IMPROVEMENT OF SOFTWARE SYSTEM TEST PROCESS THROUGH STATISTICAL PROCESS CONTROL

İSTATİSTİKSEL SÜREÇ KONTROLÜ KULLANILARAK YAZILIM SİSTEM TEST SÜRECİNİN İYİLEŞTİRİLMESİ

CANSET G. ALTUN

Başkent Üniversitesi
Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin
İSTATİSTİK ve BİLGİSAYAR Bilimleri Anabilim Dalı İçin Öngördüğü
YÜKSEK LİSANS TEZİ
olarak hazırlanmıştır.

2008

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma, jürimiz tarafından **İSTATİSTİK ve BİLGİSAYAR BİLİMLERİ ANABİLİM DALI 'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan :….....
Prof. Dr. İsmail Erdem

Üye (Danışman) :….....
Dr. Ayça Tarhan

Üye :….....
Yrd. Doç. Dr. Mehtap Akçil Temel

Üye :….....(İmza).....
(Ünvanı, Adı ve Soyadı)

ONAY

Bu tez / /2008 tarihinde, Enstitü Yönetim Kurulunca belirlenen yukarıdaki jüri üyeleri tarafından kabul edilmiştir.

/ /2008
Prof.Dr. Emin AKATA
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

ACKNOWLEDGEMENTS

At this place, I would like to thank the people who have supported me in writing this study.

First of all, I would like to thank my supervisor, Dr. Ayça Tarhan. She contributed a lot to this study, by providing me with many useful suggestions and comments.

I would like to thank my parents, and the rest of my family. They always showed great interest in my work.

Thank you Karbek, Jan, Kansav, and Nefin; your smile just makes my day.

Ankara, September, 2007

Canset ALTUN

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: Canset Altun

Signature: _____

ÖZ

İSTATİSTİKSEL SÜREÇ KONTROLÜ KULLANILARAK YAZILIM SİSTEM TEST SÜRECİNİN İYİLEŞTİRİLMESİ

Canset G. ALTUN
Başkent Üniversitesi Fen Bilimleri Enstitüsü
İstatistik ve Bilgisayar Bilimleri Anabilim Dalı

Yazılım süreçlerinin gelişiminde istatistiksel metodların kullanılması, süreçleri ve onlara ilişkin nicel analizi iyileştirmek için gereklidir. Bu metodların uygulanabilirliği en uygun olan süreçlerden birisi de doğrulama ve geçerleme sürecidir.

Bu çalışmada bir proje kapsamında belirlenen test durumlarına, iki sistem test yöntemi (yol ve düğüm), prospektif (ileriye yönelik) şekilde toplanan veriler üzerinde, yöntemleri karşılaştırmak amacı ile istatistiksel metodlar kullanılarak analiz yapılmıştır. Uygulama sırasında daha önce sekiz çalışmada retrospektif (geriye yönelik) olarak kullanılan SPC-AM yönteminden ve istatistiksel araçlardan yararlanılmıştır.

Bu çalışma ile;

1. Sistem test süreci için belirlenen ölçümlerin yararını anlamak,
2. Kullanılan sistem test yöntemlerinin etkinliğini değerlendirerek daha etkin olan yöntemi belirlemek hedeflenmiştir.

Çalışma sonucunda sistem test süreci kapsamında uygulanan test yöntemleri için belirlenen ölçümlerin verileri kümeleme yöntemi ile gruplanarak değerlendirilmiş ve süreç için etkili olabilecek yöntemin belirlenen kısıtlara göre uygunluğu konusunda öneride bulunulmuştur.

Anahtar Kelimeler: Yazılım ölçümleri, istatistiksel süreç kontrolü, sistem test kapsam analizi
Danışman: Dr. Ayça TARHAN, Hacettepe Üniversitesi, Bilgisayar Mühendisliği Bölümü.

ABSTRACT

IMPROVEMENT OF SOFTWARE SYSTEM TEST PROCESS THROUGH STATISTICAL PROCESS CONTROL

Canset G. ALTUN

Başkent University Institute of Science,

The Department of Statistics and Computer Science

Application of statistical methods on software processes is a required capability to improve processes and their quantitative understanding. Verification and Validation process is one of the most applicable process for these statistical methods.

In this study, two different testing techniques (path and node coverage) are applied on the defined test cases of a project, and statistical methods were implemented prospectively (looking forward) to compare these two techniques on prospectivelly collected test case data. While implementing these statistical methods, an assessment model (SPC-AM) and statistical tools are used which had been previously implemented for eight different processes retrospectively (looking back).

This study aims to:

1. Understand the use of measurements defined for the system test,
2. Identify which test coverage technique would be useful for the validation process by evaluating the effectiveness of two black-box test coverage techniques.

As a result; metric data for test coverage techniques are evaluated by applying process clustering, and suggestions were proposed on the effectiveness of the techniques under related circumstances.

KEY WORDS: Software Metrics, Statistical Process Control, System Test Coverage Analysis.

Supervisor: Dr. Ayça TARHAN, Hacettepe University, Computer Engineering Department.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CMMI	Capability Maturity Model Integrated
GG	Generic Goal
GP	Generic Practice
GQM	Goal-Question-Metric
L2	Maturity Level 2
L3	Maturity Level 3
L4	Maturity Level 4
L5	Maturity Level 5
LCL	Lower Control Limit
MUQ	Metric Usability Questionnaire
OCP	Out-of-Control Point
PEQ	Process Execution Questionnaire
QPM	Quantitative Process Management
SEI	Software Engineering Institute
SG	Specific Goal
SP	Specific Practice
SPC	Statistical Process Control
SPC-AM	Assessment Model for Statistical Process Control
SQM	Software Quality Management
SRS	Software Requirements Specification
SW	Software
UCL	Upper Control Limit
V&V	Verification and Validation
CV	Coefficient of Variation

1 INTRODUCTION

Collecting right metrics and analyzing them in a proper manner provides improving quality and making software processes more efficient while designing and implementing software. Besides the results of the analysis done being a indicator for defining processes correctly or implementing them, it can also be a indicator for the correctness of the methods that are being used for these processes' applications.

Statistical Process Control (SPC) is a statistical based approach that enables us to determine whether a process is stable or not by discriminating between the presence of common cause variation and assignable cause variation. It is a well-established technique, which has shown to be effective in manufacturing processes but not yet in software process contexts [1, 2].

Verification and validation (V&V) process which is one of the most applicable processes for statistical methods is a continuing process throughout the development. Software inspection and software test are the two methods used to verify and validate the software during the development [3].

Software testing has been defined as the process of executing software and comparing the observed behaviour with the desired behavior. The major goal of software testing is to discover errors in the software, with a secondary goal of building confidence in the proper operation of the software when testing does not discover errors [4]. Testing activities have to start at the requirements specification stage, with planning of test strategies and procedures. Data obtained from a real project were analyzed using the framework for validation.

Measurement itself is not a goal, but the goal is to improve the processes. How to measure a test process is a required capability for an effective software testing process. This implies continuous process monitoring in order to predict its behaviour, highlight its performance variations and, if necessary, quickly react to it.

Florac/Carleton explains the different steps, especially in the data collection and behavior description, using statistics in the process measurement [5]. W. Steven Demmy's study shows that SPC techniques can be used to improve the quality and productivity of large-scale software development. He discusses the advantages and

disadvantages of software SPC [6]. Manfred Widera's study shows that even the simplest data flow oriented criterion contains significantly more information than node coverage [7]. In the literature, there are number of articles that discuss the suggestions on implementation of SPC for process improvement in software. These studies indicate that almost all characteristics of processes and products display variation when they are measured.

It is indicated that software process data often represent multiple sources that need to be treated separately, and discovering multiple sources requires the careful investigation of process executions. Clustering is a technique used to analyze or divide a universe of data into homogeneous groups. If the executions of a process show similarity in terms of these attributes, it will be assumed that process executions form a homogeneous subgroup (or "cluster") which consistently performs among its executions; and the process cluster is subject to using SPC techniques.

In this study, two case studies were implemented at a project-based working software organization which had achieved Level 3 in the Software-Capability Maturity Model Integrated (SW-CMMI). The project used here is a large data entry and query system developed on networked, client/server, server utilizing Java and IBM DB2. The project was developed during 6 months with a staff of 5 with approximately 8,000 lines of code.

Two different testing techniques (path and node coverage) were applied on the defined test cases of the project, and statistical methods were implemented to compare these two techniques on prospectivelly collected test case data. While implementing the statistical methods, an assessment model (SPC-AM) which supports process clustering and metric usability evalulation and its tool (SPC-AAT) were used. By this study; it was aimed to understand the use of measurements defined for the system test, and to identify which test coverage technique would be useful for the validation process by evaluating the effectiveness of two black-box test coverage techniques.

The main quantitative tool used in this study was SPC by utilizing control charts. The project analyzed lifecycle data collected during development for testing. Defects were collected during this life-cycle and were quantitatively analyzed using

statistical methods. As a result; metric data for the two test coverage techniques were evaluated and suggestions were proposed on the effectiveness.

1.1 Overview

This chapter gives an overview of this thesis.

Chapter 2 gives basic knowledge on software processes like Validation, CMMI approach, software measurement and SPC. It introduces important terms and concepts that are used in the following chapters.

Chapter 3 provides a survey of the literature on test coverage and SPC implementations for software.

Chapter 4 provides the details related to the assessment model and the assessment process. It describes basic components of the model and explains the assets developed for use in the assessment.

Chapter 5 contains the application part of this study. It gives detailed flow of the case studies.

Chapter 6 discusses results of the implemented test coverage methods which software measures are useful for validation process. In this chapter this study is summarized and the result and experiences from the thesis are discussed.

2 BACKGROUND

2.1 Software Test Process

The software engineering process is a set of sequential practices that are functionally coherent and reusable for software engineering organization implementation and management. It is usually referred to as the software process or simply the process [8].

A software process is structured approach that describes the different activities that will lead to a developed product. Software processes are complex and no two projects are completely the same hence there is not one process that is applicable in all cases. Many organizations use tailoring (modifying process elements and changing the workflow) to develop organization and project specific processes. It is not uncommon for a project to use different processes for different components of a product [9]. There are a number of generic process models, for example the waterfall model, evolutionary development, formal systems development and re-use development [10].

The fundamental activities are the same in all processes: specification, design and implementation, validation. The specification of the software is critical for the further development, because a mistake here will lead to difficulties in the design and implementation. The specification of the software should define its functionality and constraints. This activity is also known as requirements engineering.

The implementation activity is to design and program according to the specification, and it will result in an executable system. If the development process is evolutionary, the specification may also be changed. During the design, the designers decide the structure of the software, the interfaces, the components, and sometimes also the data structures and algorithms. The later part of the design is interleaved with the implementation, and that is why design and implementation is stated as one activity. Some software projects put little effort on design, and instead start to implement almost immediately. This approach is not to recommend, because the lack of structure may create a software that is hard to maintain. There are no general implementation guidelines to follow, but all programmers develop

their own style. The programmers do not only program, but they do also some testing and debugging. Testing is to discover failures, and debugging is to find and correct the place in the code that caused it [9].

Software validation is an activity to make sure that the system meets the specification and the expectations from the end user (Figure 1). After the implementation, different modules of the system work independently, and the next step is to test the modules together. After this test, it is time to test the whole system. The system test includes to validate the functional- and non-functional requirements, and to test the most important properties.

The final step in the validation process is the acceptance test. This means to test the system with data from the end user instead of simulated data. The acceptance test will reveal whether it meets the requirements, and if the performance is acceptable [11].

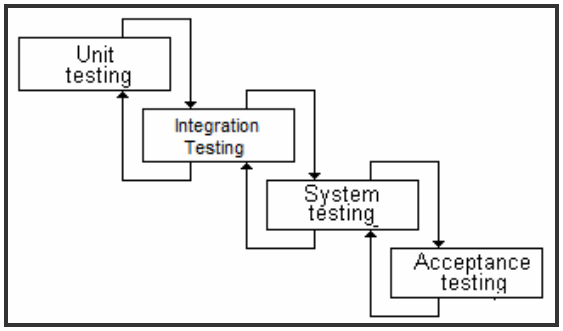


Figure 1 The Software Testing Stages

2.1.1 Verification and Validation (V&V)

Verification and validation are most times used in the same context, but it is important to remember that they have a different meaning given in the following definition [3]:

• “Validation: The right product is being built?”

• “Verification: The product is being built right?”

