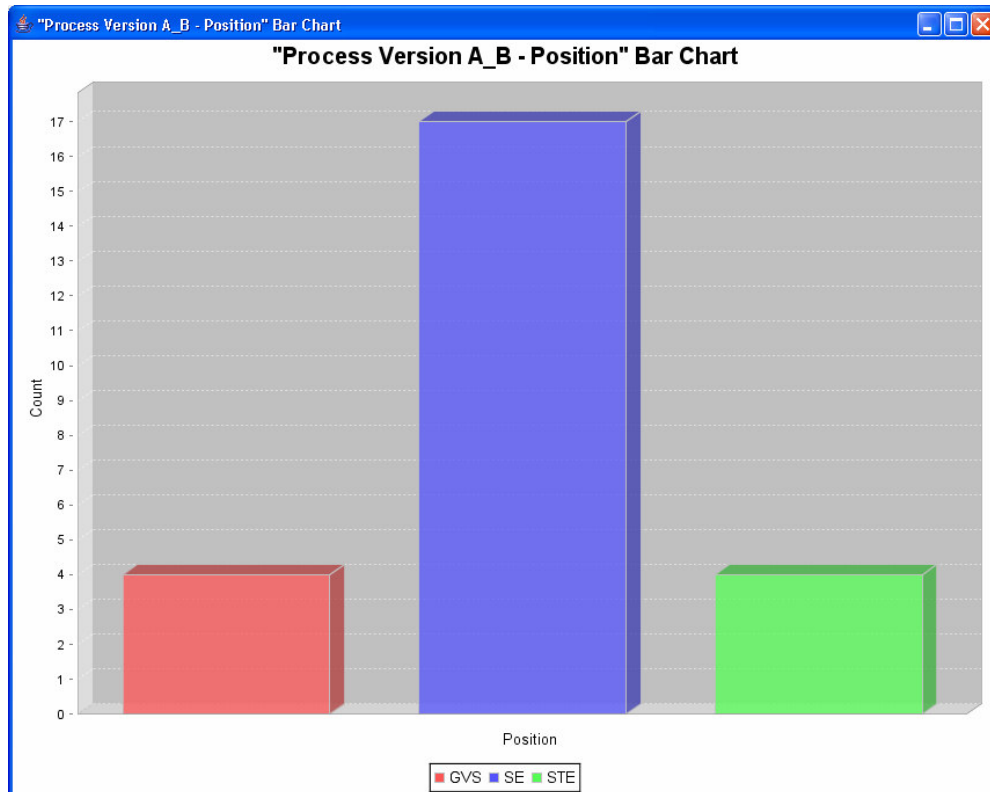


Figure 104 Bar Chart for Position

Findings from the study:

During the implementation of the case study, we have detected improvements about the process analyzed, bugs and improvements for our tool and improvements about the SPC assessment model we have used (SPC-AM) besides gaining inside about the analyzed process. These will be described in detail below:

Inside about the process: We have learned mean values and control limits of process metric data. This information is very precious especially for planning and tracking. We have also concluded that recruitment process is not capable to reach the goals about recruitment. Mean values and control limits of process metrics for each cluster are given in the table below.

Table 10 Results of Case Study B

Process	Metric	Cluster	Mean	LCL	UCL
Recruitment	Actual Procurement Time	Overall	68.87	18.096	119.643
		Version B	70.579	10.739	130.419
	Procurement Time Variance	Overall	6.208	-46.868	59.284
		Version B	10.9	-56.009	77.809

* UCL: Upper Control Limit, *LCL: Lower Control Limit

Improvements about the process: Two possible improvement points have been detected for the analyzed process:

- There are inefficiencies in the recruitment process and they cause recruitment process to take much time than the aimed. To detect the inefficiencies in the process and find possible solutions, process improvement meetings will be arranged together with HR specialists and Managers. Reducing recruitment time is very critical to remain competitive.
- The variance between the planned procurement time and the realized is high. We expect that the variance will be reduced if the process improvement studies about reducing recruitment time reach the goals. But in any case planned dates for recruitment should also be estimated correctly to reduce the variance. Therefore, we will provide the control chart results of our study to Managers to use them in the future recruitment planning. In other words, planning phase of recruitment process will be improved by providing recent control chart results to Managers.

Improvements for our tool: The following improvements are detected during the study:

- Performing statistical analysis among Process Execution Records (PERs) which have the specified metric value (Ex: the specific position looked for, Software Engineer)
- Sorting according to a chosen metric
- Moving Process Execution Records (PERs) up or down in the summary table
- Adding key short cuts for some operations
- Using tab key to navigate between GUI components
- At Process Similarity Matrix, showing Activities in the order they are in PERs

Bugs for our tool: Three minor bugs had been detected during the study and they were corrected later.

Improvements for SPC-AM: The following improvements are detected during the study:

- Questions in “Metric Definition” part of Metric Usability Questionnaire should be enhanced to consider the metrics in Date type.

4.4. Context-3 (Case Study C)

Within the first context, we worked on bug fixing (MR solving) process of a telecom project. We especially focus on one feature of the project because managements needed a detailed analysis about excessive time spent for this feature implemented in the scope of this project. We saw this as a great chance to use our tool to analyze this situation. We chose the bug fixing sub-process because this part of the development process was problematic. Bugs had been reported during component integration tests and system tests. Bugs found during unit tests are not included in the study. Bug fixing process had been performed by using a change management tool which is used organization-wide. Bug fixing for

the selected feature is started at May 2006 and finished at November 2006. We could analyze all the bugs reported during this 6 month period.

Bug fixing process starts when a bug is reported by a submitter via the configuration management tool or via an e-mail. Some bugs found in the project are entered to the tool by a submitter (Tester). The submitter should enter the following fields: subject, problem description, project name, responsible person, priority, status. Creation date and unique ID is automatically created by the tool for the bug reported. Then the responsible person (Developer or Feature Owner) gets an e-mail notification automatically from the tool and starts the analysis. Some bugs are not entered to the tool but submitter sends an e-mail directly to Developer and Developer starts analysis. After deciding on a solution, implementation starts and implemented solution is tested by the Developer. Then changes are checked-in to the configuration management tool. After this step, the state of the bug is changed to “solved” in the change management tool and resolution test is entered to explain the solution implemented if bug is reported via the tool. Also a tester (usually submitter) is assigned to test the implemented solution. After these steps taken by the developer, an e-mail notification is automatically sent to the submitter by the tool. If bug is not reported via the CM tool, explanation about the solution is sent to the submitter via an e-mail.

The analysis was performed together with the owner of the related feature of the project. We spent 5 hours for collecting the data, applying the approach, performing the analyses, and interpreting the results. The set of assets produced during this case study with the control charts are provided in Appendix E.

The study was retrospective, and instead of identifying process attribute values to put on process similarity matrices by filling process execution records, we preferred drawing general process flows with the Feature Owner (FO). We depicted the executions as draft on a paper first together with the FO, and then converted the flows into MS Visio files using eEPC (Extended Event Driven Process Change) notation. The flow for bug fixing process is given in Figure 105.

For the bugs which are reported via CM tool, the metric data had been exported in an MS Excel file. This file can be seen in Appendix B. Metric data is exported in the time order from CM tool since this is very critical for statistical analysis. We first imported this metric data to our tool. Therefore, one process execution record is automatically created for each bug report entry in the Excel file imported. There was 33 data points which were collected via CM tool. Then we used the elements (inputs, outputs, activities, roles, and tools) used to represent process flows showed us typical values of process attributes. Therefore we could add Process Attributes on the similarity matrix and checked against process executions. The process similarity matrix for bug fixing executions is provided in Figure 106.

For the bugs which are not reported via CM tool, we check the e-mails sent to the Developer against the bugs reported for the analyzed feature. We found 29 bug reports notified via e-mail directly from the submitter. We read the creation date and resolution date of bug reports from the Received Date field of bug report e-mails and the Sent Date field of e-mails about resolution in MS Outlook. We have inserted the information collected from e-mails to an Excel file. Then we also imported this data on Excel file to our tool. Therefore, one process execution record is automatically created and appended to the existing table for each bug report entry in the Excel file imported. By the help of eEPC diagram, we created process attributes needed for the newly added process executions and put the checks on similarity matrix accordingly. As a result, we have created 29 more process execution records and this yielded 62 process execution records totally. So that we completed process similarity matrices for Inputs, Outputs, Activities, Roles, and Tools & Techniques of bug fixing process.

But there was a problem at this point: The execution records were not in the time order. We used sorting functionality added to our tool during the last case study to sort the process execution records according to creation date metric values.

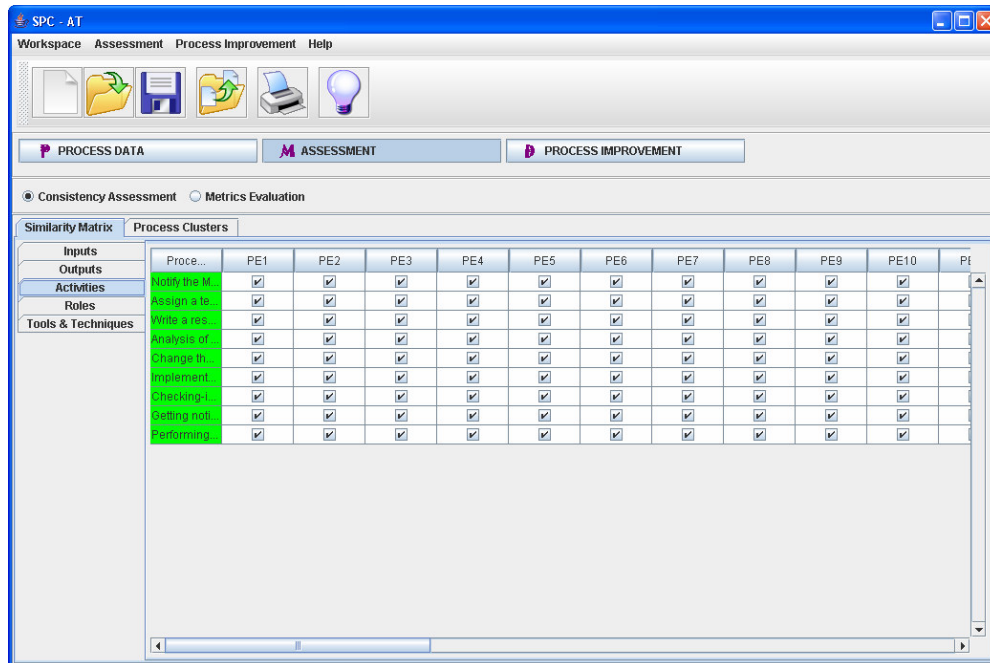


Figure 106 Process Similarity Matrix for Bug Fixing

After finalizing the process similarity matrices, we checked under “Process Clusters” tab-sheet to see the automatically identified process clusters by our tool. Our tool identified 2 process clusters labeled by “Version A” and “Version B” as shown in Figure 107, by observing the similarities between process executions. Version A was representing the bug fixing process where bugs are reported and tracked via CM tool while Version B was representing the bug fixing process where bugs are reported via e-mails. The number of data points was enough for both process clusters (33 and 29) to perform statistical analysis.

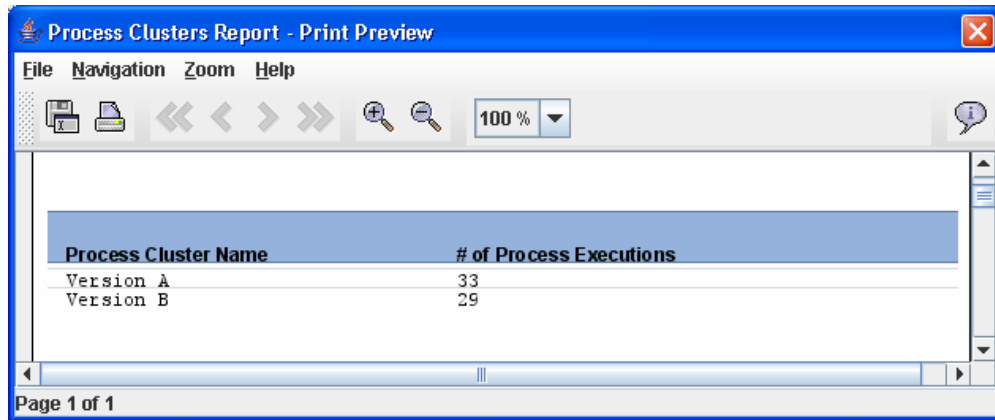


Figure 107 Base Process Clusters for Bug Fixing Process

Table 11 Processes and Data Sets (Original and Derived) in Context-3

Metric Name	Metric Type	Explanation
Creation Date	Base	The date bug is reported
Actual Finish Date	Base	The date bug is resolved
Estimated Finish Date	Base	The last date bug is expected to be resolved
Bug Aging	Derived	Actual Finish Date - Creation Date + 1
Estimated Bug Aging	Derived	Estimated Finish Date - Creation Date + 1
Estimation Variance	Derived	Estimated Bug Aging - Bug Aging
Estimation Capability	Derived	Estimated Bug Aging / Bug Aging
Priority	Base	Priority of the bug reported. Allowed values: 1 (massive), 2 (serious), 3 (little effect), 4 (cosmetic)
Problem Source	Base	Source of the bug reported
Test Result	Base	Result of the test done after correction. Allowed values: T* (successful), T- (unsuccessful), T0 (not tested)
Project Name	Base	Name of the project for which bug is reported
MR Id	Base	Unique Id for the bug

After we identified base process clusters, we have changed our view to Metric Evaluation to assess the usability of the process metrics. The metrics which are imported to our tool from Excel at the beginning of the case study are shown in

the table above as Base metrics. We decided to derive four new metrics with the imported base metrics: Bug Aging, Estimated Bug Aging, Estimation Variance and Estimation Capability. Derived metrics and their formulas are also shown in the above table. The relationships between the base metrics and the derived metrics are shown visually in Figure 108. The metrics at upper side represent the base metrics, all others are derived metrics.

We created one entry under Derived Metrics tab-sheet for each derived metric. After filling Metric Formula attribute for the created entries, metric data for the derived metrics were calculated automatically and stored by our tool.

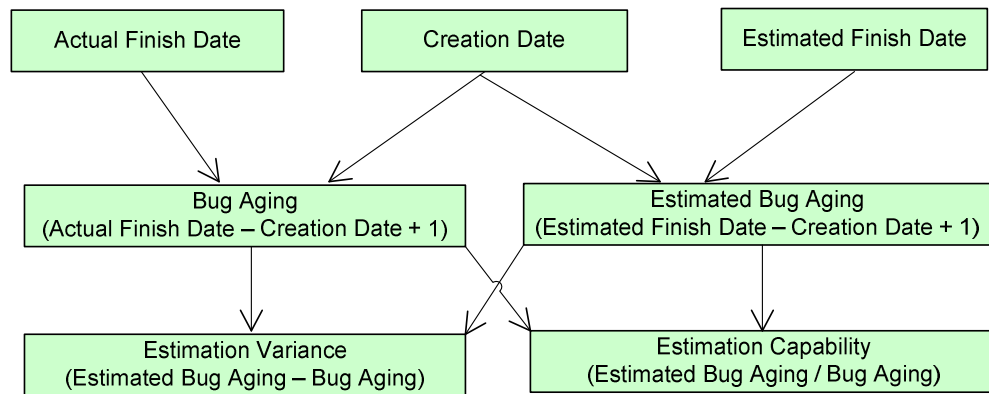


Figure 108 Base and Derived Metrics Identified in Context-3

After deciding on the derived metrics, we filled Metric Usability Questionnaire for each base and derived metric from Questionnaire tab-sheet under Metric Evaluation view. Example questionnaire for “Bug Aging” derived metric is shown in Figure 109 (completed questionnaires for all metrics identified in Context-3 are provided in appendix E). The usability status of all base and derived metrics are listed in Figure 111. In the next step, only metrics which are evaluated as “usable” would be used for control charting. To be clearer, Expected Comp Date, Estimated Bug Aging, Estimation Variance and Estimation Capability metrics are

evaluated as “Not Usable” because of missing metric data points and could not be used for statistical analysis (see Appendix E for details).