In other words verification is to make sure that the product meets its specified functional and non-functional requirements. Validation is to make sure that the product is functioning the way that the end user wants. The objective with

verification and validation is not to make the system completely defect free, but to make it good enough for its intended use. V&V process which consists of inspection, review, audit and test subprocesses (Figure 2) is a continuing process throughout the development. Software inspection and software test are the two methods used to verify and validate the software during the development. Software inspection does not require an executable program and can therefore be used throughout the whole development. Software testing does on the other hand require an executable program and can only be used in the later stages. Testing is something that is inevitable in all software development.

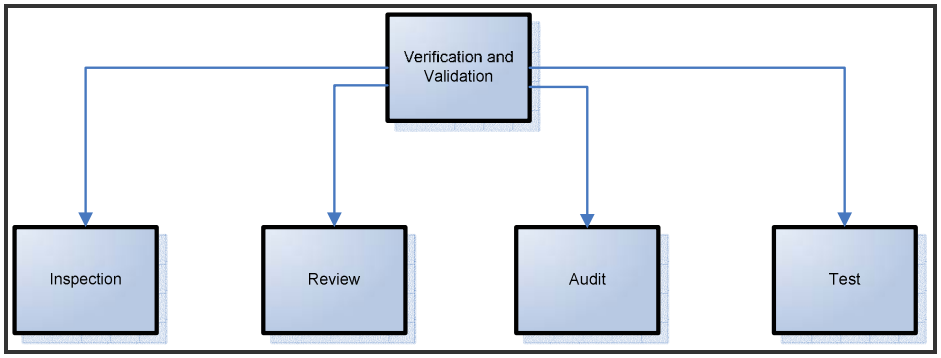


Figure 2 Verification and Validation Process

2.1.2 Testing Methods

In testing there are two different approaches when looking at the code. Static testing is done without executing the code. Instead one goes through the code manually to find faults. Dynamic testing is done by actually executing the code and looking for faults [12].

One method for dynamic testing is black-box testing. Black-box tests the specification without any knowledge of the implementation. This means that the only criterion for success in the testing is if the result is what it should be according to the requirement specifications. The input is chosen very carefully to get the desired result. For each demand a test is designed and the output is compared with the expected one. If there are no discrepancies then the product is considered to be correct.

Various flaws can arise when using this method. There is no way to be sure that all of the code is executed and that all of the cases in the code really is tested. This

means that faults can arise at a later stage when the same demand is tried but under different conditions.

Black-box testing is a very simple approach from the tester's point of view. All they have to do is study the specification and write tests to check that every demand is fulfilled. They can concentrate completely on the functional demands and therefore this approach is also sometimes called functional testing. When discrepancies are found, this method is often much more comfortable for the tester than for the developer. When writing fault reports using this method it is often not really known what caused the fault but rather only that there was a fault. This makes revising more difficult as the developer in a greater extent have to search for the fault in a much wider part of the product, especially if the fault occurs late in the developing process.

Black-box testing is perfect for checking a thorough specification to ensure that the end user's demands are fulfilled. But the method is much better on confirming that the demands in the specification is fulfilled than finding all faults due to the difficulty in deciding on input values. The method is fairly easy for the testers as they do not have to read the developer's code but on the other hand the revising could take longer as it can be difficult to decide where the fault occurred.

Many coverage criteria for software testing such as statement and path coverage, treat each statement as a single node. The testing techniques considered in this study are classified in the literature as black-box, because to generate the test cases for these techniques, a thorough understanding of the source-code of the programs are not needed. The following two test coverage techniques were studied:

- Node Coverage requires the execution of each processing node was executed.

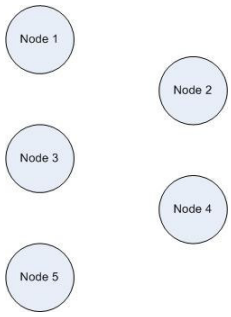


Figure 3 Basic Node Testing Model Representation

Path coverage requires the execution of all possible paths, for instance; branches, statements, and other paths in a program (Figure 3). Faults may not be discovered if the parts containing them have not been executed. The paths should have distinct branches from the start to end of a control flow graph of a program. Thus, essentially, thorough testing is possible through this technique. But, in practice, the number of such paths can be too large in large programs.

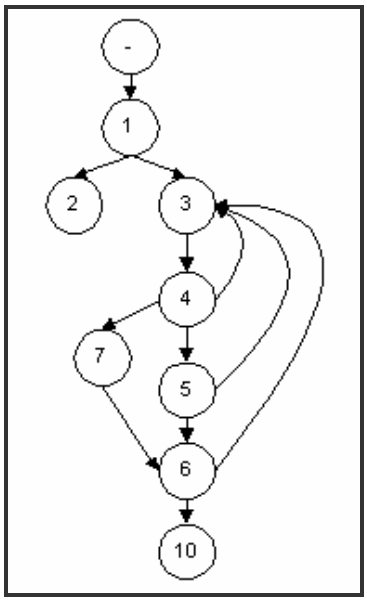


Figure 4 Basic Path Testing Model Representation

Similar to node based models, the path based models consider software architecture with components and interfaces. Initially the different paths in system are obtained either experimentally or algorithmically. Path reliability is the product of all component reliabilities along the path. The system reliability is average of all the path reliabilities. Node based models analytically account for the infinite loops in a path but path based models terminate the loop to one or to an average execution time of the path. Mathur developed a method to combine architecture and failure process by estimating the path reliabilities based on the sequence of components executed for a single test run and the average over all test runs to obtain the system reliability [13].

2.1.2.1 System Testing

System testing is testing that is conducted on the complete, integrated system to evaluate the system's compliance with its requirements. System testing is generally based on black-box testing techniques. In black-box testing the internal workings of the test object are not known and the tester focuses mostly on how the system reacts to different inputs. This is opposed to white-box testing which studies and tests different parts of the system, in detail. System testing tends to be more of an investigatory testing phase, where testers tend to have an almost destructive attitude and not only test the design, but also the behaviour and the believed expectations of the end user. System testing is intended to test up to and beyond the software and hardware requirements specifications. As software faults are found during system testing new software builds are released that include corrections of detected faults. The incremental nature of system testing is controlled by defining regression tests.

2.2 Software Process Management

Software process management is about successfully managing the work processes associated with developing, maintaining, and supporting software products and software intensive systems [11]. Successful management is that the products and services produced meet the business objectives of the organization responsible for producing the products. The concept of process management is found on the principles of statistical process control. These principles hold that by establishing and sustaining stable levels of variability, processes will yield predictable results. We can then say that the processes are under control statistically.

Predictable results should not be interpreted to mean identical results. Results always vary; but when a process is under statistical control, they will vary within predictable limits. If the results of a process vary unexpectedly—whether randomly or systematically—the process is not under control, and some of the observed results will have assignable causes. These causes must be identified and corrected before stability and predictability can be achieved. Controlled processes are stable processes, and stable processes enable us to predict the results. This in turn enables us to prepare achievable plans, meet cost estimates and scheduling

commitments, and deliver required product functionality and quality with acceptable and reasonable consistency. If a controlled process is not capable of meeting end user requirements or other business objectives, the process must be improved or retargeted (Figure 4).

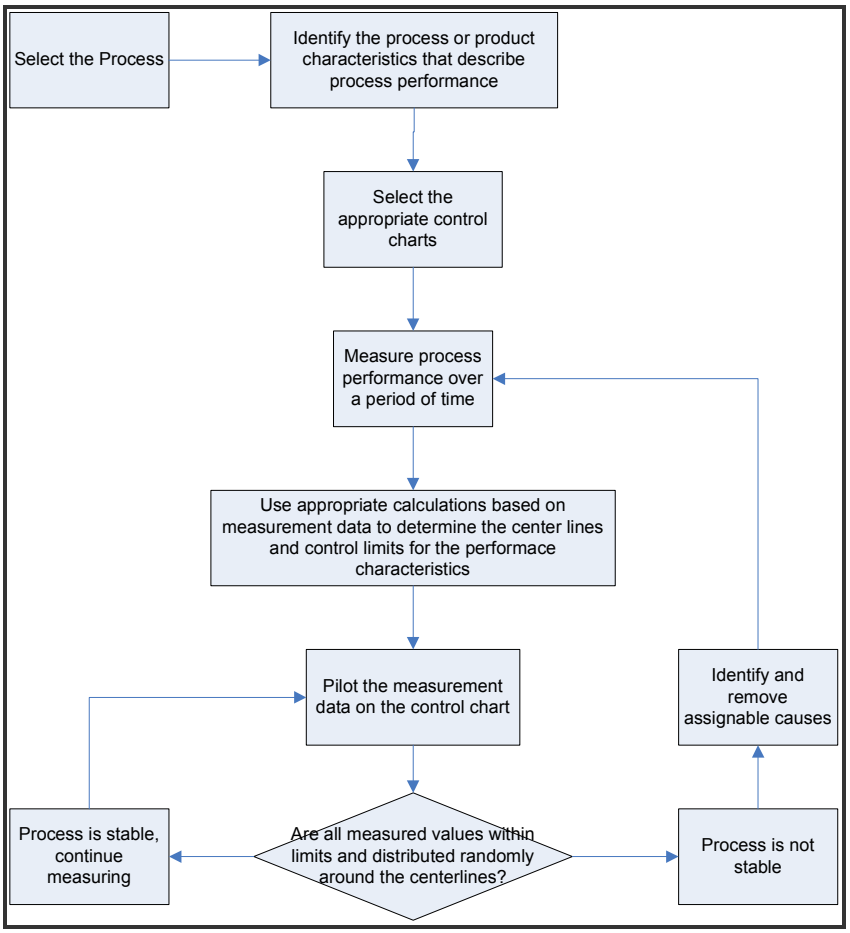


Figure 5 Steps for Using Control Charts to Evaluate Process Stability [14]

At the individual level then, the objective of software process management is to ensure that the processes you operate or supervise are predictable, meet end user needs, and (where appropriate) are continually being improved. From the larger, organizational perspective, the objective of process management is to ensure that the same holds true for every process within the organization.

There are four key responsibilities of software process management which are *define the process, measure the process, control the process, improve the process*. The flow between these processes are shown in Figure 5 [11, 14].

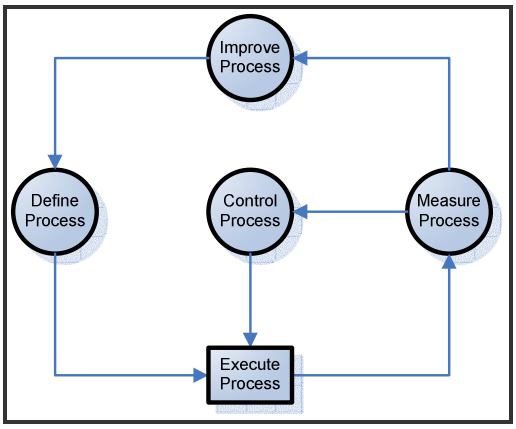


Figure 6 The Four Key Responsibilities of Process Management

2.2.1 The CMMI Approach

CMMI stands for Capability Maturity Model Integration [16] and it is a process improvement approach that provides organizations with the essential elements of effective processes. It can be used to guide process improvement across a project, a division, or an entire organization. CMMI helps integrate traditionally separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes.

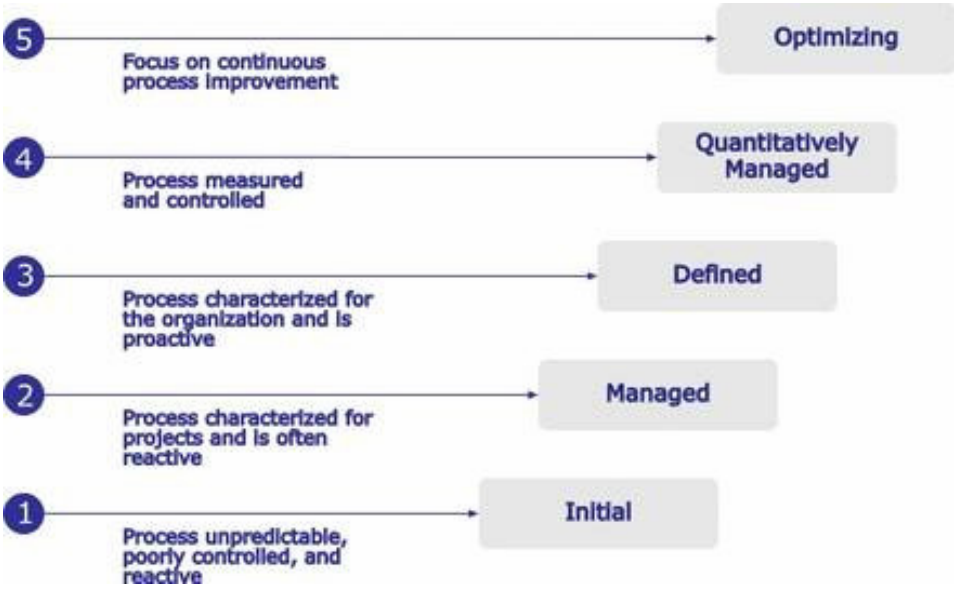


Figure 7 Capability Maturity Model Integration (CMMI)

The CMMI is a model that needs to be interpreted based upon the business environment and technical needs of the project; it is not a standard that must be implemented exactly as documented.