The screenshot shows the SPC - AT software interface. The main window has a menu bar (Workspace, Assessment, Process Improvement, Help) and a toolbar with icons for file operations. Below the toolbar are three tabs: PROCESS DATA, ASSESSMENT (selected), and PROCESS IMPROVEMENT. Under the ASSESSMENT tab, there are two radio buttons: Consistency Assessment and Metrics Evaluation (selected). The Metrics Evaluation section has two sub-tabs: Base Metrics and Derived Metrics. The Derived Metrics tab is active, showing a list of metrics on the left, including 'Bug Aging'. The 'Bug Aging' metric is selected, and its details are shown in the 'Metric Usability Assessment Detail' window. This window has three tabs: General Info, Questionnaire, and Usability Rating. The Usability Rating tab is active, displaying a table of questions and answers. The table has columns for Question ID, Question, and Answer. The questions are numbered 6 through 9. The answers are: 'Yes' for question 6, '33' for question 7, '0' for question 8, and 'No' for question 9. Below the table, there is a 'Usability Rating' section with a green circle icon and a dropdown menu showing 'F'. At the bottom right of the window is a 'Save Metric Usability Assessment' button.

Q.	Question	Answer
6	Is metric data existent?	Yes
7	What is the amount of overall observations?	33
8	What is the amount of missing data points?	0
9	Are data points missing in periods? (if yes, please state obse...	No

Usability Rating: F

Save Metric Usability Assessment

Figure 109 Metric Usability Questionnaire for “Bug Aging” Derived Metric of “Bug Fixing” Process

SPC - AT

Workspace Assessment Process Improvement Help

PROCESS DATA ASSESSMENT PROCESS IMPROVEMENT

Consistency Assessment Metrics Evaluation

Base Metrics Derived Metrics

Process Metrics

Metric Name Metr...
Bug Aging

Metric Usability Assessment Detail

General Info Questionnaire Usability Rating

Metric Usability Attributes	Rating	Expected Rating
Metric Identity	F	F
Data Existence	F	F
Data Verifiability	L	L
Data Dependability	L	L
MUF-3&4 for Create Date	L	L
MUF-3&4 for Res Date	L	L

Save Metric Usability Assessment

Figure 110 Metric Usability Ratings for “Bug Aging” Derived Metric of “Bug Fixing” Process

Metric Name	Type	Metric Usability
Pri	Base	Usable
MR Id	Base	NOT Usable
Result	Base	Usable
Create Date	Base	Usable
Res Date	Base	Usable
Bug Aging	Derived	Usable
Problem Source	Base	Usable
Error Reason	Base	Usable
SB Found	Base	Usable
Expected Comp Date	Base	NOT Usable
Estimated Bug Aging	Derived	NOT Usable
Estimation Variance	Derived	NOT Usable
Estimation Capabi..	Derived	NOT Usable

Figure 111 Metric Usability Results (Case C)

After identifying the base process clusters and assessing the usability of the metrics, we have changed our view to Process Clusters (under PROCESS IMPROVEMENT view) to finalize the process clusters. The number of data points was enough (at least 20) for both base process clusters to apply control charting. Therefore we decided to apply control charting on process clusters Version A and Version B separately. But we also decided to apply SPC tools on the process cluster which is produced by merging these two base process clusters to be able to see the overall picture.

SPC tools could only be applied to the qualified process cluster – metric pairs. In this case, these were “Version B – Bug Aging” and “Version A – Bug Aging” for control charting since just Bug Aging metric was assessed as usable and were the type of ratio/absolute. Control charts drawn for these two process cluster – metric pairs are shown in Figure 113 and Figure 116.

When we checked the control chart drawn for “Version B – Bug Aging” pair, we observed two Out-of-Control Points (OCPs). When we checked these OCPs

detected, we observed that last 9 bugs of Version B had been resolved in one day. This is not surprising when we consider the fact that only minor errors remain at the end of testing periods. Therefore, we decided not to count such situations as OCPs and changed the related configuration accordingly (see Figure below). After changing the configuration for OCP Rules we re-drew the control chart in Figure 95, we saw that process was under control. The mean was about 1.5 days and Upper Control Limit (UCL) was about 3.1. According to the control chart results, we conclude that the process is under control. According to the control chart results, we conclude that the process is not only under control but also capable since mean and UCL were consistent with the service level agreement of the company.

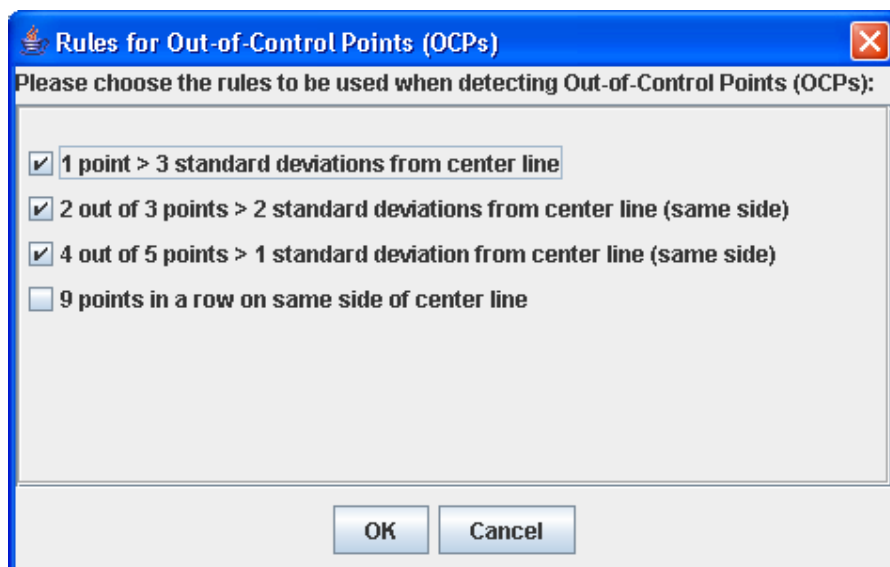


Figure 112 OCP Rules for Case C

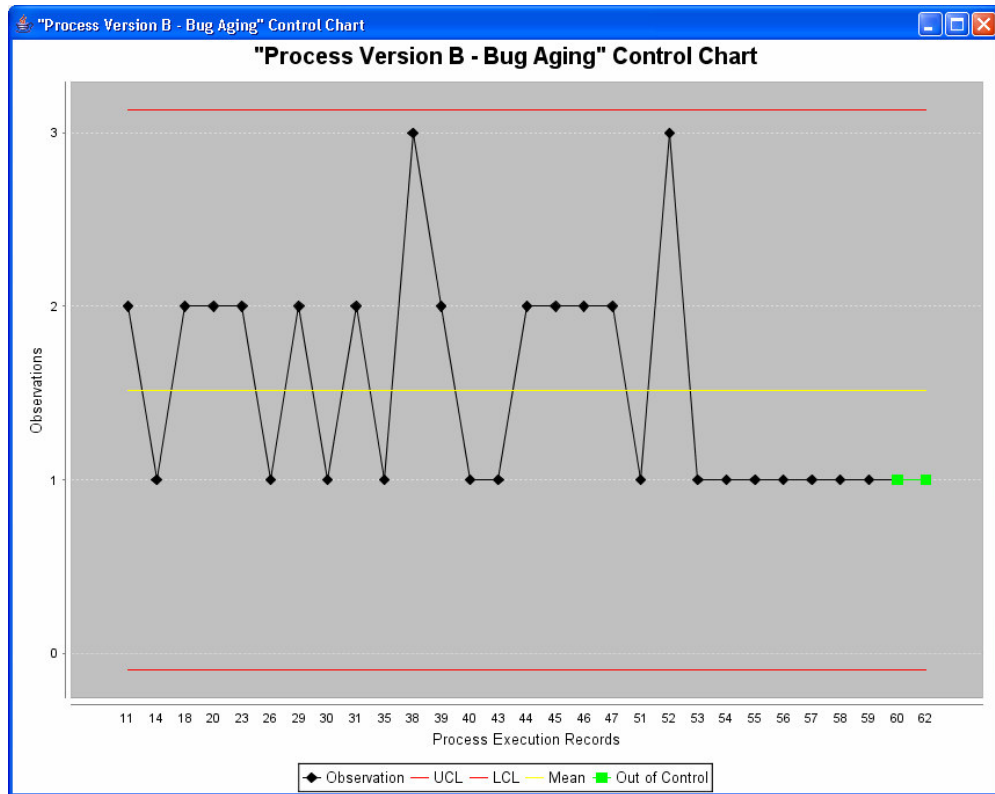


Figure 113 Version B – Bug Aging Control Chart

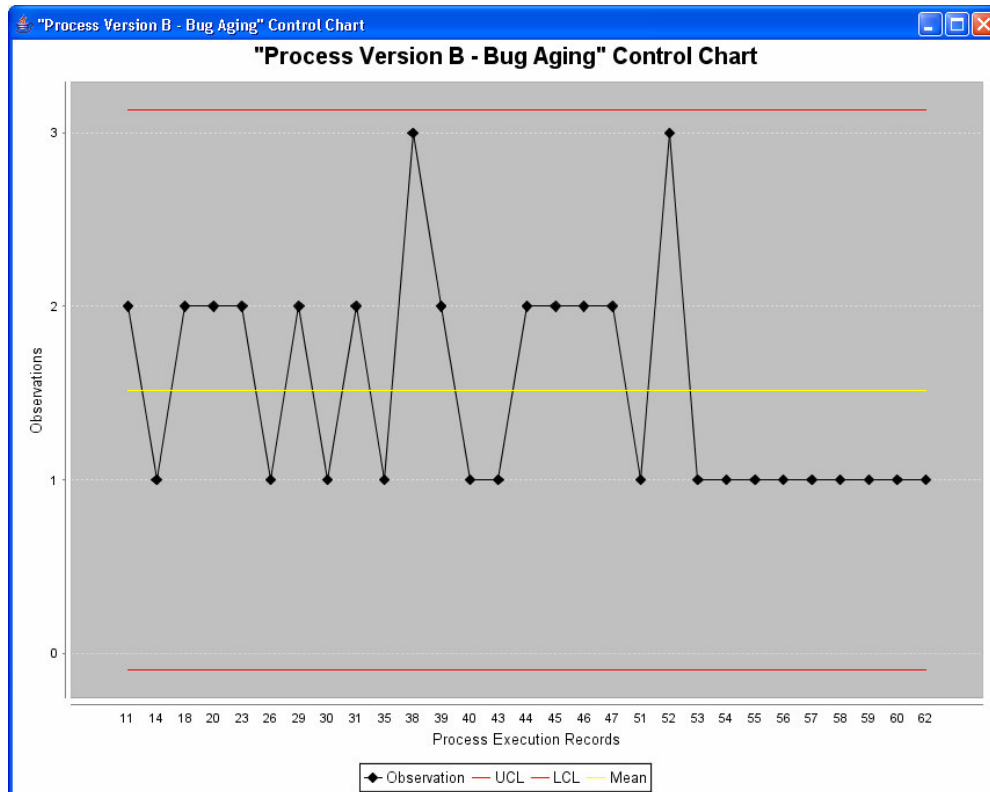


Figure 114 Version B – Bug Aging Final Control Chart

When we checked the control chart drawn for “Version A – Bug Aging” pair, we observed two Out-of-Control Points (OCPs). When we checked the OCP detected (process execution No: 28), we found out that Project Manager had assigned the developer to other more important tasks. We counted this action as an unusual process instance and therefore we decided to exclude this metric data point from analysis and re-drew the control chart. This time, we detected three OCPs. When we filled Process Execution Questionnaire for the detected OCP (process execution No: 22), we have found out that for this bug fixing process instance, there had been a problem about regenerating the error and it had been forgotten for 3 weeks. We have reported this as a possible improvement point for the project. For example, such bugs can be discussed in more details at the weekly

project meetings and final decision can be made for old enough bugs. Then we excluded this metric data point.

After excluding the metric data point from analysis and re-drew the control chart. This time, we detected two OCPs. Although we filled Process Execution Questionnaire for the detected OCP (process execution No: 15), we did not found out any specific reason. Therefore we decided to exclude this metric data point from analysis and re-drew the control chart. This time, we detected just one OCP (process execution No: 21). Bug Aging metric value for this instance was 7 days and this was not an extreme value for a bug with priority 4. Therefore, we decided not to count such situations as OCPs and changed the related configuration accordingly (see Figure 115).

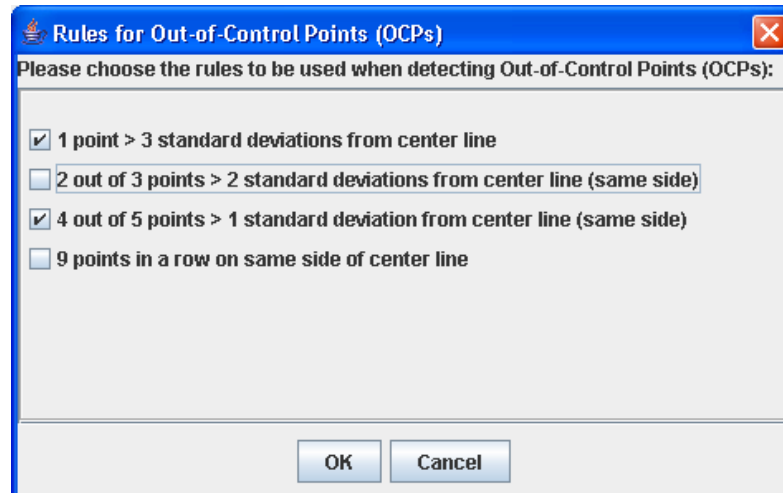


Figure 115 OCP Rules for Case C (Final)