The CMMI is structured in the five maturity levels (Figure 7), the considered process areas, the specific goals (SG) and generic goals (GG), the common features and the specific practices (SP) and generic practices (GP) are given in Figure 8. The process areas are defined as follows:

“The Process Area is a group of practices or activities performed collectively to achieve a specific objective.”

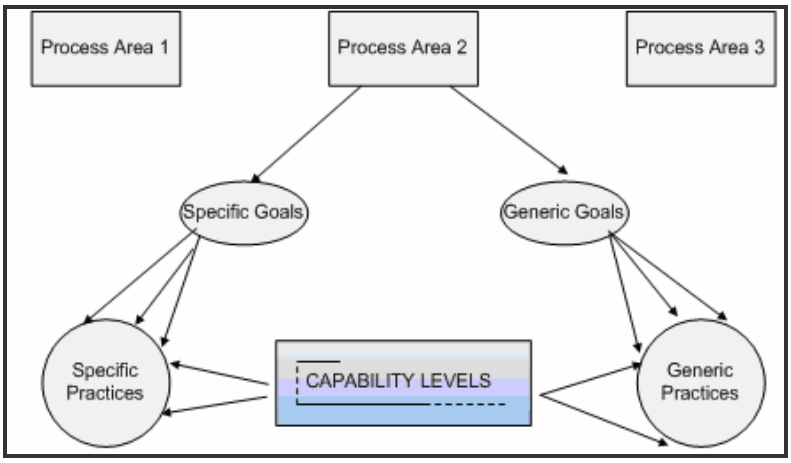


Figure 8 The CMMI model components

Such objectives could be the part of requirements management at the level 2, the requirements development at the maturity level 3 or the quantitative project management at the level 4.

CMMI based process improvement benefits include;

- Improved schedule and budget predictability
- Improved cycle time
- Increased productivity
- Improved quality (as measured by defects)

- Increased end user satisfaction
- Improved employee moral
- Increased return on investment
- Decreased cost of quality

2.2.1.1 CMMI Process Maturity Levels

Initial (Level 1): The initial environment has ill-defined procedures and controls.The organization does not consistently apply software engineering management to the process, nor does it use modern tools and technology. Level 1 organizations may have serious cost and schedule problems.

Repeatable (Level 2): At L2, the organization has generally learned to manage costs and schedules, and the process is now repeatable. The organization uses standard methods and practices for managing software development activities such as cost estimating, scheduling, requirements changes, code changes, and status reviews.

Defined (Level 3): At L3, the process is well-characterized and reasonably well understood. The organization defines its process in terms of software engineering standards and methods, and it has made a series of organizational and methodological improvements.These specifically include design and code reviews, training programs for programmers and review leaders, and increased organizational focus on software engineering. A major improvement in this phase is the establishment and staffing of a software engineering process group that focuses on the software engineering process and the adequacy with which it is implemented.

Managed (Level 4): At L4, the process is not only understood but it is quantified, measured, and reasonably well controlled. The organization typically bases its operating decisions on quantitative process data and conducts extensive analyses of the data gathered during software engineering reviews and tests. Tools are used increasingly to control and manage the design process as well as to support data

gathering and analysis. The organization is learning to project expected errors with reasonable accuracy.

Optimized (Level 5): At L5, the organization has not only achieved a high degree of control over its process, it has a major focus on improving and optimizing its operation. This includes more sophisticated analyses of the error and cost data gathered during the process as well as the introduction of comprehensive error cause analysis and prevention studies. The data on the process are used iteratively to improve the process and achieve optimum performance.

The Software Engineering Institute's Software Capability Maturity Model (SW-CMMI) L4 quantitative analysis leads to SW-CMMI L5 activities. L4 Software Quality Management (SQM) key process area analysis, which focuses on product quality, feeds the activities required to comply with defect prevention (DP) at L5 [1]. Quantitative Process Management (QPM) at L4 focuses on the process that leads to technology change management and process change management at L5. At L3, metrics are collected, analyzed, and used to status development and to make corrections to development efforts, as necessary. At L4, measurements are quantitatively analyzed to control process performance of the project and to develop a quantitative understanding of the quality of products to achieve specific quality goals. This study presents the application of statistical process control (SPC) to accomplish the SQM and QPM and apply these results to DP. Real project results are used to demonstrate the use of SPC as applied to software development. An overview of control charts is presented along with L4 quality goals and plans to meet these goals.

An organization performing L4 quantitative analysis recognizes that it leads to L5 activities. This study presents this progressive relationship in project examples where statistical process control (SPC) is used to analyze measurements. Results of this analysis are used to gain a quantitative understanding of process capability, manage progress toward achieving quality goals, and for defect prevention.

Rigorous statistics have been used in manufacturing but have had limited use in software development. The SEI's Capability Maturity Model IntegratedSM (CMMI) calls for rigorous statistics at L4 and emphasizes SPC. This study shows that

control charts and other statistical methods can easily and effectively be applied in a software setting [17].

2.3 Software Measurement

Measurement in software engineering is called software metrics, or more precise software metrics are any type of measurement that relates to a software system, process or its documentation. Software measurement is the objective quantification of attributes of software entities: processes, products and resources [18]. Software measurement is needed to gain control over excessive cost of software, low productivity, and poor quality.

Measurement is a mean to acquire quantitative information of software processes and products for the purpose of managing them. Measurement can be used to define the status of processes or product quality, to analyze the effects of changes, or o follow-up the progression of improvement actions. The main reason for measuring a software project is to get information about it and the organization, and be able to control the projects better. Software measurement can help to keep the people informed about their concerns, but it does not claim to give any absolute solutions.

Analysis and interpretation of measurement data must be done within the context of other information about the process or product. Measurement data by themselves are neither bad news nor good news. A report indicating zero defects in the two months following product release may be very good news (if the product is being used by a large number of end users) or very bad news (if there are few to zero end users using the product). Measurement results must be examined in the context of other information about the product or process to determine whether action is required and what action to take. Unexpected measurement results generally require additional information to properly assess the meaning of the measurement [11].

In order to understand what must be measured, organizational goals must be understood. If one of the organizational goals is to improve product quality, then the test process document must define metrics that allow evaluating improvements in

software quality. Test Metric is a standard means of measuring some attribute of the software testing process. . They are a means of establishing test progress against the test schedule and may be an indicator of expected future results. Pusala introduces test metrics in two forms, Base Metrics and Derived Metrics, as listed below [12].

• Example of Base Metrics:
Test Cases
New Test Cases
Test Cases Executed
Test Cases Unexecuted
Test Cases Re-executed
Passes
Fails
Test Cases Under Investigation
Test Cases Blocked
1st Run Fails
Test Case Execution Time
Testers

• Example of Derived Metrics:
% Test Cases Complete
% Test Cases Passed
% Test Cases Failed
% Test Cases Blocked
% Test Defects Corrected

2.3.1 Software Process Measurement

Controlling a process means making it behave the way we want it to. This provides two things for organization: predict results and produce products that have characteristics required by the end users. With control, we can commit to dates when products will be delivered and live up to such commitments.

There are five perspectives that are central to process measurement [11]:

• Performance

• Stability

• Compliance

• Capability

• Improvement and investment

2.3.2 Why Measure?

There are four reasons for measuring software processes, products, and resources [11]:

• To characterize

They are characterized to gain understanding of processes, products, resources, and environments, and to establish baselines for comparisons with future assessments.

• To evaluate

They are evaluated to determine status with respect to plans. Measures are the sensors that let us know when our projects and processes are drifting off track, so that we can bring them back under control. We also evaluate to assess achievement of quality goals and to assess the impacts of technology and process improvements on products and processes.

• To predict

They are predicted so that we can plan. Measuring for prediction involves gaining understandings of relationships among processes and products and building models of these relationships, so that the values we observe for some attributes can be used to predict others. We do this because we want to establish achievable goals for cost, schedule, and quality—so that appropriate resources can be applied. Predictive measures are also the basis for extrapolating trends, so estimates for cost, time, and quality can be updated based on current evidence. Projections and

estimates based on historical data also help us analyze risks and make design/cost tradeoffs.

• To improve

They are measured to improve when we gather quantitative information to help us identify roadblocks, root causes, inefficiencies, and other opportunities for improving product quality and process performance. Measures also help us plan and track improvement efforts. Measures of current performance give us baselines to compare against, so that we can judge whether or not our improvement actions are working as intended and what the side effects may be. Good measures also help us communicate goals and convey reasons for improving. This helps engage and focus the support of those who work within our processes to make them successful.

2.3.3 Measurement Scales And Scale Types

Measurement Scales [20];

Ratio: Numeric data with equal distances corresponding to equal quantities of the attribute.

Interval: Numeric data with equal distances corresponding to equal quantities of the attribute.

Ordinal: Observations result in assigning discrete rankings.

Nominal: Observations result in assigning a category or class.

Scale Types;

Discrete or event (attribute):

- Counted and plotted as discrete values
- Possible values are finite over any given interval

Continuous (variable):

- Measured and plotted on a continuous scale

- Can assume all values between any two given values
- Effectively, infinite number of values is possible

Large (discrete) counts may be treated as continuous for many purposes.

2.3.4 Why do we need metrics?

A major percentage of software projects suffer from quality problems. Software testing provides visibility into product and process quality. Test metrics are key "facts" that project managers can use to understand their current position and to prioritize their activities to reduce the risk of schedule over-runs on software releases.

Test metrics help us to measure our current performance. Because today's data becomes tomorrow's historical data, it is ever too late to start recording key information on your project. This data can be used to improve future work estimates and quality levels. Without historical data estimates will just be guesses.

The benefits of having good metrics;

- Test metrics data collection helps predict the long term direction and scope for an organization and enables a more holistic view of business and identifies high-level goals.
- Provides a basis for estimation and facilitates planning for closure of the performance gap.
- Provides a means for control/status reporting.
- Identifies risk areas that require more testing.
- Quickly identifies and helps resolve potential problems and identifies areas of improvement.
- Test metrics provide an objective measure of the effectiveness and efficiency of testing.

2.3.5 The Goal/Question/Metric Method (GQM)

The GQM method represents a systematic top-down approach to defining and collecting measurements, and on the other hand, a bottom-top approach when analyzing data against stated measurement goals. One of the method's main aims to establish a visible link from measurement goals to the data collected. The underlying idea is to avoid the high risk of wasting resources when measurement data is collected without an idea of its usage. GQM adapts and integrates organizational objactives into measurement goals, and refines them into measureable attributes on a step-by-step basis; therefore, GQM helps to identify the exact metrics necessary for meeting case-specific objectives.

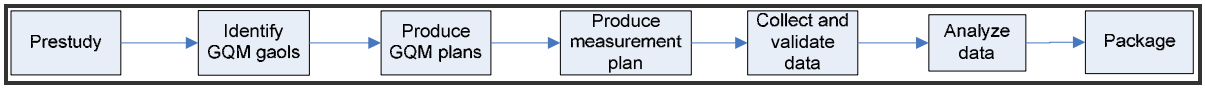


Figure 9 The activities of a GQM measurement programme [9]

A GQM model is a hierarchical structure as shown in Figure 9. It starts with a goal specifying purpose of the measurement, object to be measured, issue to be measured, and viewpoint from which the measure is taken. Objects of measurement include products, processes, and resources. The goal is refined into several questions that usually break down the issue into its major components. Questions try to characterize the object of measurement (product, process, or resource) with respect to a selected quality issue, and to determine its quality from the selected viewpoint. Each question is then refined into metrics, either objective or subjective. Objective metrics include the data that depend only on the object that is being measured and not on the viewpoint from which they are taken. Subjective metrics depend on both the object that is being measured and the viewpoint from which they are taken. The same metric can be used to answer different questions under the same goal. Several GQM models can have questions and metrics in common.

The goal-driven measurement process is based on 3 precepts, and it consists of 10 steps [20, 21].

The three precepts are;

- Measurement goals are derived from business goals.
- Evolving mental models provide context and focus.
- GQ(I)M translates informal goals into executable measurement structures.

The 10 steps are;

1. Identify your business goals.
2. Identify what you want to know or learn.
3. Identify your subgoals.
4. Identify the entities and attributes related to your subgoals.
5. Formalize your measurement goals.
6. Identify quantifiable questions and the related indicators that you will use to help you achieve your measurement goals.
7. Identify the data elements that you will collect to construct the indicators that help answer your questions.
8. Define the measures to be used, and make these definitions operational.
9. Identify the actions that you will take to implement the measures.
10. Prepare a plan for implementing the measures.

GQM is currently the best approach and it has been successfully used in many software organizations. But due to its shortcomings researches have proposed a number of improved GQM approaches. One of them is V-GQM that is described below. Olsson and Runeson [20] present an extended GQM, which they call V-GQM (Validation Goal Question Metric). The purpose of the V-GQM is to take unforeseen benefits of the metrics into account and to improve subsequent GQM studies. When the original GQM stops after the analysis of the gathered data, V-

GQM has three additional steps, which are metric validation, question analysis, and goal refinement as indicated in Figure 9.

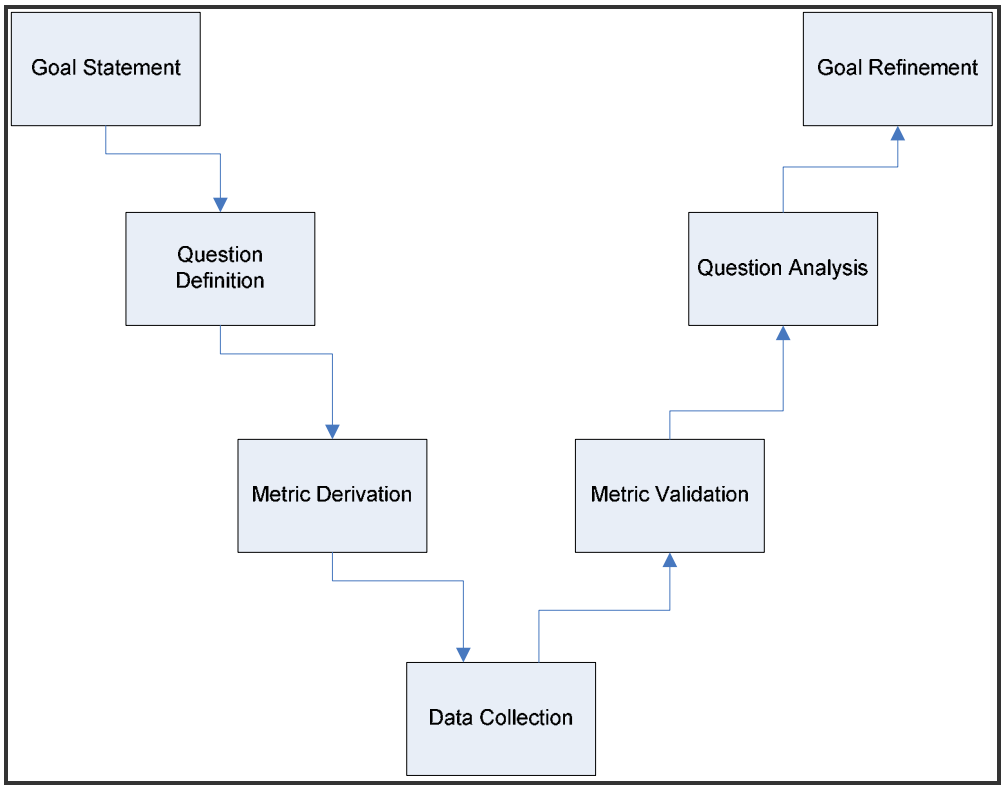


Figure 10 The V-GQM Model

First Step: Goal Definition

Analyze	The system test process
For the purpose of	improving
With respect to	efficiency
From the viewpoint of	the system tester
In the context of	product XXXXX

Second Step: Defining Questions

Q1. Which is the most effective test technique?

Third Step: Identify Metrics

M1. Test Effectiveness

By creating goals, questions and linking them to metrics, data extraction will be made in a more structured way: each metric will have a clearly defined purpose and a traceable dependency to the defined goals. This facilitates making analyses on the collected data and helps drawing conclusions on improvement suggestions.

2.4 SPC

Statistical process control (SPC) involves using statistical techniques to measure and analyze the variation in processes [22]. The intent of SPC is to monitor product quality and maintain processes to fixed targets. Statistical quality control refers to using statistical techniques for measuring and improving the quality of processes and includes SPC in addition to other techniques, such as sampling plans, experimental design, variance reduction, process capability analysis, and process improvement plans.

SPC is used to monitor the consistency of processes used to generate a product as designed. It aims to get and keep processes under control. No matter how good or bad the design, SPC can ensure that the product is being generated as designed and intended. Thus, SPC will not improve a poorly designed product's reliability, but can be used to maintain the consistency of how the product is made and, therefore, of the generated product itself and its as-designed reliability.

A primary tool used for SPC is the control chart, a graphical representation of certain descriptive statistics for specific quantitative measurements of the processes. These descriptive statistics are displayed in the control chart in comparison to their "in-control" sampling distributions. The comparison detects any unusual variation in the process, which could indicate a problem with the process. Several different descriptive statistics can be used in control charts and there are several different types of control charts that can test for different causes, such as how quickly major vs. minor shifts in process means are detected. Control charts are also used with product measurements to analyze process capability and for continuous process improvement efforts.

There is an increased interest in using control charts for monitoring and improving software processes, particularly quality control processes like reviews and testing. In a control chart, control limits are established for some attributes and, if any point falls outside the limits, it is assumed to be due to some special causes that need to be identified and eliminated. If the control limits are too tight, they may raise too many false alarms and, if they are too wide, they may miss some special situations [22].

Control Chart (Figure 11): Control charts are simple 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.

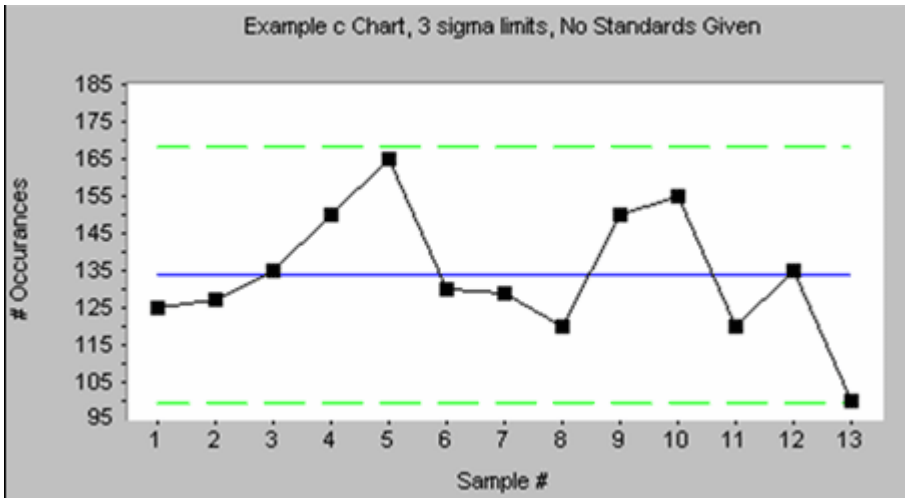


Figure 11 Control Chart Example

The application of SPC by Florac and Carleton [21] is based on the following general characterization of software process management:

Define the process as,

- Design processes that can meet or support business and technical objectives
- Identify and define the issues, models, and measures that relate to the performance of the processes

Measuring the process as,

- Collect data that measure the performance of each process
- Analyze the performance of each process
- Retain and use the data as follows: to assess process stability and capability, to interpret the results of observations and analyses, to predict future costs and performance, to provide baselines an benchmarks, to plot trends, to identify opportunities for improvement

Controlling the process as,

- Determine whether or not the process is under control (is stable with respect to the inherent variability of measured performance)
- Identify performance variations that are caused by process anomalies (assignable causes)
- Eliminate the sources of assignable causes so as to stabilize the process

Improve the process as,

- Understand the characteristics of existing processes and the factors that affect process capability
- Plan, justify, and implement actions that modify the processes so as to better meet business needs
- Assess the impacts and benefits gained, and compare these to the costs of changes made to the processes

The Florac/Carleton approach [24] is addressed to the beginning of process measurement and explains the different steps using statistics in the process measurement, data collection and behaviour description especially.

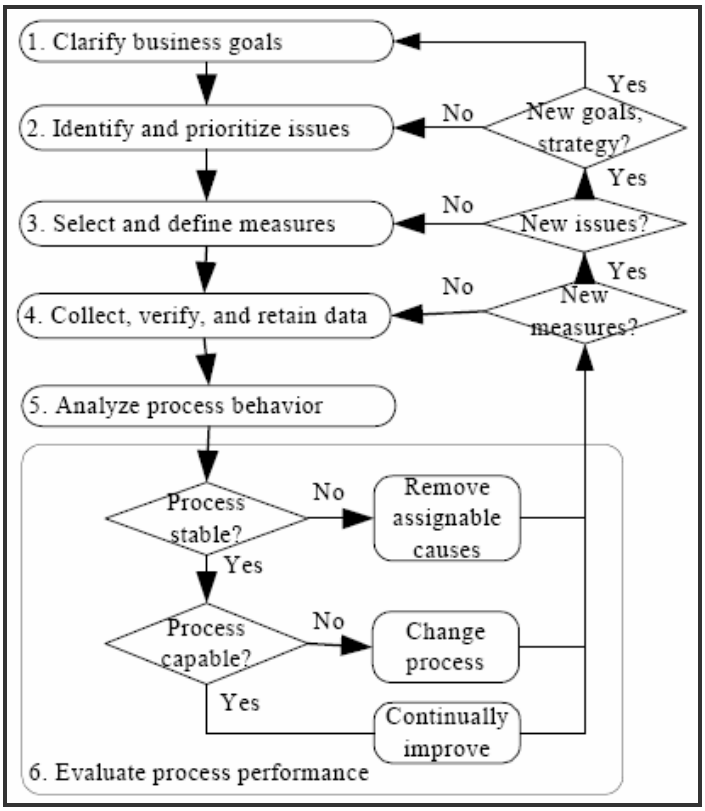


Figure 12 Florac/Carleton Approach for Process Measurement [24]

There are several methods for performing SPC: Scatter diagrams, run charts, cause and effect diagrams, histograms, bar charts, pareto charts, and control charts. Although all of these methods are useful, we will focus this study on control charts.

SPC control charts, if successfully applied, can be a significant impetus for software process improvement. By knowing our normal process, we can reengineer it to obtain improvement in some performance aspect. And, by identifying anomalous behavior, we can seek the special cause (an influence from outside the system) and take action to prevent it from affecting future performance.

The fundamental idea of process improvement is that as the system is observed over time, the process decreases its variation and, increasingly, gets closer to achieving its planned performance objective because of the introduction of improvements. SPC control charts facilitate this process improvement concept. Thus, you have the reason why the recently issued Software CMM Integration (CMMI) has specifically used the words "statistically manage" in its CMMI L4 Process Area, "Quantitative Project Management" [17].

There are seven SPC control chart types, each having a specific application. The control chart required for our application is termed "Individuals and Moving Range." Symbolically, it is shown as XmR, where X represents the individual observations, and mR represents the moving range, the difference between successive observations. The XmR control chart is used when there is only one measurement of the variable in an observation period.

For all types of control charts, the control limits establish filtering. The high limit is plus three sigma from the average of the observations, whereas the low limit is the average minus three sigma. Sigma is a standard statistical measure of the variation in the process [25].

3 LITERATURE REVIEW

Coverage Measurement Experience During Function Test [26];

Piwowarski, Ohba, Caruso discussed that measurement of statement and branch coverage of large system software can be done, and is cost effective in removing errors and if a good test coverage measurement tool is available, an exit criteria of unit test can be 100% statement coverage.

Improving State-Based Coverage Criteria Using Data Flow Information [27];

Briand, Labiche, Lin show that data flow information can be used to select the best transition tree when more than one satisfies the transition tree criterion. They further propose a more optimal strategy for the transition tree criterion, in terms of cost and effectiveness. The improved tree strategy is evaluated through the two case studies and the results suggest that it is a cost-effective strategy that would fit into many practical situations.

Measurement Issues and Software Testing [28];

Cem Kaner worked on measurement issues to identify the methods used in software testing.

Data Flow Coverage for Testing Erlang Programs [7];

Manfred Widera's study concludes that while the proposed data flow oriented coverage criteria are more complex to check than simple node coverage (especially they rely on the computation of a flow graph), measurements show that even the simplest data flow oriented criterion contains significantly more information than node coverage.

Statistical Process Control: Measuring the Software Process – Statistical Process Conrol for Software Process Improvement [24];

The Florac/Carleton approach is addressed to the beginning of process measurement and explains the different steps using statistics in the process measurement, data collection and behaviour description especially.