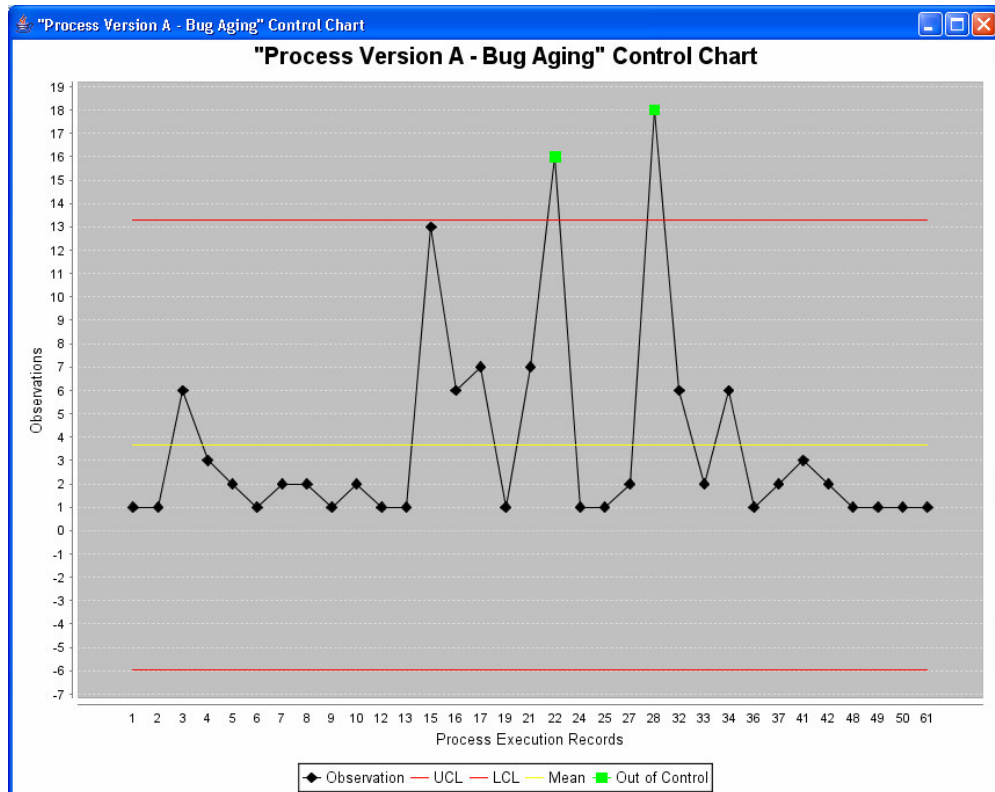


Figure 116 Control Chart drawn for “Version A – Bug Aging” pair

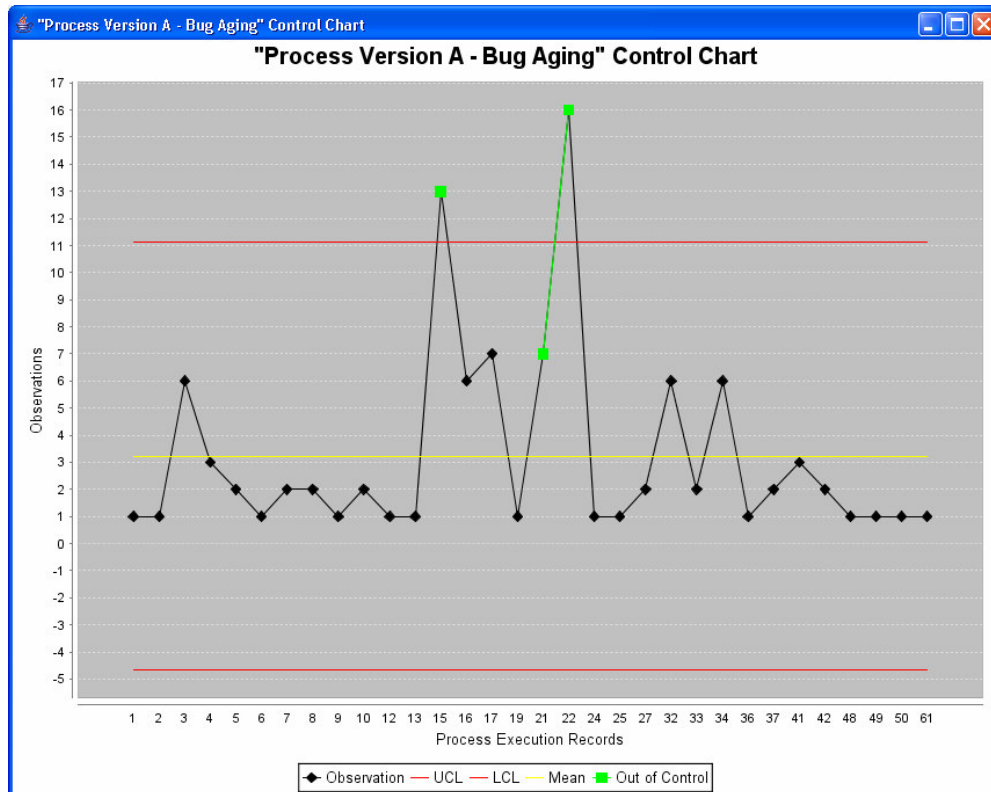


Figure 117 Control Chart drawn for “Version A – Bug Aging” pair

After changing the configuration for OCP Rules we re-drew the control chart in Figure 118, we saw that process was under control. The mean was about 2.5 days and Upper Control Limit (UCL) was about 8 days. According to the control chart results, we conclude that the process is not only under control but also capable since mean and UCL were consistent with the aims for the process.

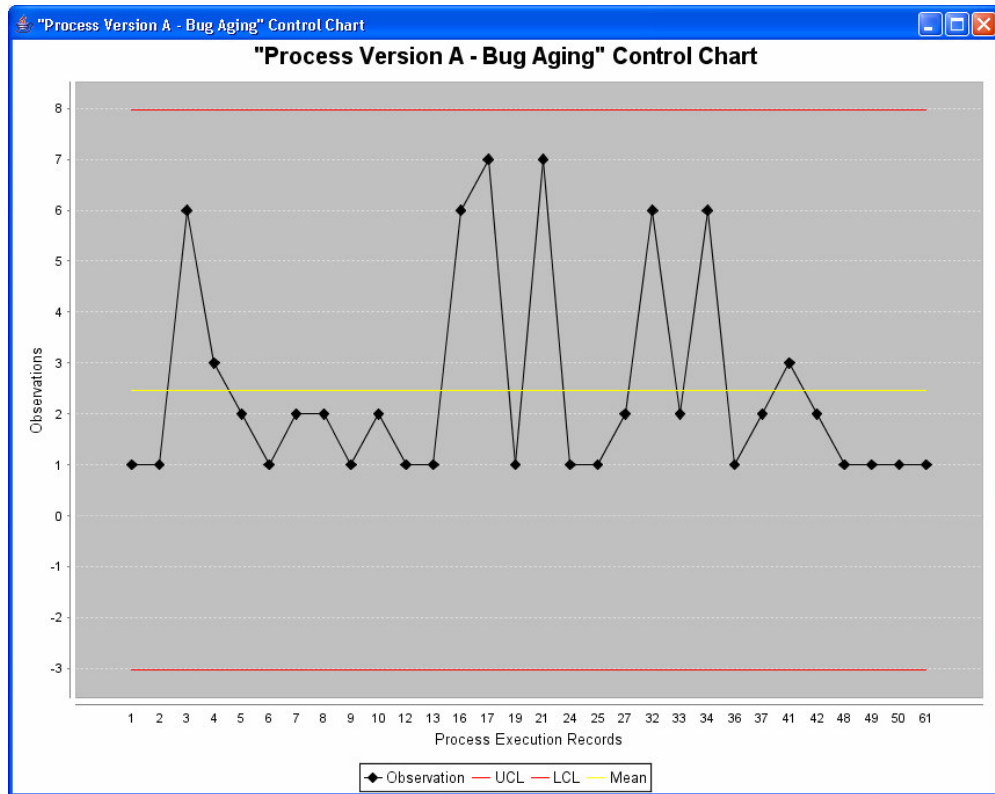


Figure 118 Final Control Chart drawn for “Version A – Bug Aging” pair

Process Cluster Name	# of Process Executions
Version A_B	62

Figure 119 Process Clusters after the merge

As the second step, we merged the two base process clusters and drew control charts for the combined data (see Figure 120). When we checked the control charts drawn for “Version A_B” process cluster, we observed six Out-of-Control Points (OCPs) on both the control chart and these were for the process executions 34, 32, 21, 17, 16 and 3 besides three OCPs excluded before. In other words, Bug fixing process was not under control for “Version A_B – Bug Aging” pair. Since there were many OCPs, we thought this indicated a mixture of multiple cause systems within the process. In this case, Version A and Version B should be analyzed separately since Version A_B was not appropriate for statistical analysis.

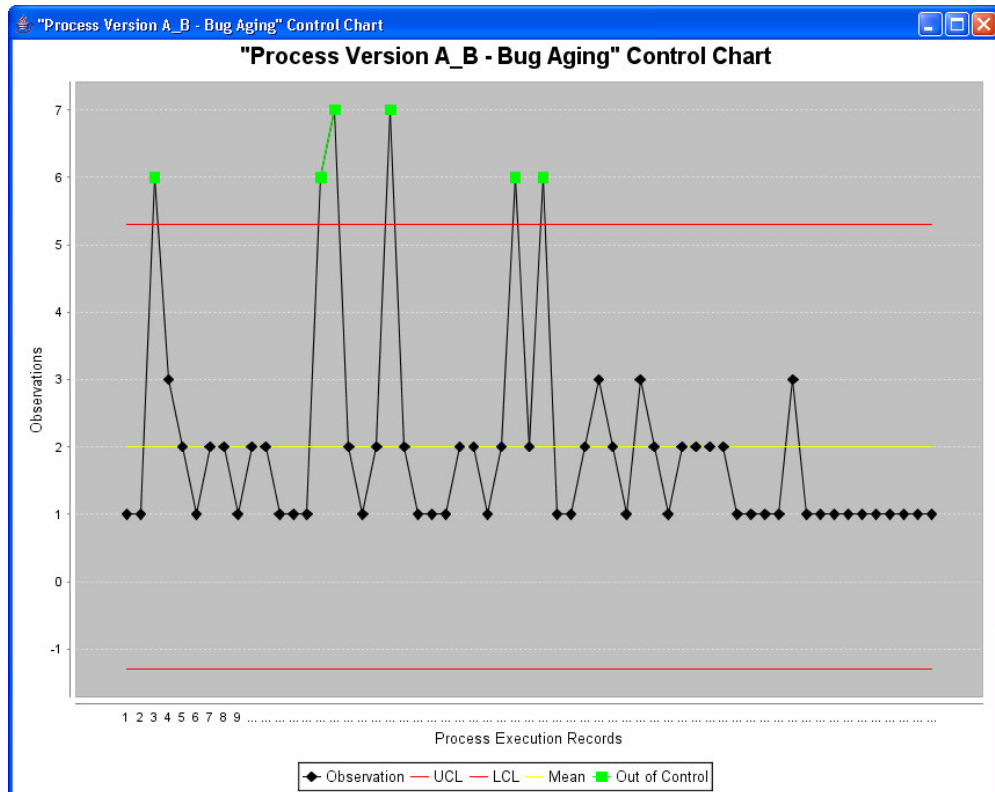


Figure 120 Control Chart for Combined Data of Bug Fixing

A summary of analysis done with control charts can be found in the following tables.

Table 12 Initial Results from Charted Data in Context-3

Process	Metric	Cluster	Status
Bug Fixing	Bug Aging	Overall	Many OCPs
		Version A	3 OCPs
		Version B	Under Control

* OCP: Out-of-Control Point

Table 13 Assignable Causes for Out-of-Control Points in Context-3

Metric	Cluster	OCPs	Assignable Cause
Bug Aging	Version A_B	Many OCPs	Mixture of multiple cause systems
	Version A	3 OCPs	One OCP was due to assignment of the developer to other more important tasks by Project Manager. One of others was due to forgetting a bug (couldn't regenerated) open for 3 weeks. For the last OCP, we couldn't find a specific reason.
	Version B	None	Not applicable

* OCP: Out-of-Control Point

By using our tool, we have also drawn histograms for the metric data which are used at final control charts of Version A, Version B and Version A_B. We have used histograms for under control processes to visualize the frequency distribution of metric data. These can be seen below:

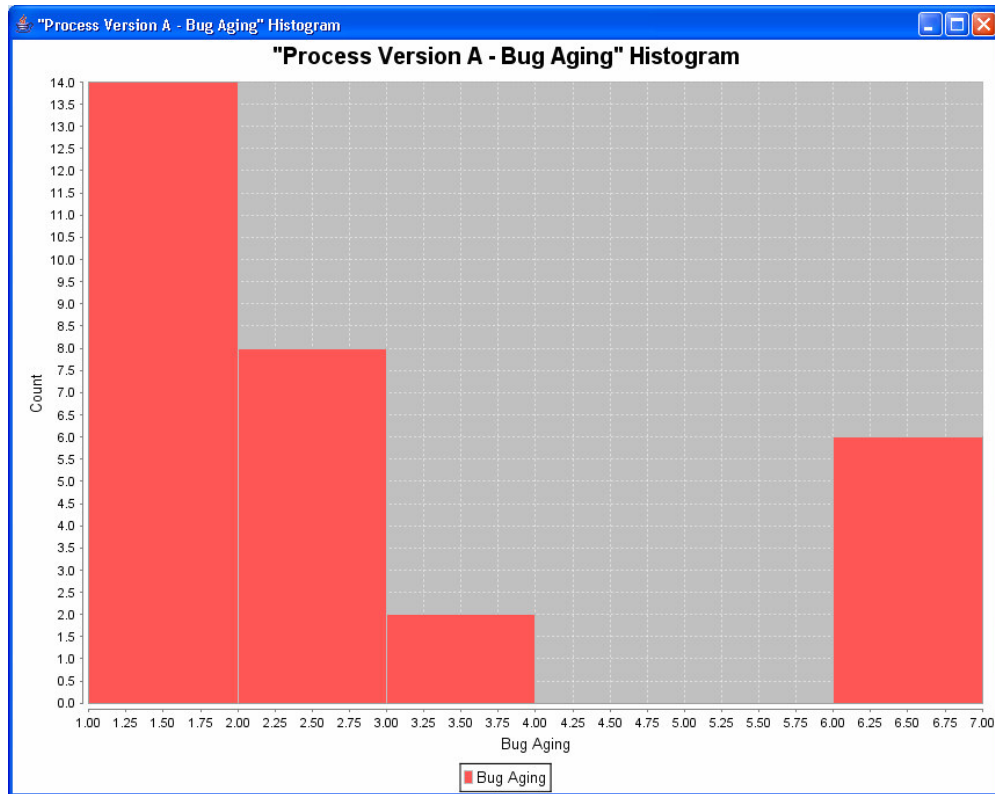


Figure 121 Version A – Bug Aging Histogram

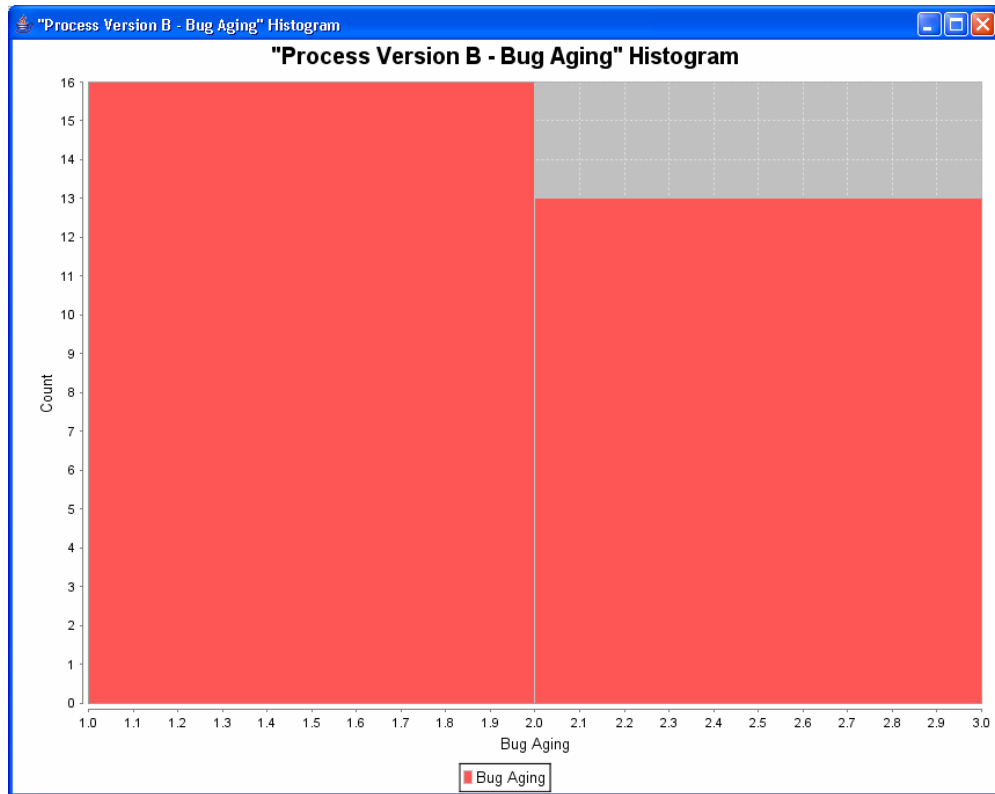


Figure 122 Version B – Bug Aging Histogram

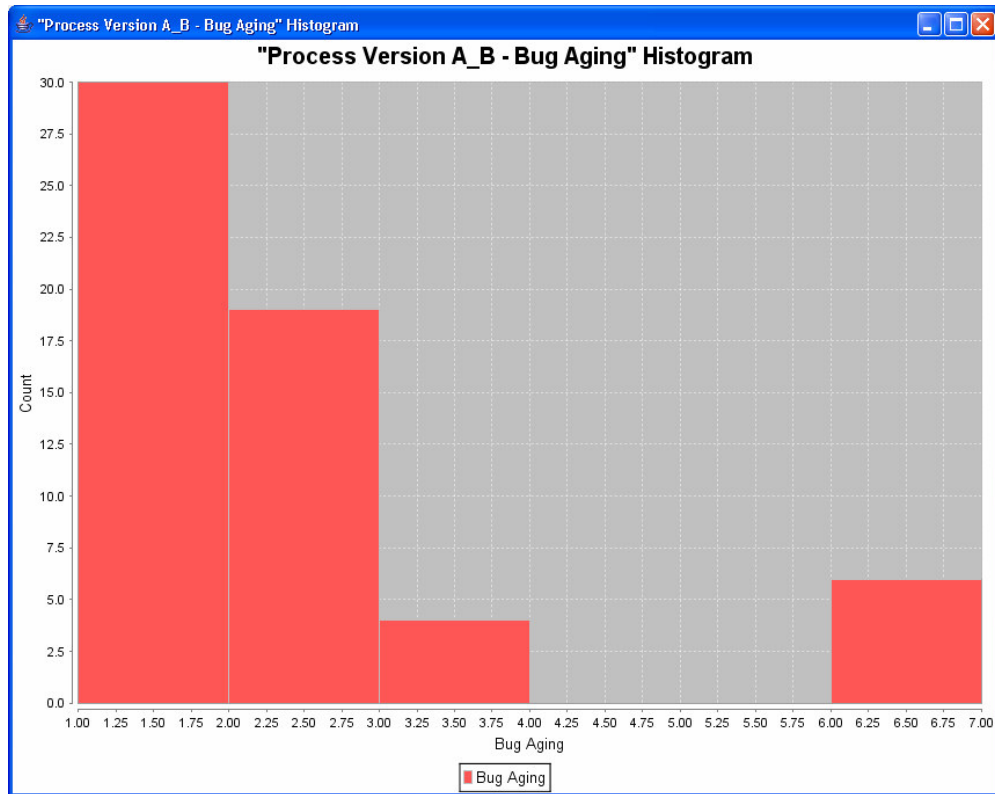


Figure 123 Version A_B – Bug Aging Histogram

Until now we have shown usage of SPC tools for metrics type of ratio or absolute with our tool, but we have also supported analysis of nominal and ordinal metric types. We have drawn bar chart for status of bugs, shown in Figure 124. This bar chart revealed another problem about the project. Normally, when a bug is resolved its status is changed to RES. After that tests are performed to check whether the problem reported is really resolved and status is changed to FIN. When we checked the bar chart, there were some bugs which are in state RES. In other words, we had bugs which had been resolved but not tested. This was also another point for improvement of the project.

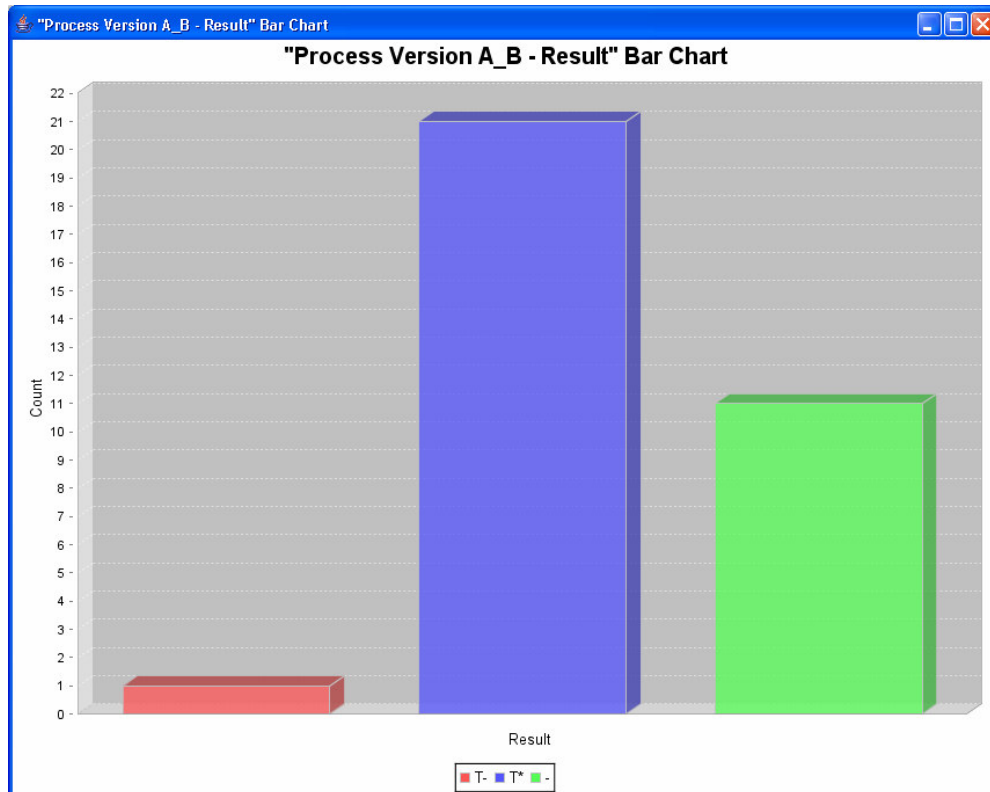


Figure 124 Bar Chart for Test Results

We have also drawn pareto charts for “problem source”, “error reason” and “should-be found” metrics of bugs, shown in Figure 125, Figure 126 and Figure 127. These pareto charts showed us the problematic areas in the whole software development process. When we checked the pareto chart of “should-be found” metric, we saw that about half of the bugs should have been normally detected during Component Integration Testing (CIT). This pointed possible problems about the process of CIT. We decided to initiate a study to detect the problems about CIT. If we could remove all the problems about CIT, number of bugs reported would be reduced to the half. Another major problematic phase of the project was detected as Requirements Inspection. According to the pareto chart, %85 of the bugs could have normally been detected at Requirements Inspection

and CIT. Therefore we decided to initiate a study to detect the problems about Requirements Inspection too.

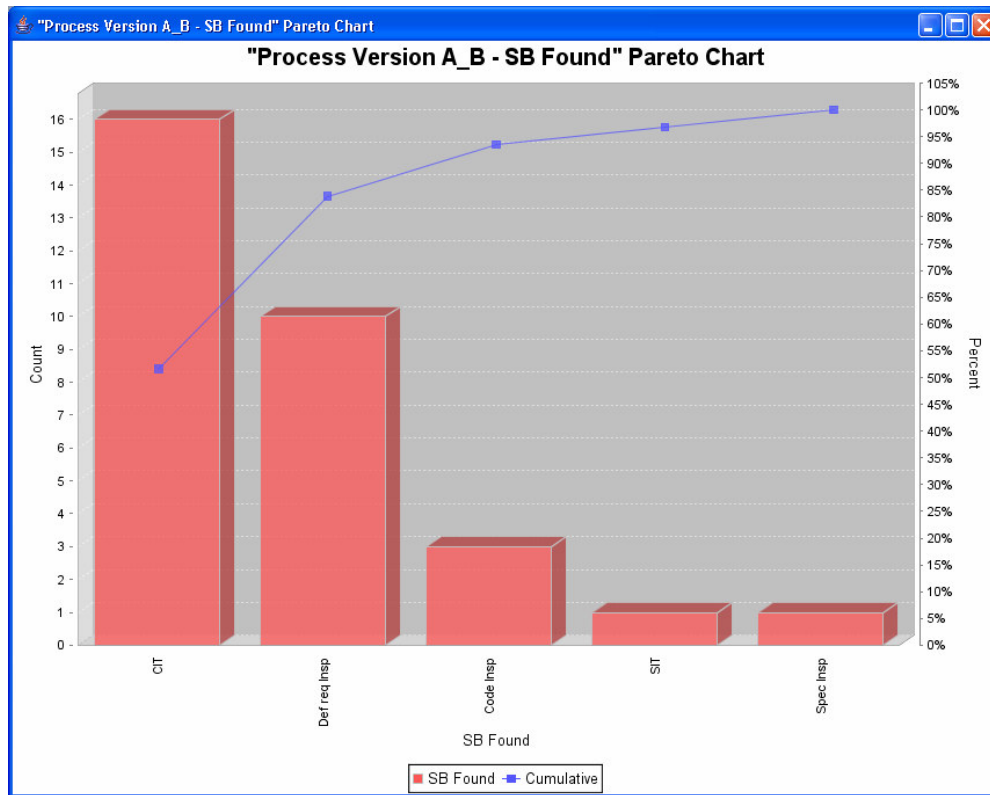


Figure 125 Version A_B – SB Found Pareto Chart

When we checked the pareto chart of “error reason” metric, we reached similar results with the last pareto chart. We saw that error reasons for %75 of the bugs were detected as “Test: Not Escaped” or “Specification: Bad Review”. These results were consistent with the results and decisions from the analysis of “should-be found” metric.

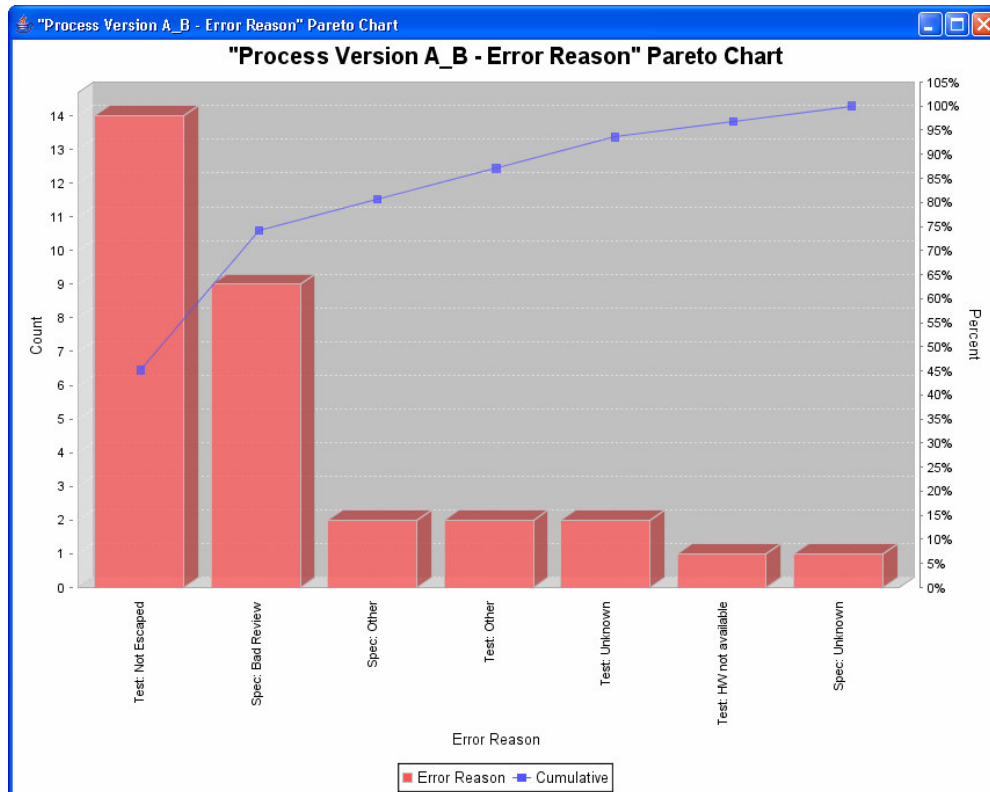


Figure 126 Version A_B – Error Reason Pareto Chart

When we checked the pareto chart of “problem source” metric, we saw that problem sources for %65 of the bugs were detected as “EC: Error in unchanged code”, “IO: Implementation Other”, “RU: Requirement Unclear” or “RM: Requirement Missing”. These were the four major sources for the bugs. The feature whose bug fixing process was analyzed had been very similar to another feature and source code of this feature was heavily reused. This explained the reason for many “EC: Error in unchanged code” as the problem source. Another important observation from the pareto chart was bad quality of the requirements. Missing or unclear requirements were the sources of about %30 of the bugs reported. This was another argument that supported the process improvement initiative for Requirements Inspection.

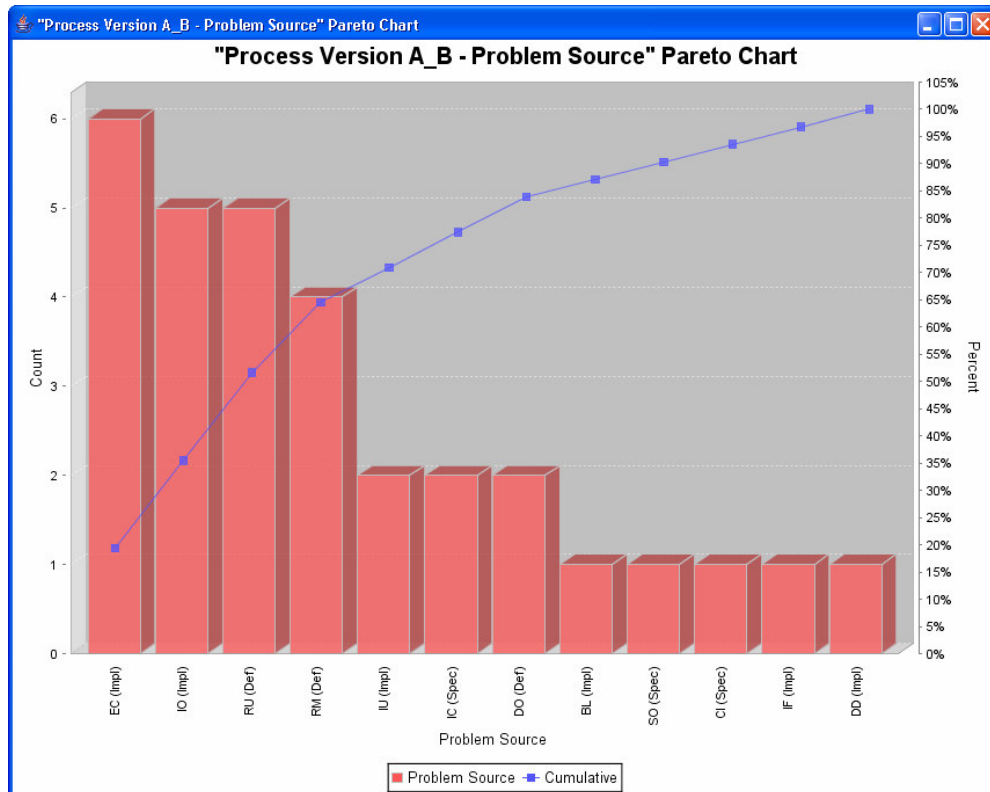


Figure 127 Version A_B – Problem Source Pareto Chart

Findings from the study:

During the implementation of the case study, we have detected improvements about the process analyzed, bugs and improvements for our tool and improvements about the SPC assessment model we have used (SPC-AM) besides gaining inside about the analyzed process. These will be described in detail below:

Inside about the process: We have learned mean values and control limits of bug fixing process of the project. This information is very precious especially for planning and tracking. Mean values and control limits of process metrics for each cluster are given in the table below.

Table 14 Results of Case Study C

Process	Metric	Cluster	Mean	UCL
Bug Fixing	Bug Aging	Overall	2.000	-
		Version A	2.467	7.969
		Version B	1.517	3.132

* UCL: Upper Control Limit

Improvements about the process: Two possible improvement points have been detected for the analyzed project:

- Bugs which can not regenerated for one week can be discussed in the project meeting and decision should be given. Currently, this is not considered in the process.
- Problems about testing of resolved bugs are detected. There are some bugs which are resolved but not tested. For each bug resolved, a tester can be assigned and testing can be tracked. This should be a part of project management.
- We have detected %85 of the bugs could have normally been detected at Requirements Inspection and CIT. Therefore process improvement studies were initiated for these two sub-processes of software development process.

Improvements for our tool: The following improvements are detected during the study:

- For some Out-of-Control points shown in the CCs, it is understood that they are not out-of-control actually. There can be some way to ignore these points.
- “No” field of Process Attributes can be automatically assigned during creation

- Combo boxes can be used for the answers field of the appropriate questions at questionnaires.
- “Recorded On” field can be set automatically according to the current time while creating Process Execution Records (PERs)

Bugs for our tool: Four minor bugs had been detected during the study and they were corrected later.

Improvements for SPC-AM: The following improvements are detected during the study:

- Metric usability ratings can be assigned automatically according to the answers given at Metric Usability Questionnaire.
- Questions in “Process Environment” part of Process Execution Questionnaire (PEQ) concentrate on a change in the process. But may be the cause for an OCP is not a change but a mistake or missing thing in the current process. Questions can be enhanced to consider this.
- “Is metric data recorded precisely?” question at Metric Usability Questionnaire is not clear. This question can be enhanced to increase the understandability.
- The questions in the “*Data Dependability*” part of Metric Usability Questionnaire do not intuitively direct the assessor for the correct usability rating.

4.5. User Evaluation

The SPC-AAT has been evaluated by the users of the tool during case studies. A questionnaire has been filled by these users for evaluating our tool. In the questionnaires, the tool is rated according to the criteria which were defined as critical according to our aims. In addition, open-ended questions are also supplied for getting additional comments from the users. The filled questionnaires are provided in Appendix F.

According to the user evaluations, we can conclude that SPC-AAT satisfied the aims set at the beginning of the study. All three users have agreed about the success of SPC-AAT about meeting the stated objectives and adequate utilization of SPC tools. In the first questionnaire, we got feedbacks about the problems related with user friendliness and use of SPC-AAT. These problems were fixed later and with the next questionnaires SPC-AAT has been evaluated as user friendly and easy to install and use. These were also some important aims targeted in our study. According to the questionnaires, the most problematic area is detected as help documentation. This is added as a future work for our study.

The users who evaluated the tool proposed some improvements concerning both the functionality and usability of the tool. Some of them are also noted and added as future work.

The positive feedbacks of the Manager from the organization were also very important for our study (see questionnaire 2 in Appendix F). The Manager was very eager to use SPC-AAT in the other processes of the organization. This also proves that the people who were involved in our case studies saw and believed the usefulness of our tool and eager to use in the future.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1.Conclusion

Statistical process control (SPC) which includes very powerful techniques used in other mature engineering disciplines for managing projects with allowed variation is not used by many software organizations. To disseminate and effectively use SPC especially for emergent and/or low maturity organizations, guidelines and software tools to implement SPC techniques should be developed. In this study, we developed a software tool (SPC-AAT) to assess the suitability of software processes and metrics for SPC and use SPC tools.

SPC-AAT aims to ease and enhance application of SPC especially for emergent and/or low maturity organizations and reduce the time required to implement SPC. SPC-AAT works integrated with the other tools in the environment which hold measurement data about the processes performed in the organization. There are two ways to integrate these tools to SPC-AAT: using INTERMEDIATE infrastructure or using CSV, Excel file exports directly from the tools. For INTERMEDIATE infrastructure, a collector which is a small application that invokes the measurement tool and produces metric values is used to provide measurement data to SPC-AAT as an XML file from other tools. By using INTERMEDIATE infrastructure, commercial measurement tools, custom-made applications and databases can be integrated to SPC-AAT. When measurement

data is imported to SPC-AAT, all necessary assets are created automatically by the tool before SPC assessment and analysis are started. We used SPC-AM as the assessment model to test the suitability of SPC for software processes and metrics. In the scope of SPC-AM, process attributes are defined for process executions and process clusters are identified by using this info. While doing this, insight about the process analyzed is gained. Then metrics to use for statistical analysis are identified and a questionnaire is filled for each metric. In this way, the characteristics of metric data are discovered and performance of basic measurement practices during data collection is investigated. At the end of suitability assessment, qualified “process cluster – process metric” pairs are identified for using SPC tools on them. On the qualified “process cluster – process metric” pairs, control charts, histograms, bar charts and pareto charts are applied as the supported SPC tools by SPC-AAT. In this way, SPC-AAT guides software organizations to use SPC tools in a correct and efficient way on their processes and metrics.

To validate our model, we performed three case studies in a multiple-case-study context. For each of the case studies, we identified the different versions of the process (process clusters), evaluated the usability of process metrics and performed SPC analysis for the suitable process clusters and metrics. We worked on recruitment process at one organization and bug fixing processes of two different projects at the same organization. The organization at which case studies were performed is a software development organization having CMMI L3. The processes analyzed were not defined explicitly by the company. In the first case, we investigated utilization of bug aging, effort and status metrics of bug fixing process of an integration project. In the second case, we worked on planned procurement time, actual procurement time, procurement time variance, and position metrics of recruitment process of the software development organization. In the third case, we worked on bug aging, estimated bug aging, estimation variance, estimation capability, error reason, problem source, should-be found and status metrics of bug fixing process of another project in the organization.

During the case studies we observed the benefits of SPC-AAT clearly. Since SPC-AAT supports common file formats in Excel and CSV to which most of the tools can export their data, we saved time during collecting metric data. In the previous studies, it was stated that most of the effort for the analysis is spent during collecting and organizing metric data [43] [14] [47] [48]. We can say that SPC-AAT significantly enhance organizing metric data process and reduce time required as it was aimed at the beginning of the study. As it is known metric data should be in the time order to draw the control charts. If it is not satisfied then results of the control charts will not be dependable. With the sorting feature of SPC-AAT metric data could be put into time order according to any chosen appropriate metric during case studies. We can again say that SPC-AAT significantly enhance organizing metric data process and reduce time required as it was aimed at the beginning of the study.

Another observation about benefits of SPC-AAT is that SPC-AAT could guide users to define, understand metrics and choose the appropriate ones for analysis by the help of questions answered by the users. With little expertise in the area, users could decide the metrics which can be used in the statistical analysis just by answering the questions in the questionnaires. With the reporting feature of SPC-AAT metric definition reports have been taken, so that metrics had been defined implicitly while filling metric usability questionnaires. Such reports are also requested by CMMI; therefore this can be counted as another benefit of metric definition reports. We also observed that SPC-AAT could guide users to choose correct SPC tools for the metrics. SPC-AAT restricts the SPC tools that can be used according to the type of the metric data (numeric or nonnumeric). Therefore users did not need any expertise to choose the SPC tools to be used as it was aimed at the beginning of the study. Especially for control charts, there are lots of different types for different metrics and distributions. SPC-AAT hides these details from the users and makes it simple.

SPC-AAT could also guide users to perform rational sampling which is very critical for statistical analysis to guarantee a single and constant system of chance

causes. SPC-AAT hides details of rational sampling from users. Users even did not hear about rational sampling but SPC-AAT gets the necessary input and performs rational sampling. Users just enter the attributes of process executions (inputs, outputs, activities, roles, tools & techniques) and SPC-AAT automatically identifies the process clusters. Therefore we can say that SPC-AAT was successful to ease rational sampling process by hiding the details and reduce the time required. We also observed that defining new derived metrics by using existing base or derived metrics was quite easy for users. User could define new metrics by just typing the name and the formula of the new derived metric and metric values for all process executions could be calculated automatically. We can say that SPC-AAT enhance defining derived metrics and reduce the time required for calculation as it was aimed at the beginning of the study.

Drawing control charts, histograms, pareto charts and bar charts without needing a third party tool was also one of the benefits of SPC-AAT. This feature reduced the time required for statistical analysis by providing a central place to analyze the metric data besides collecting, organizing and assessing. During case studies another observation about benefits of SPC-AAT is that SPC-AAT could guide users to interpret the chart outcomes and detect possible problems. For control charts, assignable causes (out of control points) are shown as large green points, and when a green point is clicked on, a questionnaire is opened to find out the reasons that cause this extraordinary measure. When questionnaire is filled, user can choose to exclude this point if necessary and control chart is refreshed by recalculating the mean and the limits. The same process is started again at this point. We observed that this was very intuitive to the users and it could be easily realized. In this way, users could detect the assignable causes which make the analyzed process out-of-control. We can again say that SPC-AAT significantly enhance interpreting chart outcomes and reduce time required as it was aimed at the beginning of the study.

Another observation about benefits of SPC-AAT is that SPC-AAT could ease the what-if analysis done for different rational sampling choices. In other words,

SPC-AAT provides merging and splitting of process clusters and each merge or split can be regarded as a different rational sampling. After merging or splitting process clusters, users could see the analysis results immediately and could decide on the correct rational sampling easily. We can say that SPC-AAT is very useful for SPI purposes.

The results of all these enhancements can be seen from the time spent during case studies. We spent 5.5 hours for case study-A, 5 hours for case study-B, and 5 hours for case study-C.

Besides easing applying SPC and reducing the time required, we have detected improvements about the processes analyzed, bugs and improvements for SPC-AAT and improvements about the SPC assessment model we have used (SPC-AM). We have also gained insight about the analyzed processes by knowing mean values and control limits of the processes. This information is very precious especially for planning and tracking of the projects.

With SPC-AAT, we also contributed to the software engineering domain by developing a tool which relate process metric data to process data for statistical analysis. Using both data for statistical analysis in one tool is rare.

5.2.Future Work

During this study, we had some limitations because of time and data constraints. Nevertheless, this study is a step to effectively use and disseminate statistical process control in software industry and a good opportunity is left for researchers who want to get into the issue in more detail. The following are some of the possible items for further research studies in addition to the minor improvements detected during case studies:

- SPC-AAT can be enhanced to monitor all the necessary processes in a software organization at the same time. In this way, SPC-AAT can be used as the central place in a software organization on which all necessary

processes and metrics are defined and the performance of the processes are tracked.

- Integrating Goal-Question-Metric (GQM) approach to our tool to be able to choose the metrics according to the goals of the organization
- Adding “Process Capability Evaluation” feature to report process capability, define the target capability and track changes in the process capability with process improvements. Mean and control limits on control charts can be used to identify process capability.
- Reading process definition from a file or providing a GUI to define process visually and generating process attributes automatically from this
- Performing statistical analysis not just on process clusters but adding a second dimension: metric value (Ex: the specific position looked for, Software Engineer)
- Providing an enhanced help documentation by using an third party java library
- Implementing a “Wizard Mode” to guide the User by showing the next steps to be taken
- Supporting more SPC tools. Scatter Diagrams can be useful to detect the relations between the metrics

REFERENCES

- [1] Argyris, C., “Good Communication that Blocks Learning”, HBR, July-August 1994.
- [2] Barnard, J., and Carleton, A., “Analyzing a Mature Software Inspection Process Using Statistical Process Control”, National SEPG Conference, Pittsburgh, 1999.
- [3] Basili, V.R., "Software Modeling and Measurement: The Goal Question Metric Paradigm," Computer Science Technical Report Series, CS-TR-2956 (UMIACS-TR-92-96), University of Maryland, College Park, MD, September 1992.
- [4] Basili, V.R., Caldiera, G., and Rombach, H.D., “The Goal Question Metric Approach”, Encyclopedia of Software Engineering, Vol.1, pp.528-532, John Wiley & Sons, 1994.
- [5] Burr, A., and Owen, M., Statistical Methods for Software Quality, International Thomson Computer Press, 1996.
- [6] Card, D., “Statistical Process Control for Software?”, IEEE Software, pp.95-97, May 1994.
- [7] Carleton, A.D., and Paulk, M.C., “Can Statistical Process Control Be Usefully Applied To Software?”, 4th European Software Engineering Process Group (ESEPG) Conference, Amsterdam, June 1999
- [8] CMU/SEI, “Process Maturity Profile of the Software Community – 2000 Year End Update” (presentation), March 2001(a).

- [9] CMU/SEI, Capability Maturity Model Integration – Version 1.1, Technical Report (Continuous: CMU/SEI-2002-TR-001, Staged: CMU/SEI-2002-TR-002), December 2001(b).
- [10] CMU/SEI, The 2001 High Maturity Workshop, CMU/SEI-2001-SR-014, January 2002.
- [11] Deming, W.E., Statistical Adjustment of Data, John Wiley and Sons, 1943. (Re-printed by Dover Publications, July 1984.)
- [12] Deming, W.E., Out of the Crisis, Massachusetts Institute of Technology, Center of Advanced Engineering, Cambridge, Mass., 1986.
- [13] Demirörs, O., and Sargut, K.U., “Utilization of a Defect Density Metric for SPC Analysis”, 13th International Conference on Software Quality, Dallas, Texas, October 2003.
- [14] Demirörs, O., and Sargut, K.U., “Utilization of Statistical Process Control (SPC) in Emergent Software Organizations: Pitfalls and Suggestions”, Software Quality Journal, Vol.14, No.2, pp.135-157, 2006.
- [15] Drucker, P.F., “The New Society Organizations”, HBR, September-October, 1992.
- [16] Fenton, N.E., and Pfleeger, S.L., Software Metrics: A Rigorous and Practical Approach (2nd Ed.), PWS Publishing Company, 1997.
- [17] Florac, A.W., Carleton A.D., Measuring the Software Process: Statistical Process Control for Software Process Improvement. Pearson Education, 1999 (a). ISBN 0-201-60444-2.
- [18] Florac, A.W., Carleton A.D., “Statistically Controlling the Software Process” (The 99 SEI Software Engineering Symposium), Software Engineering Institute, Carnegie Mellon University, September 1999 (b).

- [19] Florac, A.W., Carleton A.D., and Barnard, J.R., “Statistical Process Control: Analyzing a Space Shuttle Onboard Software Process”, IEEE Software, July/August 2000, pp.97-106.
- [20] Florac, A.W., Park, R.E., and Carleton A.D., “Practical Software Measurement: Measuring for Process Management and Improvement”, Guidebook: CMU/SEI-97-HB-003, 1997.
- [21] Florence, A., “Statistical Process Control Applied to Software Requirements Specification Process”, 10th European Software Engineering Process Group (ESEPG) Conference, London, June 2005.
- [22] Grady, R.B., and Caswell, D.L., Software Metrics: Establishing a Company-Wide Program, Prentice Hall PTR, 1998.
- [23] Hall, T., and Fenton, N., “Implementing Effective Software Metrics Programs”, IEEE Software, 1997.
- [24] Hetzel, W.C., Making Software Measurement Work: Building an Effective Software Measurement Program, QED Publishing, Wellesley, Mass., 1993.
- [25] Ishikawa, K., *Guide to Quality Control*. Asian Productivity Organization, 1982. ISBN 92-833-1035-7.
- [26] ISO, “ISO 9001: Quality Management Systems – Requirements”, 2000.
- [27] ISO/IEC JTC1/SC7, “ISO/IEC TR 15504: Information Technology – Software Process Assessment”, 1998(b).
- [28] ISO/IEC JTC1/SC7, “ISO/IEC 15939: Software Engineering – Software Measurement Process”, 2002.
- [29] Jacob, L., and Pillai, S.K., “Statistical Process Control to Improve Coding and Code Review”, IEEE Software, May/June 2003, pp.50-55.
- [30] Jalote, P., Dinesh, K., Raghavan, S., Bhashyam, M.R., and Ramakrishnan, M., “Quantitative Quality Management through Defect Prediction and

Statistical Process Control”, Proceedings of Second World Quality Congress for Software, September 2000.

- [31] Lantzy, M.A., "Application of Statistical Process Control to Software Processes", WADAS '92. Proceedings of the Ninth Washington Ada Symposium on Empowering Software Users and Developers, 1992, pp.113-123.
- [32] McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J., and Hall, F., Practical Software Measurement: Objective Information for Decision Makers, Addison-Wesley Professional, 1st edition, 2001. ISBN 0201715163.
- [33] MINITAB Statistical Software, Release 14,
<http://www.minitab.com/products/minitab/14/default.aspx>, Last Access
Date: 19 February 2007
- [34] Montgomery, D.C., *Introduction to Statistical Quality Control (5th Ed.)*. John Wiley & Sons, Inc., U.S.A., 2005. ISBN 0-471-65631-3.
- [35] NATO, “AQAP-150: NATO Quality Assurance Requirements for Software Development (Edition 2)”, September 1997.
- [36] Offen, R.J., and Jeffery, R., “Establishing Software Measurement Programs”, IEEE Software, 1997.
- [37] Park, R.E., Goethert, W.B., and Florac, W.A., “Goal-Driven Software Measurement”, CMU/SEI-96-HB-002, August 1996.
- [38] Paulk, M.C., Weber, C.V., Curtis, B., and Chrissis, M.B., The Capability Maturity Model: Guidelines for Improving Software Process, Addison-Wesley Publishing, October 1995.
- [39] Paulk, M.C., “Practices for High Maturity Organizations”, Proceedings of the 1999 Software Engineering Process Group Conference, Atlanta, Georgia, March 1999, pp.28-31.