Define the processes; design processes that can meet or support business and technical objectives and identify the issues, models, and measures that relate to the performance of the processes.

Measure the proceses; collect data that measure the performance of each process and analyze the performance of each process.

Control the processes; determine whether or not the process is under control (is stable with respect to the inherent variability of measured performance) and identify performance variations that are caused by process anomalies (assignable causes). Control the processes by eliminating the sources of assignable causes.

Improve the processes; understand the characteristics of existing processes and the factors that affect process capability. Plan, justify, and implement actions that modify the processes so as to better meet business needs. Assess the impacts and benefits gained, and compare these to the costs of changes made to the processes.

Statistical Process Control in Software Quality Assurance

W. Steven Demmy's study [6] shows may SPC techniques be used to improve the quality and productivity of large-scale software development. He concludes with the advantages and disadvantages of Software SPC. Process monitoring has two major advantages compared to the detailed inspection of completed software units. First, errors may be detected earlier or prevented altogether. Second, less effort may be required to Successful applications insure that processes are operating discipline. They require correctly than is required to perform detailed checks on all the outputs of that process. Thus, higher quality may be achieved at a lower development expense. Despite the advantages listed above, there are several potential disadvantages. Successful applications require an organizational climate that rewards the detection and correction of problems. Once formal process monitoring has been implemented, failures in discipline, in planning, or in commitment will be quickly visible. If the organizational climate views problem detection as a means of assigning blame, rather than of solving problems, attempts to support of the system will be replaced by attempts at system subversion.

The Florac/Carleton approach [24] is addressed to the beginning of process measurement and explains the different steps using statistics in the process measurement, data collection and behaviour description especially.

Niessink and Vliet [29] worked on measurement-based improvement which is that measurement itself is not a goal, but the goal is organisational, or to solve an organisational problem. It is assumed that the measurement activities are performed in combination with improvement activities to reach the goal. The process starts at the leftmost dot with an organisational problem or a goal. The organisation analysis the problem and arrive in the middle, with either a solution or a cause to the problem. If they have enough information to solve the problem they implement it and arrive at the goal (leftmost dot). If they have not enough information they need to implement a measurement program or design an experiment (right dot). Analysing the gathered information takes the organisation back to the middle with a solution. They then implement the solution and arrive at the goal (left dot). This model is very simplified and it might be that the organisation has to loop the right part many times to find a solution.

4 AN ASSESMENT MODEL FOR STATISTICAL PROCESS CONTROL

The assessment approach includes an assessment process that guides the evaluation, an assessment model that defines assets to evaluate a process and metrics, and an assessment tool that supports this evaluation [30].

The assessment model aims to test the suitability of a software process and metrics for quantitative analyses. It investigates two basic requirements for quantitative implementation: Stratification of process executions and data, and metric and data utilization for statistical analyses [30, 31].

The assesment model was previously utilized on eight case studies in several industrial contexts. The assesments were performed retrospectively on past process executions and data in all case studies. The assessments were performed by individuals who are software experts. Process performers were the basic information source while trying to capture contextual information of past process executions.

4.1 Model Components

The first requirement is the stratification of process executions and data. The purpose of stratification is to obtain and use data that are representative of the performance of the process with respect to the issues being studied. If it can be considered that observations are made under essentially the same conditions and that differences between the measurements are primarily due to common cause variation, then the observations are very likely grouped rationally.

Since the sampled process executions as being from a single and constant system of chance causes, a clustering method was developed based on process attributes such as inputs, outputs, activities, roles, and tools and techniques. The relation of these attributes to the process is given in Figure 13. If repetitions of a process show similarity in terms of these attributes, then it is assumed that the process is consistently performed among its executions. Process attributes are briefly described below:

Input: 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: An entity that have been produced by the process or created in its operation to satisfy process purpose. The process has a number of outputs from each execution.

Activity: 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: 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: 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.

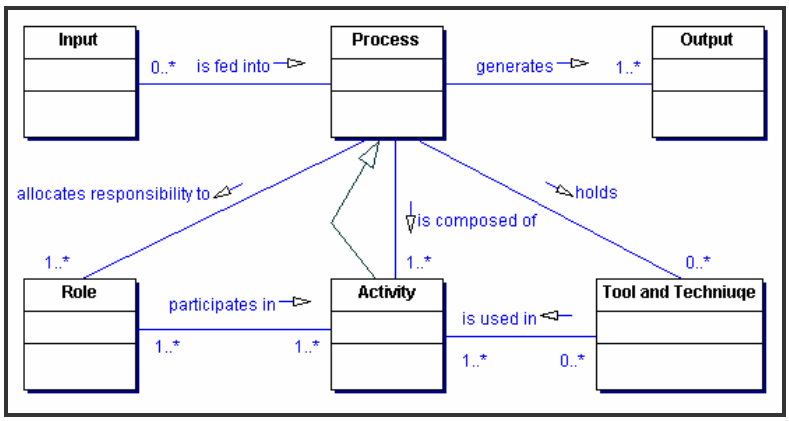


Figure 13 Process Attributes used for Stratification

Process consistency is assessed for similarity in process attribute values of process executions. The attribute values were recorded of each execution on a form, and to compare the similarity of these recorded values on a matrix. Ideally it is desirable that the process has a unique version in execution. The idea behind process

consistency assessment as basis for stratification is to identify, if any, these differing versions of a process in execution.

The second requirement is metric utilization. This includes elaboration of basic measurement practices as well as metric data existence and characteristics. Measurement practices should be performed for a specific purpose and, metrics should be uniquely understood to enable consistent implementation. Unique understanding (mostly enabled by constructing operational definitions) requires three criteria: communication, repeatability, and traceability. The traceability requirement is especially important to assessing and improving process performance. Because measures of performance can signal process instabilities, it is important that the context and circumstances of the measurement be recorded. This helps identifying assignable causes of the instabilities. There are studies that define procedures for successfully implementing measurement practices and for incorporating measurement capability into the projects of an organization. The CMMI for example, introduces Measurement and Analysis process area at maturity level 2, and recommends practices for defining data collection, storage, analysis, and reporting. Existence and implementation of these practices can be questioned for a specific project or organization to determine the utilization of existing metrics and data. Also, there are high-maturity companies that developed the factors to consider for measurement evaluation and to determine what measures to select for their specific use.

To evaluate metric utilization, a number of metric usability attributes were identified, and developed questionnaires based on these attributes for base and derived metrics separately. Table 1 lists and explains these attributes. Questionnaires include a rating system based on the answers of questions, and accordingly, evaluate the usability of a specific metric for applying SPC. A metric must satisfy the scale type requirement (absolute or ratio) and have enough data points to use (20 at a minimum) as specified by the first two attributes. Verifiability and dependability of metric data significantly contribute to the confidence in data analysis results. Data verifiability is related with the consistency in metric data recording and storage among executions. Data dependability requires all metric data be recorded as close to its source with accuracy and precision. The awareness of data collectors on metric data (why it is collected, how it is utilized, etc.) plays a significant role in data

dependability. The last two attributes, data normalizability and data integrability, are related with the usefulness of a metric and should be satisfied if we expect SPC analysis provide more insight for process understanding and improvement.

Table 1 Metric Usability Attributes used for Evaluating Metric Utilization

Metric Usability Attribute	Explanation
Metric Identity	Metric should be identified including entity and attribute to measure; scale type, unit, formula; and data type and range. Included in the identity is the scale type of the metric. Nominal and ordinal scale metrics cannot be used for control charting.
Data Existence	For any analysis, there should be measurement data. For control limits to be calculated reliably there should be at least 20 data points.
Data Verifiability	Metric data should be recorded at the same place in the process, by the same responsible body, and using the same method every time.
Data Dependability	Metric data should be recorded and stored as it is generated to ensure accuracy and precision; and be collected for a specific purpose. Feedback mechanisms should exist and be known by data collectors regarding data analysis and reporting.
Data Normalizability	Metric data can be normalized with a parameter or with another metric. Normalizing metric-A with a parameter-P provides comparable values of metric-A in terms of the parameter-P. Normalized metrics provide more insight in terms of statistical analysis (e.g., normalizing number of defects in a product with product size).
Data Integrability	Metric data can be integrated at project or organization levels. In practice, metric data should be integrated from individual level up to organization level for the results of statistical analysis to be effective organization-wide.

4.2 Assessment Process

The assessment process to follow when applying the model is given in Figure 14.

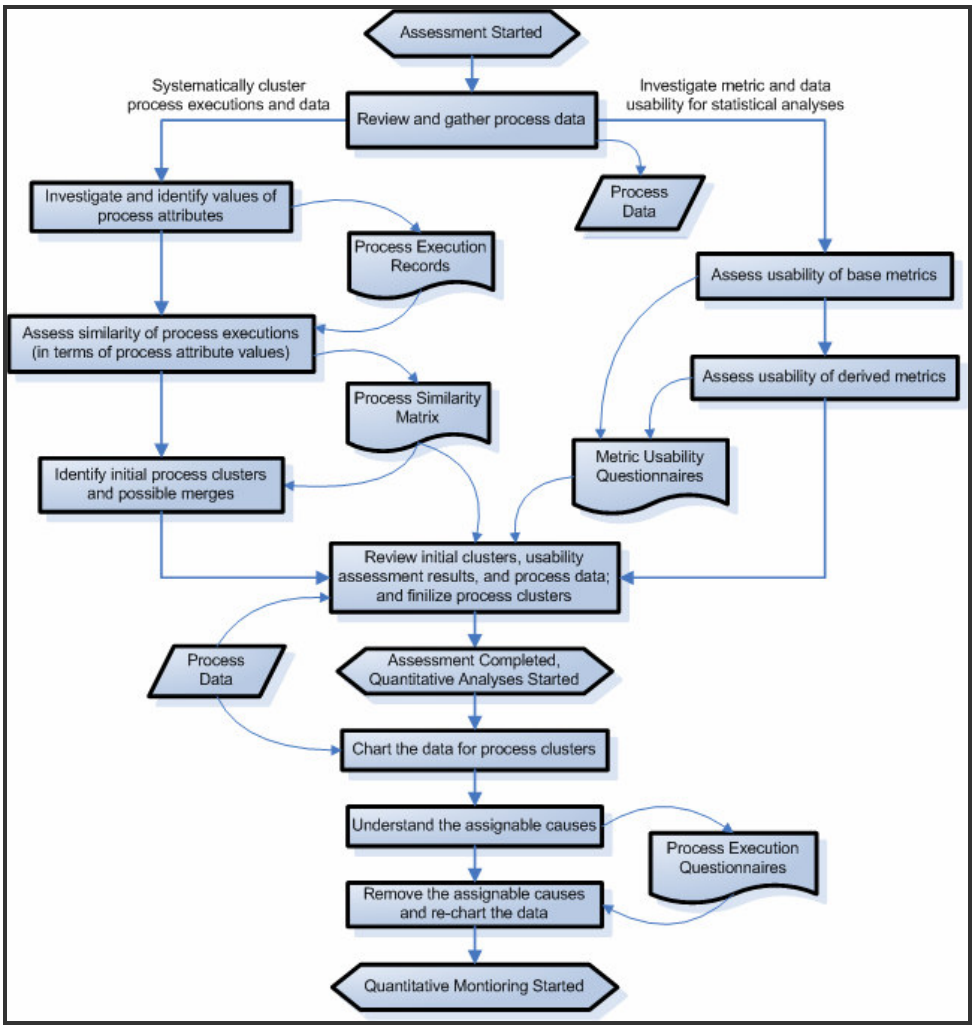


Figure 14 The Assessment Process

The first step of the assessment process is reviewing and gathering process data typically in a data file. Data should be consolidated in time sequence and in a form that is appropriate for comparison among different projects and product types. During consolidation, traceability should be established between process executions and data, typically by giving the same identifier to both. The data of process executions having missing, incomplete, or invalid data points should be excluded.

The flow at the left side of the figure is for performing stratification. The values of process attributes were investigated and identified for process executions by filling

out process execution records. If the study is retrospective then several executions were sampled from past process performances and fill a record for each. A merged list of values is built from process attribute values of sampled executions on records and entered into process similarity matrix for verification against entire set of process executions. The list on the matrix is extended during verification when a new value shows up.

If the study is prospective, a process execution record is filled when a new instance of the process is being executed. This increases the confidence on the values of process attributes for a process execution. Another difference in a prospective study is that a process execution questionnaire was completed for each instance of the process in execution and at the same time a process execution record (not while searching for the assignable causes later in the process as shown in figure). This was to capture the external factors affecting the process execution more timely, and have the chance of identifying likely assignable causes in advance.