- [40] Paulk, M.C., “Considering Statistical Process Control for Software”, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, September 2001.
- [41] Radice, R., “Statistical Process Control for Software Projects”, 10th Software Engineering Process Group Conference, Chicago, Illinois, March 1998.
- [42] Rozum, J.A., The SEI and NAWC: Working Together to Establish a Software Measurement Program, Technical Report, CMU/SEI-93-TR-07, ESC-TR-93-184, December 1993.
- [43] Sargut, U., Application of Statistical Process Control to Software Development Processes, Master Thesis Dissertation, Informatics Institute of METU, May 2003.
- [44] Sengul, E.S., Intermediate: An Integration Tool for Measurement Data Collection, Master Thesis Dissertation, Informatics Institute of METU, November 2001.
- [45] Shewhart, W.A., Economic Control of Quality of Manufactured Product, Van Nostrand, New York, 1931. (Re-printed by American Society of Quality Control, Milwaukee, Wisc., 1980.)
- [46] Shewhart, W.A., Statistical Method: From the Viewpoint of Quality Control, Lancaster Press Inc., 1939.
- [47] Tarhan, A., An Assessment Model for the applicability of Statistical Process Control for Software Processes, Doctoral Thesis Dissertation, Informatics Institute of METU, August 2006.
- [48] Tarhan, A., Demirors O., Investigating Suitability of Software Process and Metrics for Statistical Process Control, EuroSPI 2006, 2006.
- [49] The Standish Group, The annual CHAOS report 10th edition, 2004.

- [50] Weller, E.F., “Practical Applications of Statistical Process Control”, IEEE Software, May/June 2000, pp.48-55.
- [51] Wheeler, D.J., Advanced Topics in Statistical Process Control, SPC Press, Knoxville, Tenn., 1995.
- [52] Wikipedia, Control Chart, http://en.wikipedia.org/wiki/Control_chart, Last Access Date: 19 February 2007
- [53] Yin, R.K., Case Study Research: Design and Methods, Applied Social Research Methods Series Vol.5, SAGE Publications, 2003.

APPENDICES

A. SPC-AM ASSETS

SPC ^{AM} Process Execution Record (Internal Attributes)			
Process Name:			Recorded On:
Process Execution No:			Recorded By:
1. Inputs: Please list the inputs to the process execution.			
No	Name	Description	
2. Outputs: Please list the outputs from the process execution.			
No	Name	Description	
3. Activities (in sequence): Please list in sequence the activities that were performed while executing the process.			
No	Name	Description	
4. Roles: Please list the roles that were allocated responsibilities in process execution.			
No	Name	Description	
5. Tools and Techniques: Please list the tools and techniques that are used to support process execution.			
No	Name	Description	

Figure A.1 Process Execution Record

Process Name:		Recorded On:	
Process Execution No:		Recorded By:	
External Attributes		Status (Yes/No)	Explanation
PROCESS PERFORMERS			
Q1	Are process performers trained in their roles in the process?		
Q2	Are process performers experienced in their roles in the process?		
Q3	Are process performers differed per role basis during execution of the process?		
PROCESS ENVIRONMENT			
Q4	Has there been a recent change in location?		
Q5	Has there been a recent change in support systems? (infrastructure, technology, etc.)		
Q6	Has there been a recent change in communication channels and mechanisms? (structure, media, etc.)		
Q7	Has there been a recent change in funding and resources allocated for the process?		
Q8	Has the process been tailored for this specific execution?		
OTHER FACTORS (Please list if any)			

Figure A.2 Process Execution Questionnaire

SPC SM Process Attributes Description			
	Process Name:		Described On:
	Process Cluster:		Described By:
1. Inputs: Please list the inputs to the process.			
No	Name	Description	
2. Outputs: Please list the outputs from the process.			
No	Name	Description	
3. Activities (in sequence): Please list in sequence the activities that are performed while executing the process. You can refer to another process description if an activity consists of sub-activities.			
No	Name	Description	
4. Roles: Please list the roles that are allocated responsibilities in the process.			
No	Name	Description	
5. Tools and Techniques: Please list the tools and techniques that are used to support process execution.			
No	Name	Description	

Figure A.4 Process Attributes Description

Metric Name:		Please rate each attribute in four scales, based on answers to questions as indicators:	
Conceptual Definition:		F : Indicators of the attribute are fully satisfied (%86-100)	
Assessed On:		L : Indicators of the attribute are largely satisfied (%61-85)	
Assessed By:		P : Indicators of the attribute are largely satisfied (%16-50)	
		N : Indicators of the attribute are not satisfied (%0-15)	
Attributes	Answers	Rating	Expected Answers
Metric Identity			
Q01	Which entity does the metric measure?	MUF-1	F
Q02	Which attribute of the entity does the metric measure?		
Q03	What is the scale of the metric data? (nominal, ordinal, interval, ratio, absolute)		Ratio, Absolute
Q04	What is the unit of the metric data?		
Q05	What is the type of the metric data? (integer, real, etc.)		
Q06	What is the range of the metric data?		
Data Existence			
Q07	Is metric data existent?	MUF-2	F
Q08	What is the amount of overall observations?		Available > 20
Q09	What is the amount of missing data points?		
Q10	Are data points missing in periods? (If yes, please state observation numbers for missing periods)		
Q11	Is metric data time sequenced? (If no, please state how metric data is sequenced)		
Data Verifiability			
Q12	When is metric data recorded in the process? (at start, middle, end, later, etc.)	MUF-3	F
Q13	Is all metric data recorded at the same place in the process? (at start, middle, end, later, etc.)		Yes
Q14	Who is responsible for recording metric data?		Yes
Q15	Is all metric data recorded by the responsible body?		Yes
Q16	How is metric data recorded? (on a form, report, tool, etc.)		Yes
Q17	Is all metric data recorded the same way? (on a form, report, tool, etc.)		Yes
Q18	Where is metric data stored? (in a file, database, etc.)		Yes
Q19	Is all metric data stored in the same place? (in a file, database, etc.)	MUF-4	F
Data Dependability			
Q20	What is the frequency of generating metric data? (asynchronously, daily, weekly, monthly, etc.)		
Q21	What is the frequency of recording metric data? (asynchronously, daily, weekly, monthly, etc.)		
Q22	What is the frequency of storing metric data? (asynchronously, daily, weekly, monthly, etc.)		
Q23	Are the frequencies for data generation, recording, and storing different?		No
Q24	Is metric data recorded precisely?		Yes
Q25	Is metric data collected for a specific purpose?		Yes
Q26	Is the purpose of metric data collection known by process performers?		Yes
Q27	Is metric data analyzed and reported?		Yes
Q28	Is metric data analysis results communicated to process performers?		Yes
Q29	Is metric data analysis results communicated to management?		Yes
Q30	Is metric data analysis results used as a basis for decision making?		Yes
Data Normalizability			
Q31	Can metric data be normalized by parameters or metrics? (If yes, please specify them)		
Data Integrability			
Q32	Is metric data integrable at project level?		
Q33	Is metric data integrable at organization level?		

Metric Name:		
Conceptual Definition:		
Assessed On:		
Assessed By:		
Metric Usability Attributes	Rating	Expected Rating
Metric Identity (MUA-1)	F	F
Data Existence (MUA-2)	F	F
Data Verifiability (MUA-3)	F	L or F
Data Dependability (MUA-4)	F	L or F
Metric Usability Result	U	L or F (Usable) -- Not Usable otherwise

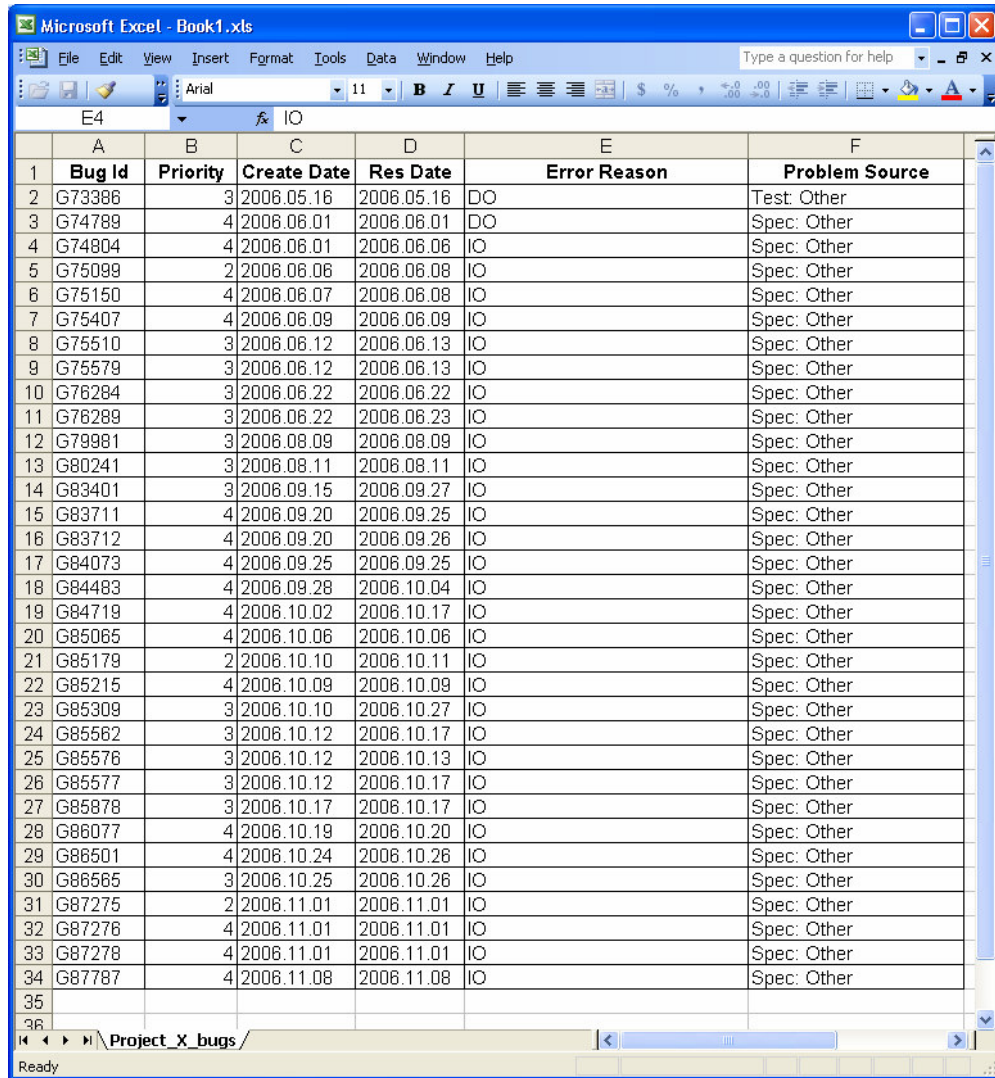
Figure A.5 Metric Usability Questionnaire for Base Metrics

		Please rate each attribute in four scales, based on answers to questions as indicators:	
Metric Name:		F : Indicators of the attribute are fully satisfied (%96-100)	
Conceptual Definition:		L : Indicators of the attribute are largely satisfied (%61-85)	
Assessed On:		P : Indicators of the attribute are largely satisfied (%61-85)	
Assessed By:		N : Indicators of the attribute are not satisfied (%0-15)	
Attributes	Answers	Rating	Expected Answers
Indicators			
Metric Identity		MUF-1	F
Q1	What is the metric formula? (please refer to related base metrics)		
Q2	What is the scale of the metric data? (nominal, ordinal, interval, ratio, absolute)		Ratio, Absolute
Q3	What is the unit of the metric data?		
Q4	What is the type of the metric data? (integer, real, etc.)		
Q5	What is the range of the metric data?		
Data Existence		MUF-2	F
Q6	Is metric data existent?		Available > 10
Q7	What is the amount of overall observations?		
Q8	What is the amount of missing data points?		
Q9	Are data points missing in periods? (If yes, please state observation numbers for missing periods)		
Q10	Is metric data time sequenced? (If no, please state how metric data is sequenced)		
Data Verifiability		MUF-3	F
Q11	How is metric data calculated? (by a tool, manually, etc.)		
Q12	Is all metric data calculated the same way? (by a tool, manually, etc.)		Yes
Q13	Is all metric data calculated according to metric formula?		Yes
Q14	Where is metric data stored? (in a file, database, etc.)		
Q15	Is all metric data stored in the same place? (in a file, database, etc.)		Yes
Data Dependability		MUF-4	F
Q16	Is metric data stored precisely?		Yes
Q17	Is metric data stored for a specific purpose?		Yes
Q18	Is the purpose of metric data storage known by process performers?		Yes
Q19	Is metric data analyzed and reported?		Yes
Q20	Is metric data analysis results communicated to process performers?		Yes
Q21	Is metric data analysis results communicated to management?		Yes
Q22	Is metric data analysis results used as a basis for decision making?		Yes
Data Normalizability			
Q23	Can metric data be normalized by parameters or metrics? (if yes, please specify them)		
Data Integrability			
Q24	Is metric data integrable at project level?		
Q25	Is metric data integrable at organization level?		

Metric Name:		
Conceptual Definition:		
Assessed On:		
Assessed By:		
Metric Usability Attributes	Rating	Expected Rating
Metric Identity (MUF-1)	F	F
Data Existence (MUF-2)	F	F
Data Verifiability (MUF-3)	F	L or F
Data Dependability (MUF-4)	F	L or F
MUF-3&4 for base metric-1		L or F
MUF-3&4 for base metric-2		L or F
MUF-3&4 for base metric-n		L or F
Metric Usability Result	U	L or F (Usable) -- Not Usable otherwise

Figure A.6 Metric Usability Questionnaire for Derived Metrics

B. TOOL INFORMATION



	A	B	C	D	E	F
	Bug Id	Priority	Create Date	Res Date	Error Reason	Problem Source
2	G73386	3	2006.05.16	2006.05.16	DO	Test: Other
3	G74789	4	2006.06.01	2006.06.01	DO	Spec: Other
4	G74804	4	2006.06.01	2006.06.06	IO	Spec: Other
5	G75099	2	2006.06.06	2006.06.08	IO	Spec: Other
6	G75150	4	2006.06.07	2006.06.08	IO	Spec: Other
7	G75407	4	2006.06.09	2006.06.09	IO	Spec: Other
8	G75510	3	2006.06.12	2006.06.13	IO	Spec: Other
9	G75579	3	2006.06.12	2006.06.13	IO	Spec: Other
10	G76284	3	2006.06.22	2006.06.22	IO	Spec: Other
11	G76289	3	2006.06.22	2006.06.23	IO	Spec: Other
12	G79981	3	2006.08.09	2006.08.09	IO	Spec: Other
13	G80241	3	2006.08.11	2006.08.11	IO	Spec: Other
14	G83401	3	2006.09.15	2006.09.27	IO	Spec: Other
15	G83711	4	2006.09.20	2006.09.25	IO	Spec: Other
16	G83712	4	2006.09.20	2006.09.26	IO	Spec: Other
17	G84073	4	2006.09.25	2006.09.25	IO	Spec: Other
18	G84483	4	2006.09.28	2006.10.04	IO	Spec: Other
19	G84719	4	2006.10.02	2006.10.17	IO	Spec: Other
20	G85065	4	2006.10.06	2006.10.06	IO	Spec: Other
21	G85179	2	2006.10.10	2006.10.11	IO	Spec: Other
22	G85215	4	2006.10.09	2006.10.09	IO	Spec: Other
23	G85309	3	2006.10.10	2006.10.27	IO	Spec: Other
24	G85562	3	2006.10.12	2006.10.17	IO	Spec: Other
25	G85576	3	2006.10.12	2006.10.13	IO	Spec: Other
26	G85577	3	2006.10.12	2006.10.17	IO	Spec: Other
27	G85878	3	2006.10.17	2006.10.17	IO	Spec: Other
28	G86077	4	2006.10.19	2006.10.20	IO	Spec: Other
29	G86501	4	2006.10.24	2006.10.26	IO	Spec: Other
30	G86565	3	2006.10.25	2006.10.26	IO	Spec: Other
31	G87275	2	2006.11.01	2006.11.01	IO	Spec: Other
32	G87276	4	2006.11.01	2006.11.01	IO	Spec: Other
33	G87278	4	2006.11.01	2006.11.01	IO	Spec: Other
34	G87787	4	2006.11.08	2006.11.08	IO	Spec: Other
35						
36						