The last step of the flow at the left side of Figure 14 as basis for stratification was identifying initial process clusters and possible merges among them by analyzing the process similarity matrix.

The flow at the right side of the figure was for evaluating metric utilization. First, usability of each base metric and then usability of each derived metric is evaluated by filling a metric usability questionnaire, and calculating regarding metric usability result.

After initial process clusters were identified and usability of process metrics were evaluated, the knowledge that is gathered so far was used as well as process data to finalize process clusters and metrics as basis for control charting. This is where the flows at left and right sides join in Figure 14. Here initial process clusters and possible merges were reviewed among them, the number of data points for each process cluster, and the usability status of process metrics; and the resulting process cluster-process metric pairs to chart are identified. This model recommends charting the data for process metrics that are evaluated as “usable” for statistical analysis; however, it might be a good idea to chart the data for the metrics that are evaluated as “not usable” to validate (or invalidate) the model’s recommendation. It

is better to review the number of data points per process metric basis since there may be missing data points.

The data was separately put for *process cluster-process metric* pairs on control charts, and watched for the out-of-control points. In a retrospective study, process execution questionnaire is filled for each out-of-control point to understand the assignable causes if any. In a prospective study, previously filled process execution questionnaires were reviewed to understand the assignable causes. Additionally, performing interviews with process performers was suggested to detect any reasons for out-of control points, or potential assignable causes that the process execution questionnaires cannot catch. After removing data points regarding the assignable causes at each chart, the data was re-charted for each *process cluster-process metric* pair and watch if the data on the chart is under control. Here is the place to judge whether approach helped in starting SPC. If a chart regarding a process cluster-process metric pair validates the findings of the assessment model, then SPC monitoring begins for that pair.

4.3 Assessment Assets

The model defines several assets exist for use in the assessment to perform stratification and to evaluate metric utilization. Process execution record together with process similarity matrix is utilized to identify process clusters as basis for stratification. Metric usability questionnaires were used to evaluate metrics' usability for SPC, and process execution questionnaire was used to investigate assignable causes for an out-of-control point on a control chart. The following paragraphs describe these assets.

Process Execution Record is a form used to capture the instant values of process attributes for a process execution. Actual values of inputs, outputs, activities, roles, and tools and techniques for a specific process execution are recorded on the form (Figure 15). Recorded values were used to identify the merged list of process attribute values which were entered into Process Similarity Matrix for verification.

Process Name:			Recorded On:	
Process Execution No:			Recorded By:	

1. Inputs: Please list the inputs to the process execution.

No	Name	Description
1		

2. Outputs: Please list the outputs from the process execution.

No	Name	Description
1		
2		

3. Activities: Please list in sequence the activities that were performed while executing the process.

No	Name	Description
1		
2		

4. Roles: Please list the roles that were allocated responsibilities in process execution.

No	Name	Description
1		
2		

5. Tools and Techniques: Please list the tools and techniques that are used to support process execution.

No	Name	Description
1		
2		

Figure 15 Process Execution Record

Process Similarity Matrix is a spreadsheet used to verify process attribute values against process executions. Process attribute values were recorded into the rows of the matrix vertically and process execution numbers were recorded into the columns of the matrix horizontally. By going over process executions, the values of process attributes were questioned and marked if applicable for each process execution (Figure 16). The completed matrix helped to see the differences among process executions in terms of process attribute values, and enabled to identify stratificated samples of the process executions accordingly.

			Process Executions																			
Process Attributes			PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8	PE9	PE10	PE11	PE12	PE13	PE14	PE15	PE16	PE17	PE18	PE19	PE20	.....
1	**Inputs**																						
	1.1	<Input-1>	o	o	.....																		
	1.2	<Input-2>	o	o	.....																		
2	**Outputs**																						
	2.1	<Output-1>	o	o	.....																		
	2.2	<Output-2>		o	.....																		
3	**Activities**																						
	3.1	<Activity-1>	o	o	.....																		
	3.2	<Activity-2>	o	o	.....																		
	3.3	<Activity-3>	o	o	.....																		
	3.4	<Activity-4>		o	.....																		
4	**Roles**																						
	4.1	<Role-1>	o	o	.....																		
	4.2	<Role-2>	o	o	.....																		
5	**Tools and Techniques**																						
	5.1	<Tools and Techniques-1>	o	o	.....																		
	5.2	<Tools and Techniques-2>		o	.....																		

Figure 16 Process Similarity Matrixes

Metric Usability Questionnaire is a form used to investigate the usability of a process metric in terms of metric usability attributes. The form has two types, for base metrics (Figure 17a) and derived metrics (Figure 17b) separately. The form includes a number of questions as indicators of usability attributes. Answers to some questions are informative (shaded under “rating” column of MUQ in the figures) and answers to some are used to rate each usability attribute (expected answers to such questions are given in the rightmost column of MUQ in the figures). A metric usability attribute was rated as a corresponding metric usability factor (MUF) within four ordinal values, based on the answers to its indicators: Fully satisfied (F: %86-100), Largely satisfied (L: %51-85), Partially satisfied (%16-50), and Not satisfied (N: %0-15).

			Please rate each attribute in four scales, based on answers to questions as indicators:		
Metric Name:			F : Indicators of the atribute are fully satisfied (%86-100)		
Conceptual Definition:			L : Indicators of the atribute are largely satisfied (%51-85)		
Assessed On:			P : Indicators of the atribute are largely satisfied (%16-50)		
Assessed By:			N : Indicators of the atribute are not satisfied (%0-15)		
Attributes			Answers	Rating	Expected Answers
	Indicators				
Metric Identity			MUF-1	F	
	Q1	Which entity does the metric measure?			
	Q2	Which attribute of the entity does the metric measure?			
	Q3	What is the scale of the metric data? (nominal, ordinal, interval, ratio, absolute)			Ratio, Absolute
	Q4	What is the unit of the metric data?			
	Q5	What is the type of the metric data? (integer, real, etc.)			
	Q6	What is the range of the metric data?			
Data Existence			MUF-2	F	
	Q7	Is metric data existent?			Available > 20
	Q8	What is the amount of overall observations?			
	Q9	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			MUF-3	F	
	Q12	When is metric data recorded in the process? (at start, middle, end, later, etc.)			
	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?			
	Q15	Is all metric data recorded by the responsible body?			Yes
	Q16	How is metric data recorded? (on a form, report, tool, etc.)			
	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.)			
	Q19	Is all metric data stored in the same place? (in a file, database, etc.)			Yes
Data Dependability			MUF-4	F	
	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?			

(a) Metric Usability Questionnaire

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	**F**	L or F (Usable) -- Not Usable otherwise

(b) Metric Usability Rating

Figure 17 Metric Usability Questionnaire and Rating for Base Metrics

The values of metric usability factors were formed into a vector and evaluated to determine the metric usability result. Factor values are evaluated in the order of criticality of the attributes (1 being the most critical): 1) metric identity, 2) data existence, 3) data verifiability, and 4) data dependability. The regarding values of the vector should be at least [F, F, L, L] for a base metric to be usable (vector values of [F, F, L, P], for example, leads to a result of “not usable”). For a derived metric, vector values are evaluated together with the values of metric usability factors 3 and 4 of the base metrics that make up the derived metric. Metric usability factors of 3 and 4 of the base metrics should have a value of either F or L. A value of P or N for these attributes of a base metric leads to a result of “not usable” even if usability factor values of the derived metric satisfy [F, F, L, L]. While coding metric usability factors 3 and 4 of the base metrics for evaluation of usability of the derived metric; the lowest ordinal value was taken.

			Please rate each attribute in four scales, based on asnwers to questions as indicators:		
Metric Name:			F : Indicatiors of the atribute are fully satisfied (%86-100)		
Conceptual Definition:			L : Indicatiors of the atribute are largely satisfied (%51-85)		
Assessed On:			P : Indicatiors of the atribute are largely satisfied (%16-50)		
Assessed By:			N : Indicatiors of the atribute are not satisfied (%0-15)		
Attributes			**Answers**	**Rating**	Expected Answers
	Indicators				
Metric Identity			**MUF-1**	**F**	
	Q1	What is the 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?			

(a) Metric Usability Questionnaire

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	F	L or F
MUF-3&4 for base metric-2	F	L or F
MUF-3&4 for base metric-n	F	L or F
Metric Usability Result	**F**	L or F (Usable) -- Not Usable otherwise

(b) Metric Usability Rating

Figure 18 Metric Usability Questionnaire and Rating for Derived Metrics

For example, assume that the usability of “defect density” derived metric is evaluating and rate the attribute values as [F, F, F, L]. If the values of metric usability factors 3 and 4 of base metric “number of defects” are [F, L], the factors were coded as “L” (the lowest of [F, L]) as basis for evaluating usability of “defect density”. Similarly, if the values of metric usability factors 3 and 4 of base metric “product size” are [L, L], the factors were coded as “L” again (the lowest of [L, L]). Then, since the metric usability factors of “defect density” are rated as [F, F, F, L]

and the usability ratings for factors 3 and 4 for both base metrics are “L”, it was concluded that “defect density” derived metric is *usable* for statistical analysis. However, if the value of metric usability factor 3 or 4 was P for any of the base metrics, “defect density” would *not be usable* for statistical analysis.

Process Execution Questionnaire is a form used to investigate the external factors that might affect a process execution so that assignable causes exist. External factors are questioned in terms of changes in process performers, process environments, and other factors if any (Figure 19). While working retrospectively on existing process data, this form is used to understand the assignable causes for a process execution if it led to an out-of-control point. In a prospective study, however, the form is filled for each instance of the process in execution to identify the external factors that might be a potential assignable cause.

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 19 Process Execution Questionnaires

4.4 An Assessment and Analysis Tool for Statistical Process Control

SPC-AAT has facilities to capture data from outer environment, assess the suitability of software processes and metrics for SPC, and analyze a software process with respect to its qualifying metrics using SPC techniques like control charts, histograms, bar charts, and pareto charts. Accordingly, user interface of the tool has three main views: Process Data, Assessment, and Process Improvement.

SPC-AAT works integrated with other tools in the environment which hold measurement data about the processes performed. When measurement data is imported to SPC-AAT, all necessary assets are created automatically by the tool before SPC assessment and analysis are started.

The SPC techniques are applied on “process cluster - metric” pairs. A metric value which is detected as out-of-control point (OCP) according to the tests applied can be excluded from the analysis via the tool. To exclude an OCP and see related process execution questionnaire, one just clicks on the point on a control chart. SPC-AAT also supports what-if analysis for different stratification choices by merging and splitting current process clusters. As a last thing, SPC assessment and analysis results can be reported and printed by using the tool [32, 33].

5 CASE STUDY

The project used in this study had been implemented based on the Organization Software Development Methodology which depends on waterfall model. This methodology has been generated to cover the goals of CMMI L3 (Figure 20). This study is about the System Testing phase of the project.

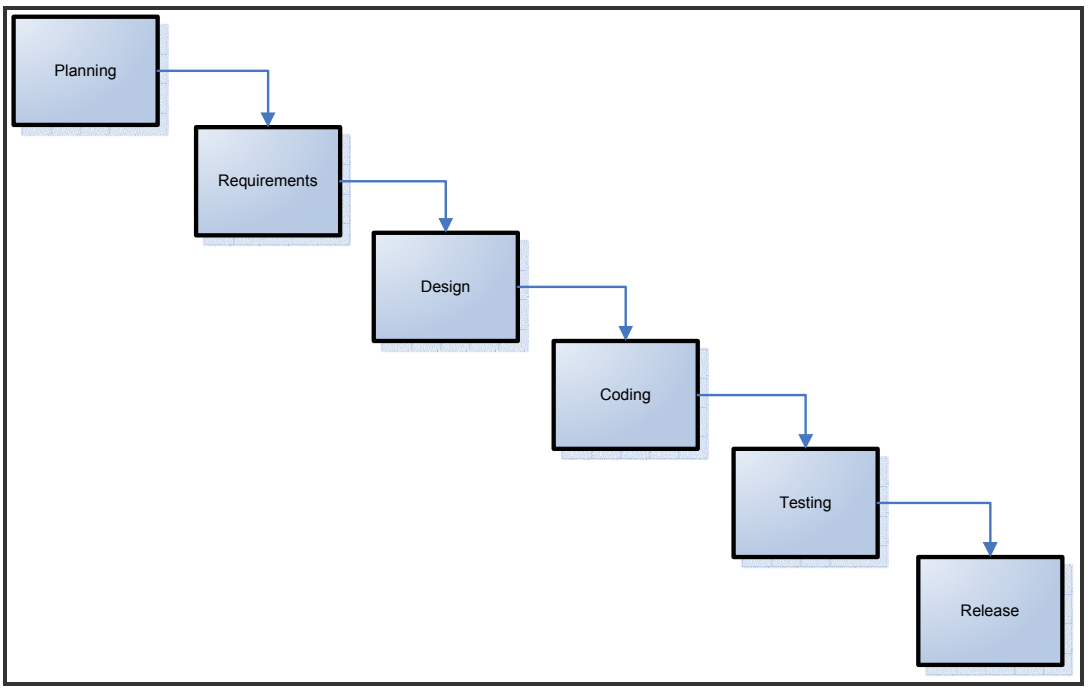


Figure 20 Organization Software Development Methodologies

One of the CMMI L3 process areas is Verification and Validation. System Testing is subject to this area and instructions of the system testing in our organization are defined below:

The system test design activity can be initiated by the completion of SRS and is completed before the start date of the system tests. System tester/test team prepares the system test cases based on the test strategy defined in the Test Plan and business scenarios/use cases identified in the SRS. System tester/test team records the system test cases into the requirements management tool and establishes the traceability between the system test cases and Use cases. The system test environment is prepared in accordance to the requirements defined in the Test Plan. System tester/test team ensures that the system test environment is ready with respect to the system environment requirements defined in the Test

Plan. The system to be tested is integrated and deployed to the system test environment. System Tester/Test Team perform(s) the system tests according to the test methods, constraints and validation criteria that are stated in the System Test Case Document. System Tester/Test Team ensure(s) that the system works as it is expected in its intended operational environments. System Tester/Test Team issues the defects that are found in product and product component test and issues via Configuration Management Tool. At the end of the system test, System Tester/Test Team update(s) the System Test Case Document or records the results in the related documents. System Tester/Test Team places the records under configuration control. System test results are analyzed and recorded periodically and corrective actions are taken if necessary.