Figure B.1 Excel file example used for importing data

```
serkan_MRs.txt - Notepad
File Edit Format View Help
id;Production_Line;Priority;State;Created_Date;Summary;Production_Date;Requirement_ID;Produced_in_Load
SYM00006932;HP8K Assistant 3.0;2;NAF;2006-08-16;Asst22_ Error when use System Management View;;;
SYM00006933;HP8K Assistant 3.0;3;NAF;2006-08-16;Error on API v150 method createGatewayPABX as Customer User;;API
v150;
SYM00006934;HP8K Assistant 3.0;4;OPU;2006-08-16;Refresh doesn't work properly when the user deletes a
bgl;;07.20.02.00;
SYM00006935;HP8K Assistant 3.0;2;NAF;2006-08-16;Modify Endpoint Error when selecting different EPP;;;
SYM00006936;HP8K Assistant 3.0;3;NAF;2006-08-16;NAQ2171078 - Error message in the Assistant when create MLHG;;;
SYM00006937;HP8K Assistant 3.0;3;NAF;2006-08-16;HiPath v2.2 REG:Problem when select BG FP displayed on page;;;
SYM00006938;HP8K Assistant 3.0;4;AS;2006-08-16;Selected PACs get lost when copy PNP with long name;;DSA_05.13.02.00;
SYM00006939;HP8K Assistant 3.0;2;OPU;2006-08-16;HiPath v2.2 REG:DSA Admin should be able to kill backup;;;
SYM00006940;HP8K Assistant 3.0;3;AS;2006-08-16;Import and delete BG appear time out messages;;DSA_09.01.03.00;
SYM00006941;HP8K Assistant 3.0;3;OPU;2006-08-16;HiPath v2.2 FRN1785:DSA Admin should be able to kill restore;;;
SYM00006942;HP8K Assistant 3.0;4;NAF;2006-08-16;Modify message on FP(busy);;DSA_05.10.03.00;
SYM00006943;HP8K Assistant 3.0;3;FIN;2006-08-16;Wrong Type of Call Forwarding;2006-08-22
00:00:00;DSA_05.18.03.00;HP8K_AST3.0R0.0.0-515
SYM00006944;HP8K Assistant 3.0;3;AS;2006-08-16;Cannot keep Subscriber Templates in DB after update;;DSA_10.08.01.00;
SYM00006945;HP8K Assistant 3.0;3;AS;2006-08-16;Wrong message when a system admin delete his
account;;DSA_03.01.09.00;
SYM00006946;HP8K Assistant 3.0;3;AS;2006-08-16;Modify Endpoint Error;;DSA_05.03.09.00;
SYM00006947;HP8K Assistant 3.0;4;FIN;2006-08-16;EPP Services Error - Simultaneous Ringing;2006-09-01
00:00:00;DSA_01.20.05.00;HP8K_AST3.0R0.0.0-535
SYM00006948;HP8K Assistant 3.0;4;AS;2006-08-16;The underline is missing in the remark field;;DSA_05.10.01.00;
SYM00006949;HP8K Assistant 3.0;4;FIN;2006-08-16;Wrong Endpoint List after Endpoint Deletion;2006-10-17
00:00:00;DSA_05.03.10.00;HP8K_AST3.0R0.0.0-613
SYM00006950;HP8K Assistant 3.0;3;OPU;2006-08-16;HiPath v2.2 FRN1785:DSA Admin should be able to query restore;;;
SYM00006951;HP8K Assistant 3.0;2;FIN;2006-08-16;error connection between dsa & dls;2006-08-30
00:00:00;DSA_07.20.03.00;HP8K_AST3.0R0.0.0-532
SYM00006952;HP8K Assistant 3.0;3;FIN;2006-08-16;table SYMCNAME could not be created.( 459 iso
installat ion);2006-10-25 00:00:00;DSA_06.01.01.00;HP8K_AST3.0R0.0.0-627
SYM00006961;HP8K Assistant 3.0;3;FIN;2006-08-16;Assistant takes a long time to start after rebooting.;2006-09-13
00:00:00;HP8K_AST3.0R0.0.0-554
SYM00006962;HP8K Assistant 3.0;3;FIN;2006-08-16;Global PAC-no BG association;2006-10-24
00:00:00;DSA_05.02.03.00;HP8K_AST3.0R0.0.0-625
SYM00006963;HP8K Assistant 3.0;3;NAF;2006-08-16;Missing Upload Information in UM;;DSA_10.01.02.00;
SYM00006964;HP8K Assistant 3.0;2;FIN;2006-08-16;UM:Cannot create BGL in HP8K Assistant;2006-10-13
00:00:00;DSA_10.03.01.00;HP8K_AST3.0R0.0.0-607
SYM00006965;HP8K Assistant 3.0;3;NAF;2006-08-16;error - modify of the DLS Device Profile of a
subscriber;;DSA_07.20.11.00;
SYM00006966;HP8K Assistant 3.0;4;OPU;2006-08-16;NFR:Screentoscreen transition time should be less than 3
sec;;DSA_01.99.43.00;
SYM00006967;HP8K Assistant 3.0;4;NAF;2006-08-16;Allow DNS host names in Endpoint SIP configuration;;;
SYM00006968;HP8K Assistant 3.0;4;FIN;2006-08-16;refresh doesn't work when add an extension;2006-09-08
00:00:00;DSA_01.14.01.00;HP8K_AST3.0R0.0.0-547
SYM00006969;HP8K Assistant 3.0;3;AS;2006-08-16;The Subscribers are displayed twice in the BG tab;;DSA_03.04.02.00;
SYM00006970;HP8K Assistant 3.0;4;FIN;2006-08-16;Delete Department used by Subscribers:Soap Message;2006-09-06
00:00:00;DSA_05.06.19.00;HP8K_AST3.0R0.0.0-542
SYM00006971;HP8K Assistant 3.0;4;FIN;2006-08-16;Create Department with used Name:Soap Message;2006-09-06
00:00:00;DSA_05.06.19.00;HP8K_AST3.0R0.0.0-542
SYM00006972;HP8K Assistant 3.0;3;FIN;2006-08-16;Edit EPP Error when selecting EP;2006-09-13
00:00:00;DSA_05.13.07.00;HP8K_AST3.0R0.0.0-554
SYM00006973;HP8K Assistant 3.0;3;AS;2006-08-16;Different Confirmation and result windows;;;
SYM00006974;HP8K Assistant 3.0;2;FIN;2006-08-16;DLS Password is visible in address field of the browser;2006-10-18
00:00:00;DSA_07.20.12.00;HP8K_AST3.0R0.0.0-616
```

Figure B.2 CSV file example used for importing data

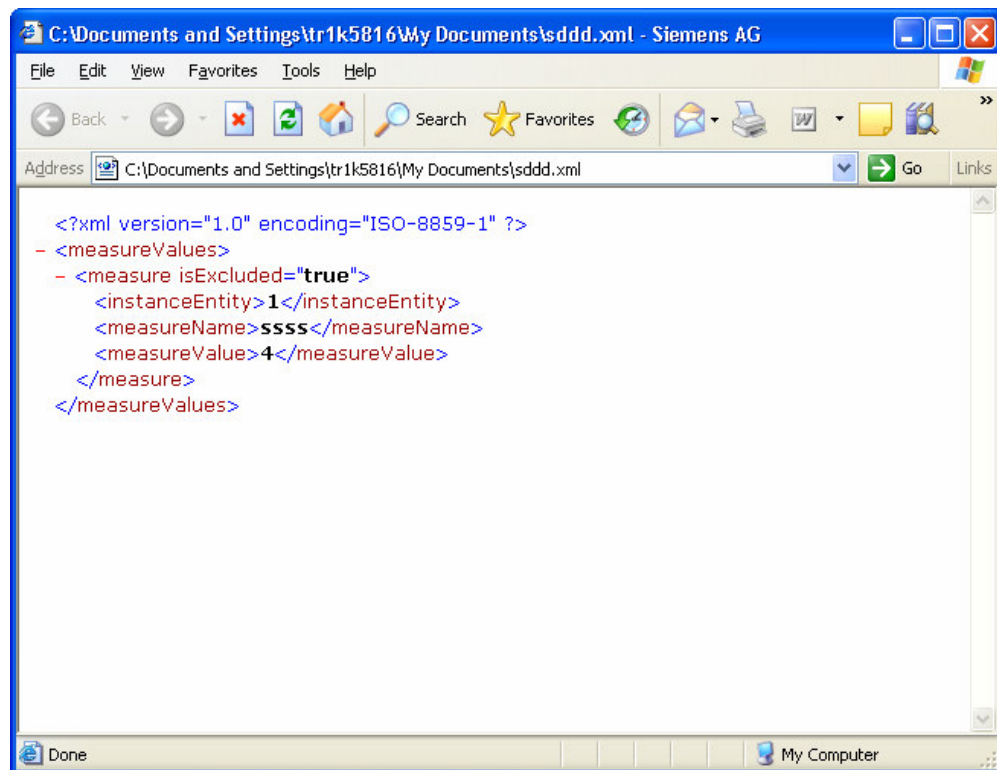


Figure B.3 XML file example used for importing data

All specific requirements:

Table 15 SPC-AAT Requirements

UC No	UC Name	Comment
1	Create a Workspace	Use cases related with <u>Workspace</u> handling
2	Open an existing Workspace	
3	Save the current Workspace	
4	Import Metric Data to Workspace	
5	Update Process Metric Data	Use cases for handling <u>Process Metric Data</u>
6	Display Process Metric Data	
7	Exclude Process Metric Data Points	
8	Create a Process Execution Record	Use cases related with <u>Process Execution Record</u> handling
9	Display an existing Process Execution Record	
10	Update an existing Process Execution Record	
11	Delete an existing Process Execution Record	
12	Display an existing Process Execution Questionnaire	Use cases related with <u>Process Execution Questionnaire</u> handling
13	Update an existing Process Execution Questionnaire	
14	Synchronize Process Similarity Matrix (PSM)	Use cases related with <u>Process Similarity Matrix</u> handling
15	Display Process Similarity Matrix (PSM)	
16	Update Process Similarity Matrix (PSM)	
17	Synchronize Process Executions accr. to PSM	
18	Identify Base Process Clusters from PSM	Use cases related with <u>Base Process Cluster</u> handling
19	Display Base Process Clusters	
20	Report Base Process Clusters	
21	Print Base Process Clusters	
22	Create Process Attributes Description for a Process Cluster	Use cases related with

23	Display a Process Attributes Description	<u>Process Attributes Description</u> handling
24	Create a Process Metric	Use cases related with <u>metrics</u> of the process
25	Display a Process Metric	
26	Update a Process Metric	
27	Delete a Process Metric	
28	Report Metric Definition for a metric	
29	Print Metric Definition for a metric	
30	Display Metric Usability Questionnaire for a metric	Use cases related with <u>Metric Usability Questionnaire</u> handling
31	Update Metric Usability Questionnaire for a metric	
32	Update Metric Usability Rating for a metric	Use cases related with <u>Metric Usability Rating</u> handling
33	Display Metric Usability Rating for a metric	
34	Report Metric Usability Results	Use cases related with <u>Metric Usability Assessment Results</u>
35	Print Metric Usability Results	
36	Merge Process Clusters	Use cases for <u>Process Clusters</u>
37	Split a Process Cluster	
38	Load Base Process Clusters	
39	Show Process Cluster Distances	
40	Draw Control Charts for Process Cluster – Metric pairs	Use cases related with using <u>SPC Tools</u>
41	Draw Bar Charts for Process Cluster – Metric pairs	
42	Draw Histograms for Process Cluster – Metric pairs	
43	Draw Pareto Charts for Process Cluster – Metric pairs	
44	Exclude Metric Data Points on Control Charts	
45	Open Process Execution Questionnaire on Control Charts	
46	Configure Rules for detecting Out-of-Control Points	
47	Display Process Control Status for each metric	Use cases related with <u>Process Control Status</u>
48	Synchronize Process Control Status	

49	Report Process Control Results	
50	Print Process Control Results	
51	Display Out-of-Control Points (OCPs)	Use cases related with <u>Out-of-Control Points (OCPs)</u>
52	Synchronize Out-of-Control Points (OCPs)	
53	Report Out-of-Control Points (OCPs)	
54	Print Out-of-Control Points (OCPs)	
55	Display Information about Tool	<u>Help</u> use cases
56	Display Help Documentation	

Class Diagrams:

ProcessAttributesDescription
- descriptionVersion : String = ""
+ getProcessVersion () + setProcessVersion () + display () + print () + getDescriptionVersion () + setDescriptionVersion () + getIdentifier () + setIdentifier () + toString () + saveToXML () + initializeWithXML ()

ProcessConsistencyMatrix
+ reCalculate () + update () + display () + print () + getProcessConsistencyAssessment () + setProcessConsistencyAssessment ()

ProcessVersion
- numberOfPEs : int = 0
+ display () + print () + getProcessAttributesDescription () + setProcessAttributesDescription () + getProcessConsistencyAssessment () + setProcessConsistencyAssessment () + toString () + getNumberOfPEs () + setNumberOfPEs () + setNumberOfPEs () + getNumberOfPEsAsInteger () + getNumberOfPEsStr () + addMergedProcessVersion () + getMergedProcessVersion () + getMergedProcessVersions () + setMergedProcessVersions () + addOldPERsOfMergedProcessVersions () + getOldPERsOfMergedProcessVersion () + setOldPERsOfMergedProcessVersions () + isSplitSupport () + isEqualTo () + calculateDistanceTo () + saveToXML () + saveHashTableToXML () + initializeWithXML () - createMergedProcessVersionsFromXML () - createOldPERsOfMergedProcessVersionsFromXML () - createHashTableFromXML ()