If we look at the purpose of this study, we need to explain which coverage methods are being used to implement system testing activity. In our organization, path coverage has been defined, but node coverage has not been defined at system test level shown in Table 2.

Table 2 The Interpretation of Path and Node Coverage at System Test Level

Path testing (path coverage)	Independent paths (basis paths) through the control structure of the operational scenarios are exercised. Activity diagrams can be used to define the test cases at system level.
Statement testing (statement/ node coverage)	Not applicable at system level

Two case studies were implemented at a project-based working software organization (referred as organization X in the study) having CMMI L3. System test

process of the project and related metrics of these processes had been worked on. These metrics were used in the case studies can be seen in Table 3.

The project used here is a large data entry and query system developed on networked, client/server, server utilizing Java and IBM DB2. The project has two modules, Data Entry and Reporting. System testing has contained both modules and these modules were tested together. The project had been developed during 6 months with a staff of 5 with approximately 8,000 lines of code. The project had achieved L3 in the SW-CMMI, and the organization is pursuing L4. All L4 processes were installed and conducted on the project during a period of time.

Table 3 Metrics (Base and Derived) used in the Case Studies

Metric Name	**Description**
Number of Test Cases Defined	Base Metric
Number of Failed Test Cases	Base Metric
Number of Passed Test Cases	Base Metric
Functional Size	Base Metric
Test Case Execution Time	Base Metric
Test Defect Density (# Failed Test Cases / # Test Cases Defined)	Derived Metric
Test Effectiveness (# Failed Test Cases / Test Case Execution Time)	Derived Metric
Test Speed (# Test Cases Defined / Test Case Execution Time)	Derived Metric

For the both case studies described in this study, two coverage methods had selected to implement. The first method is “Node Coverage” method that there has been 18 user interfaces, 191 nodes had been tested; and the second one is “Path Coverage” method that there has been 18 user interfaces, 69 paths and related 297 nodes had been tested and data had been recorded prospectively for both case studies, Case Study A and Case Study B. Number of node is larger for Path Coverage case in result of there were duplicated test cases for different paths. Interface is the unit of measure for both case studies. Test cases are utilized as data for these interfaces. Test metrics are an important indicator of the effectiveness of a software testing process. Test metrics that had been defined in this study were decided in according to section 2.3 of this study. All metrics defined

for the case studies can be seen from Table 3 were collected at interface based except the “Test Case Execution Time” base metric. Test Case Execution Time base metric was collected at test case based, then the total time for this metric was evaluated. Besides, in this study first passes of test cases were evaluated, second and third passes were not evaluated because of the time constraint caused by the organization.

Although there is no historical data and ability of the process to generate 20-25 metric data points in the near future [31], in result of this project is a real time project, executed processes are 18 for each case.

One often assumes that the data are from an approximately normally distributed population. This is frequently justified by the classical central limit theorem, which says that sums of many independent, identically-distributed random variables tend towards the normal distribution as a limit. If that assumption is justified, then about 68 % of the values are within 1 standard deviation of the mean, about 95 % of the values are within two standard deviations and about 99.7 % lie within 3 standard deviations [34].

The rules shown in Figure 21 were chosen to be used when detecting Out-of Control Points.

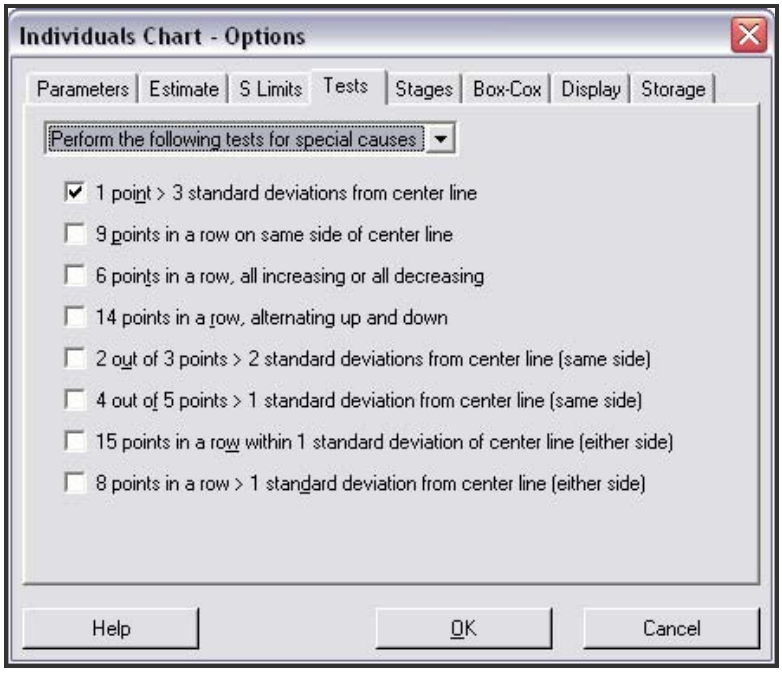


Figure 21 Rules for Out-of-Control Points

5.1 Case Study A

In the scope of Case Study A, system testing of a real time project was utilized. Firstly, SRS document was written by the system analyst of the project in April 2007. STC document was written according to SRS document in May 2007 by a system tester. Test cases were written per user interface defined in the SRS document. “Node Coverage” method has been implemented to 18 user interfaces, and 191 nodes were utilized for this case.

SPC assets were used for collecting data in this case. When this case of the study had started, SPC-AAT was not ready to use. It is aimed to collect data prospectively, therefore all information were recorded to the forms which were provided in Appendix A and these information were saved in the folders. After the SPC-AAT had got ready to utilize, all data were entered to SPC-AAT.

Process attribute values were identified to put on process similarity matrices by filling process execution records. 191 test case instances were sampled and a process execution record (completed questionnaires for all metrics identified in Case Study A are provided in appendix A) was completed for each. The information on process execution records were provided typical values of process attributes, and formed an initial base for creation of the similarity matrix. There were 18 process execution records for system test. Completed process similarity matrix for Inputs, Outputs, Activities, Roles, and Tools & Techniques of all system test process instances can be seen from Figure 22 to 26.