AProcessAttributes	ProcessAssessment
- processName : String = "" - recordedBy : String = "" + update () + delete () + display () + print () + getIdentifier () + setIdentifier () + getProcessName () + setProcessName () + getRecordedOn () + getRecordedOnStr () + setRecordedOn () + setRecordedOn () + setInputs () + inputsIterator () + addInputs () + removeInputs () + isInputsEmpty () + clearInputs () + containsInputs () + containsAllInputs () + inputsSize () + inputsToArray () + setOutputs () + outputsIterator () + addOutputs () + removeOutputs () + isOutputsEmpty () + clearOutputs () + containsOutputs () + containsAllOutputs () + outputsSize () + outputsToArray () + getRecordedBy () + setRecordedBy () + setActivities () + activitiesIterator () + addActivities () + removeActivities () + isActivitiesEmpty () + clearActivities () + containsActivities () + containsAllActivities () + activitiesSize () + activitiesToArray () + setRoles () + rolesIterator () + addRoles () + removeRoles () + isRolesEmpty () + clearRoles () + containsRoles () + containsAllRoles () + rolesSize () + rolesToArray () + getToolsAndTechniques () + setToolsAndTechniques () + toString () + getActivities () + getInputs () + getOutputs () + getRoles () + saveToXML () - saveProcessAttrVectorToXML () + initializeWithXML () - createProcessAttrVectorFromXML () + stringToInt ()	- processName : String - isRetrospective : boolean + ProcessAssessment () + getProcessName () + setProcessName () + isRetrospective () + setRetrospective () + getProcessConsistencyAssessment () + setProcessConsistencyAssessment () + getPERNosForMetric () + setMetricUsabilityAssessments () + getMetricNames () + doesMetricExist () + getMetricUsabilityAssessments () + getMetricUsabilityAssessment () + getBaseMetricUsabilityAssessments () + getDerivedMetricUsabilityAssessments () + metricUsabilityAssessmentsIterator () + addMetricUsabilityAssessments () + createBulkMetricUsabilityAssessments () + removeMetricUsabilityAssessments () + isMetricUsabilityAssessmentsEmpty () + clearMetricUsabilityAssessments () + containsMetricUsabilityAssessments () + containsAllMetricUsabilityAssessments () + metricUsabilityAssessmentsSize () + metricUsabilityAssessmentsToArray () + createTableModelForMetricUsabilityReporting () + createTableModelForMetricDefinition () + getMetricDataValue () + getMetricDataValueStr () + setMetricDataValue () + setMetricDataValueStr () + isMetricNumeric () + isTypeOfDate () + getDefaultDateFormat () + dateToString () + stringToDate () + findNonNumericMetricObject () + findNonNumericMetricValueStr () + getMetricNamesFromMetricData () + getNumericMetricNames () + getNonNumericMetricNames () + getNonNumericMetricData () + setNonNumericMetricData () + setNonNumericMetricDataValue () + setNonNumericMetricDataValueStr () + getNonNumericMetricDataValue () + getNumericMetricdata () + getNumericMetricdataSize () + setNumericMetricdata () + setNumericMetricDataValue () + setNumericMetricDataValue () + getNumericMetricDataValue () + createMetricDataForDerivedMetric () + drawControlChart () + drawBarChart () + drawHistogram () + getRulesForDetectingOCPs () + setRulesForDetectingOCPs () + saveToXML () + saveMetricDataToXML () + initializeWithXML () - setProcessConsistencyAssessmentFromXML () - setProcessExecutionRecordsFromXML () - setProcessVersionsFromXML () - setMetricUsabilityAssessmentsFromXML () - setMeasureValuesFromXML () - setRulesForDetectingOCPsFromXML () + getNodeValue ()

ProcessConsistencyAssessment
+ ProcessConsistencyAssessment () + synchronizeProcessExecutionRecords () - areProcessAttributesSame () - isPEROfGivenVersion () - calculateDifferenceBetweenProcessAttributes () + calculateClusterDistance () + calculateClusterDistances () - mergeProcessAttributes () + mergeProcessVersions () - replaceProcessVersionWith () - replaceProcessVersionForGivenPERs () - doesPERCorrespondsToExistingVersion () - setVersionOfPER () + splitProcessVersion () + identifyProcessVersionsFromPCM () + findOutBaseProcessVersionsFromPERs () + isProcessVersionUnderControl () + findOCpsFor () + createProcessExecutionRecord () + getProcessAssessment () + setProcessAssessment () + setProcessExecutionRecords () + getProcessExecutionRecords () + getNextPERNo () + getProcessExecutionRecordByExNo () + processExecutionRecordsIterator () + addProcessExecutionRecords () + createBulkProcessExecutionRecords () + createBulkProcessExecutionRecords () + removeProcessExecutionRecords () + isProcessExecutionRecordsEmpty () + clearProcessExecutionRecords () + containsProcessExecutionRecords () + containsAllProcessExecutionRecords () + processExecutionRecordsSize () + processExecutionRecordsToArray () + getProcessConsistencyMatrix () + setProcessConsistencyMatrix () + setProcessVersions () + getBaseProcessVersions () + getSPIProcessVersions () + getProcessVersion () + processVersionsIterator () + addProcessVersions () + removeProcessVersions () + isProcessVersionsEmpty () + clearProcessVersions () + containsProcessVersions () + containsAllProcessVersions () + processVersionsSize () + processVersionsToArray ()

ProcessExecutionRecord
- processExecutionNo : int + ProcessExecutionRecord () + update () + delete () + display () + print () + getProcessExecutionQuestionnaire () + setProcessExecutionQuestionnaire () + getProcessConsistencyAssessment () + setProcessConsistencyAssessment () + getProcessExecutionNo () + getProcessExecutionNoStr () + setProcessExecutionNo () + setProcessExecutionNo () + toString () + getIdentifier () + setIdentifier () + getProcessVersion () + getProcessVersionName () + isProcessVersionOf () + isProcessVersionOf () + setProcessVersion () + isMetricIncluded () + isMetricExcluded () + excludeMetric () + setMetricInclusion () + setMetricDataValue () + setMetricDataValueStr () + getMetricDataValue () + getMetricDataValueStr () + saveToXML () + initializeWithXML () + <u>stringToInt ()</u> + <u>stringToDate ()</u>

	ProcessAttrRowData + NO : String = "No" + NAME : String = "Name" + ACTIVITY_NO : String = "Act_No" + DESCRIPTION : String = "Description" - no : String = "" - name : String = "" - activityNo : String = "" - description : String = "" + ProcessAttrRowData () + ProcessAttrRowData () + ProcessAttrRowData () + getActivityNo () + setActivityNo () + getDescription () + setDescription () + getName () + setName () + getNo () + setNo () + set () + get () + isEqual () + saveToXML () + initializeWithXML ()
MetricUsabilityAssessment + MetricUsabilityAssessment () + getProcessAssessment () + setProcessAssessment () + createMetricUsabilityQuestionnaire () + displayMetricUsabilityJudgments () + getMetricUsabilityQuestionnaire () + setMetricUsabilityQuestionnaire () + isAssessmentOfBaseMetric () + saveToXML () + initializeWithXML ()	

DerivedMetricUsabilityQuestionnaire - metricFormula : String = "" + DerivedMetricUsabilityQuestionnaire () + update () + delete () + display () + print () + decideOnMetricsUsability () + getMetricFormula () + setMetricFormula () + getDependentMetrics () + getDependentMetricNames () + replaceSpacesInFormulaWith_ () + replaceSpacesInMetricNameWith_ () + createTableModelForMetricUsabilityReporting ()
--

<i>MetricUsabilityQuestionnaire</i>
<pre> # questions : String - attributes : String = {"Metric Identity", "Data Existence", "Data Verifiability", "Data Dependability", "Data Normalizability", "Data Integrability"} # emptyList : String # yesNoList : String = {"Yes", "No"} # directIndirectList : String = {"Direct", "Indirect"} # metricScaleList : String = {"Ratio", "Absolute", "Nominal", "Ordinal"} # attrCounts : int # answers : String # choicesForAnswersItems : String # reportItems : String - metricName : String = "" - conceptualDefinition : String = "" - assessedBy : String = "" + getMetricFormula () + setMetricFormula () + createTableModelForMetricUsabilityReporting () + MetricUsabilityQuestionnaire () + update () + delete () + display () + print () + decideOnMetricsUsability () + getMetricUsabilityAssessment () + setMetricUsabilityAssessment () + toString () + getMetricName () + setMetricName () + getConceptualDefinition () + setConceptualDefinition () + getAssessedOn () + getAssessedOnStr () + setAssessedOn () + setAssessedOn () + getAssessedBy () + setAssessedBy () + getAttrCount () + getAttributes () + getAttribute () + getQuestionsCount () + getQuestions () + getQuestion () + getAttributeCounts () + getAnswers () + getAnswer () + setAnswers () + setAnswer () + doesHaveStatus () + getAttrRatings () + getAttrRating () + setAttrRating () + getMUF3_4Rating () + saveToXML () + initializeWithXML () - setUsabilityRatingsFromXML () + getStatusArr () + getStatus () + getStatusStr () + setStatusArr () + setStatus () + getReportItemsCount () + getReportItems () + getReportItem () + getChoicesForAnswersItems () + getCellEditorForAnswersItems () </pre>

A Questionnaire
<ul style="list-style-type: none"> + <i>getAttrCount</i> () + <i>getAttributes</i> () + <i>getQuestionsCount</i> () + <i>getQuestions</i> () + <i>getQuestion</i> () + <i>getAttributeCounts</i> () + <i>doesHaveStatus</i> () + <i>getAnswers</i> () + <i>getAnswer</i> () + <i>setStatusArr</i> () + <i>getStatusArr</i> () + <i>getStatus</i> () + <i>getStatusStr</i> () + <i>setStatus</i> () + <i>setAnswers</i> () + <i>setAnswer</i> () + <i>getChoicesForAnswersItems</i> () + <i>getCellEditorForAnswersItems</i> () + <i>saveToXML</i> () + <i>initializeWithXML</i> () - <i>setAnswersFromXML</i> () - <i>setStatusValuesFromXML</i> ()

UsabilityRating
<ul style="list-style-type: none"> + <u>FULLY SATISFIED : String = "F"</u> + <u>LARGELY SATISFIED : String = "L"</u> + <u>PARTIALLY SATISFIED : String = "P"</u> + <u>NOT SATISFIED : String = "N"</u> - <i>rating</i> : String = "" - <i>expectedRating</i> : String = ""
<ul style="list-style-type: none"> + <i>UsabilityRating</i> () + <i>UsabilityRating</i> () + <i>UsabilityRating</i> () + <i>getRating</i> () + <i>getRatingValue</i> () + <i>setRating</i> () + <i>getExpectedRating</i> () + <i>getExpectedRatingValue</i> () + <i>setExpectedRating</i> () + <i>isRatingOK</i> () + <i>isRatingOK</i> () - <i>createRatingPrioHash</i> () + <i>getValueForRating</i> () + <i>saveToXML</i> () + <i>initializeWithXML</i> ()

BaseMetricUsabilityQuestionnaire

```
+ BaseMetricUsabilityQuestionnaire ( )  
+ update ( )  
+ delete ( )  
+ display ( )  
+ print ( )  
+ getMetricFormula ( )  
+ setMetricFormula ( )  
+ createTableModelForMetricUsabilityReporting ( )
```

ProcessExecutionQuestionnaire

```
- questions : String = {"Are process performers trained in their roles in the process?","Are process performers exp...  
- attributes : String = {"Process Performers","Process Environment"}  
- attrCounts : int = {3,5}  
# emptyList : String  
- answers : String = new String [8]  
- statusArr : boolean = new boolean [8]  
- recordedBy : String = ""  
  
+ getProcessExecutionRecord ( )  
+ setProcessExecutionRecord ( )  
+ getAttrCount ( )  
+ getAttributes ( )  
+ getQuestionsCount ( )  
+ getQuestions ( )  
+ getQuestion ( )  
+ getAttributeCounts ( )  
+ getAnswers ( )  
+ getAnswer ( )  
+ getStatusArr ( )  
+ getStatus ( )  
+ getStatusStr ( )  
+ setAnswers ( )  
+ setAnswer ( )  
+ setStatusArr ( )  
+ setStatus ( )  
+ doesHaveStatus ( )  
+ getRecordedBy ( )  
+ getRecordedOn ( )  
+ getRecordedOnStr ( )  
+ setRecordedBy ( )  
+ setRecordedOn ( )  
+ setRecordedOn ( )  
+ saveToXML ( )  
+ initializeWithXML ( )  
+ getAssignableCauseExp ( )  
+ getChoicesForAnswersItems ( )  
+ getCellEditorForAnswersItems ( )
```

C. DETAILS OF CASE STUDY-A

SPC-AM Assets

XML file for this workspace will not be provided due to space limitations.

D. DETAILS OF CASE STUDY-B

SPC-AM Assets

XML file for this workspace will not be provided due to space limitations.

E. DETAILS OF CASE STUDY-C

SPC-AM Assets

XML file for this workspace will not be provided due to space limitations.

F. SPC-AAT EVALUATION QUESTIONNAIRES

SPC-AAT EVALUATION QUESTIONNAIRE - 1

Date: 29.12.2006

Rating scale: A. excellent / B. good / C. fair / D. poor *Please circle*

1. User Friendliness of SPC-AAT

 (A) (B) (X) (D)

2. How easy is the software to use?

 (A) (B) (X) (D)

4. Is the program easy to install?

 (X) (B) (C) (D)

6. Is installation documentation adequate?

(A) (X) (C) (D)

5. Does the program work as expected, without bugs?

(A) (B) (X) (D)

7. Is help available and easy to use?

(A) (B) (X) (D)

9. Does program meet the stated objectives?

yes no

8. Is utilization of SPC tools adequate at SPC-AAT?

yes no

10. What do you see as SPC-AAT's three main strengths?

a).....is a good method for realizing your own process, shows weaknesses and things that are important in a process and helps in improving the process

b).....it is possible to see the data in different dimensions, and good to add new variables.

c).....provides useful statistics and graphs about the tasks

11. What do you see as SPC-AAT's three main weaknesses?

a).....All tasks are considered as having the same difficulty, and it is not always possible to normalize the values

b).....User needs to change between views. It could be better to reorganize the structure and provide additional info in other views, or link between contents.

c).....the link to the process is missing.

12. What are your suggestions for improving SPC-AAT?

The tool could include the process structure (integrated), and let the user modify the process while working on the tasks. The tool could also indicate problematic paths on the process.

.....
.....

Comments:

.....
.....
.....

Name (optional):

Y. G.....

We thank you for filling in the evaluation sheet and returning it after!

SPC-AAT EVALUATION QUESTIONNAIRE - 2

Date: 08.01.2006

Rating scale: A. excellent / B. good / C. fair / D. poor

Please circle

1. User Friendliness of SPC-AAT
(X) (B) (C) (D)
2. How easy is the software to use?
(A) (X) (C) (D)
4. Is the program easy to install?
(A) (B) (C) (D)
6. Is installation documentation adequate?
(A) (B) (C) (D)
5. Does the program work as expected, without bugs?

(A) (X) (C) (D)

7. Is help available and easy to use?

(A) (X) (C) (D)

13. Does program meet the stated objectives?

yes no

9. Is utilization of SPC tools adequate at SPC-AAT?

yes no

14. What do you see as SPC-AAT's three main strengths?

a).....Reporting functionality is excellent

15. What do you see as SPC-AAT's three main weaknesses?

a).....They are some minor errors but these are easy to correct

16. What are your suggestions for improving SPC-AAT?

Also reporting functionalities can be added for Process Execution Records and Metric Usability Assessments in the system. So that all PERs and MUAs can be seen in one report

.....

.....

Comments:

We investigated the recruitment process and ended with results that showed us we have a huge improvement potential in this area. With this tool we can then measure our performance in recruitment. It can be also employed in many other areas in the company but of course with the support of management.

.....

.....

Name (optional):

H. O. C.

We thank you for filling in the evaluation sheet and returning it after!

SPC-AAT EVALUATION QUESTIONNAIRE - 3

Date: 9 January 2007

Rating scale: *A. excellent / B. good / C. fair / D. poor*

Please circle

1. User Friendliness of SPC-AAT

☒ (A)

(B)

(C)

(D)

2. How easy is the software to use?

(A)

☒ (B)

(C)

(D)

4. Is the program easy to install?

☒ (A)

(B)

(C)

(D)

6. Is installation documentation adequate?

(A)

(B)

☒ (C)

(D)

5. Does the program work as expected, without bugs?

☒ (A)

(B)

(C)

(D)

7. Is help available and easy to use?

(A)

(B)

☒ (C)

(D)

17. Does program meet the stated objectives?

☒ yes

no

10. Is utilization of SPC tools adequate at SPC-AAT?

☒ yes

no

18. What do you see as SPC-AAT's three main strengths?

a) It can be used for any type of process that's applied within a company

b) Graphical User Interface

c) SPC – AAT is a software, which is open for future enhancements and improvements

19. What do you see as SPC-AAT's three main weaknesses?

a) Help

b) Updated user manual (The user manual is not up to date. The pictures should be changed accordingly. With the current document, it is not so easy to go through the software)

c) Error handling (The occurred errors are visible only in the command line. Some standard actions should be implemented that should be performed in case of an error. e.g. refreshing a GUI component, opening a page or displaying an error message)

20. What are your suggestions for improving SPC-AAT?

The user manual should contain “step by step” examples that explain the futures of the software. More sample data should be provided within the installation package.

.....
.....

Comments:

From my point of view SPC – AAT has fulfilled its requirements.

.....
.....
.....
.....

Name (optional):

M. E.

We thank you for filling in the evaluation sheet and returning it after!