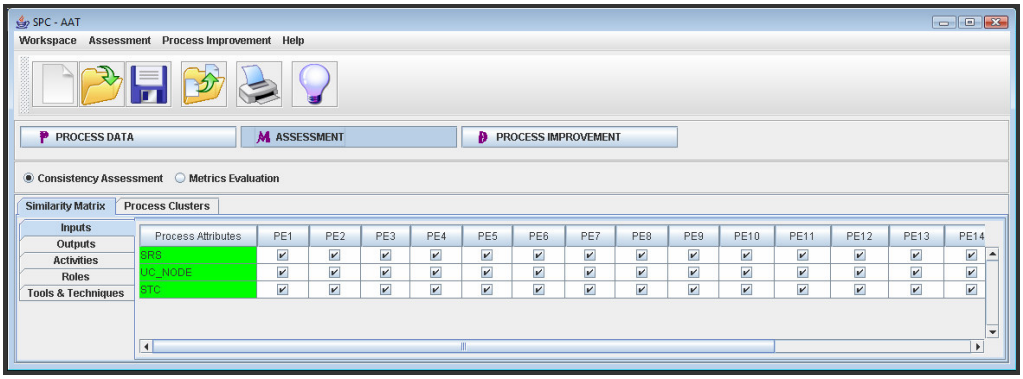


Figure 22 Similarity Matrixes for Inputs – Case Study A

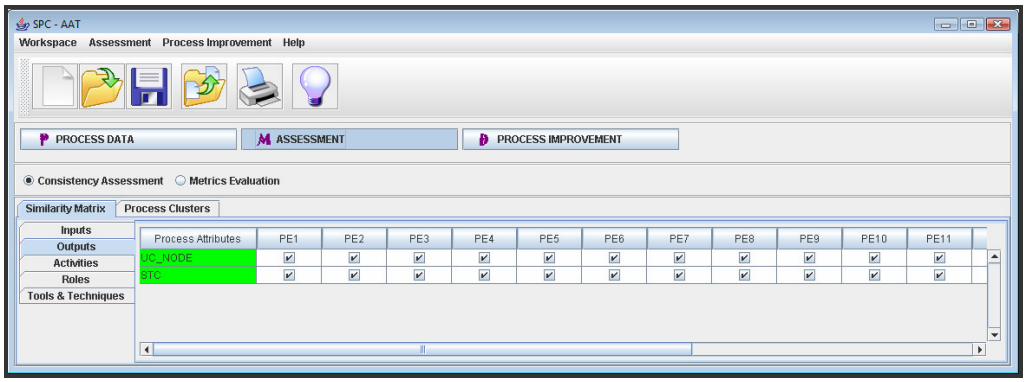


Figure 23 Similarity Matrixes for Outputs – Case Study A

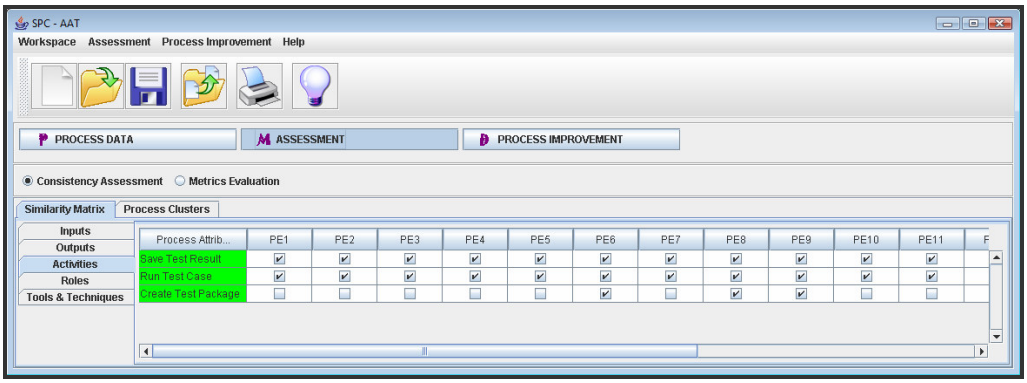


Figure 24 Similarity Matrixes for Activities – Case Study A

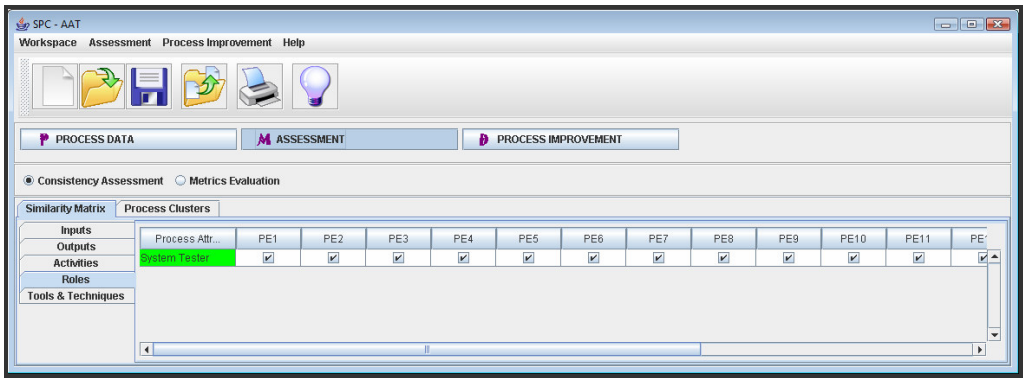


Figure 25 Similarity Matrixes for Roles – Case Study A

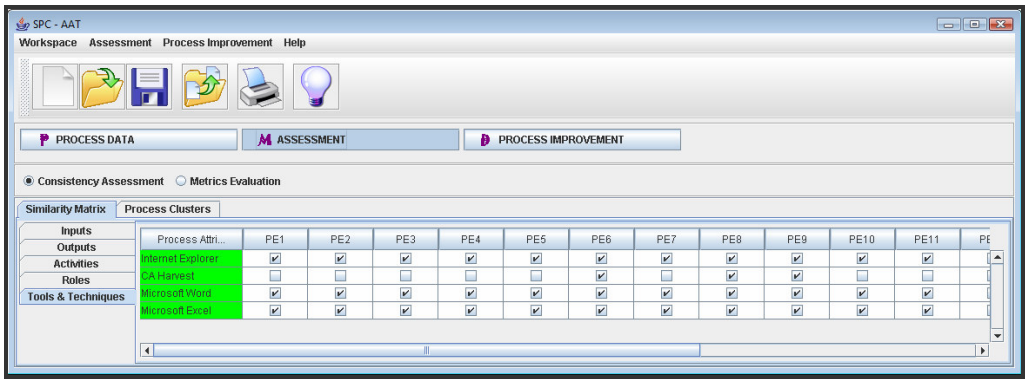


Figure 26 Similarity Matrixes for Tools & Techniques – Case Study A

Process similarity matrix for similarity and differences were analyzed in process executions. After finalizing the matrix, 2 process clusters were labeled A and B as shown in Figure 27. The number of data points were not enough (at least 20) for Version A and Version B. Though, we decided to chart data separately for these two versions to understand the effects of process clustering.

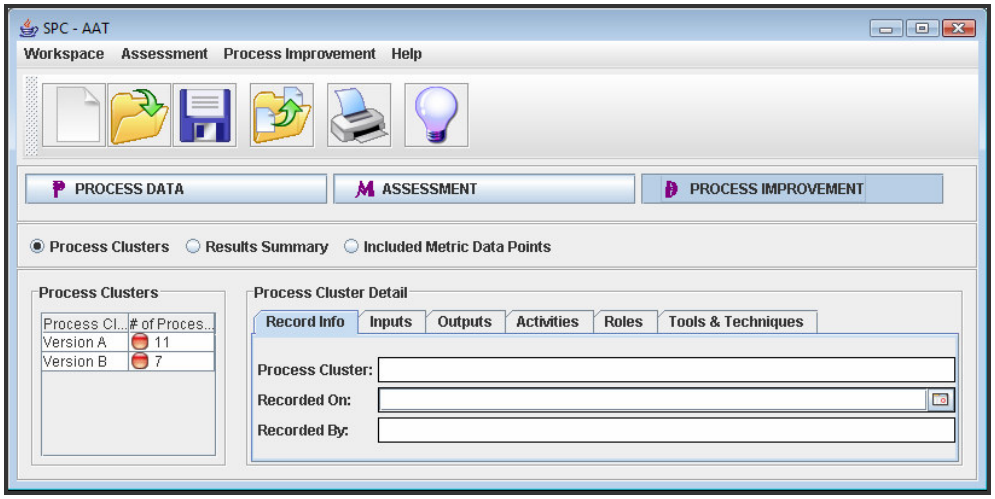


Figure 27 Base Process Clusters for System Test Process

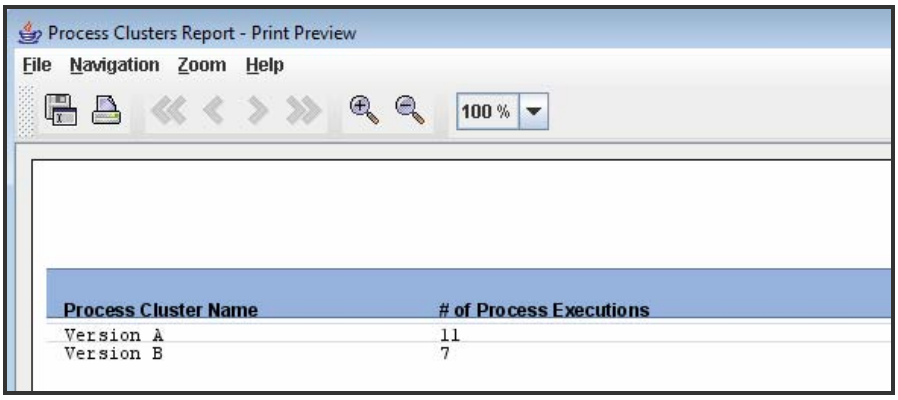


Figure 28 Process Clusters Report

As shown in Figure 29 distances between the process clusters is 2. These distances are based on Process attributes defined in activities and tools&techniques of Process Clusters A and B.

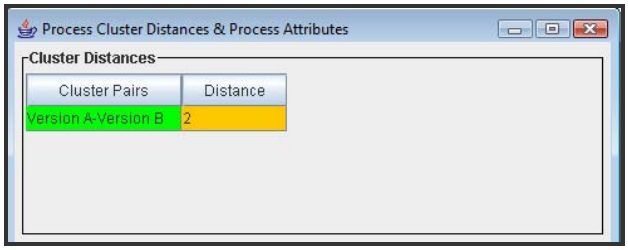


Figure 29 Process Cluster Distances & Process Attributes

Create test package activity shown and CA Harvest tool made the difference between these two clusters shown in Figure 30 and Figure 31.

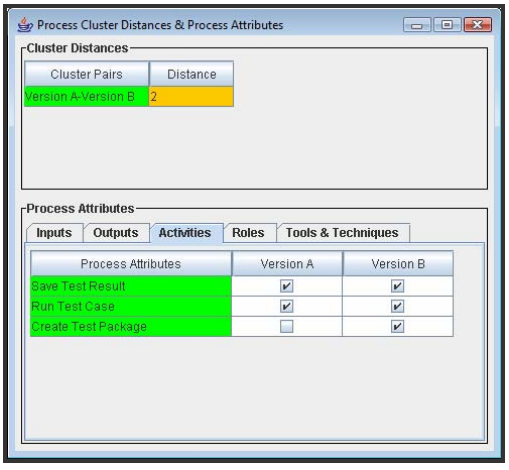


Figure 30 Process Cluster Distances & Process Attributes

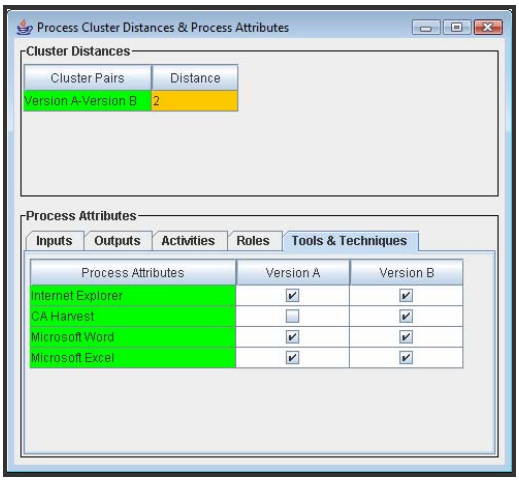
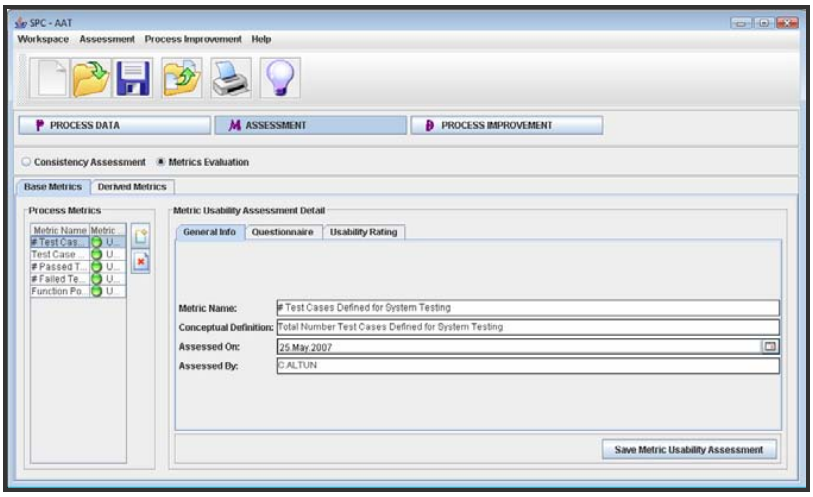


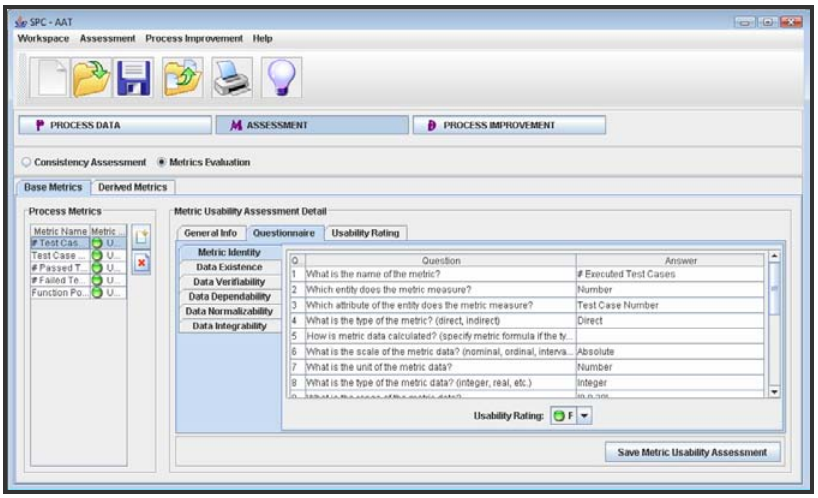
Figure 31 Process Cluster Distances & Process Attributes

After identification of initial process clusters, process metrics were utilized to evaluate their usability for statistical analysis. Number of test cases defined, test case execution time, number of passed test cases, number of failed test cases, and functional size as base metrics were identified. These were the metrics for which data were available on the tool. From the base metrics, test defect density, test effectiveness and test speed were identified as derived metrics of the system test process.

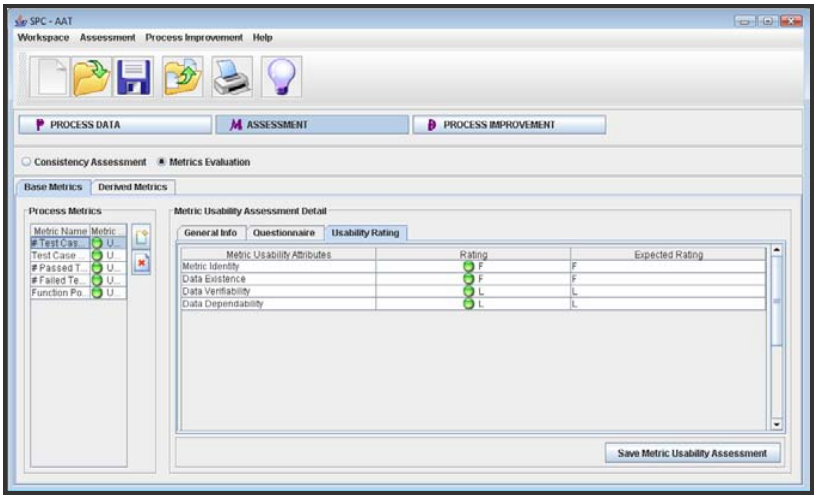
Metric Usability Questionnaire was filled for each base and derived metric from Questionnaire tab-sheet under Metric Evaluation view (excel sheet was filled before the tool had been ready to use). Example questionnaire for “Number of Test Cases Defined” base metric with its info, questionnaire and usability ratings are shown in Figure 32 (completed questionnaires for all metrics identified in Case Study A are provided in Appendix C).



(a) General Info Tab



(b) Questionnaire Tab



(c) Usability Rating Tab

Figure 32 Metric Usability Questionnaire and Rating for Number of Test Cases Defined

The usability status of all base and derived metrics are listed in Figure 33. All the metrics which were defined at the beginning of this case are usable for the node coverage method for system testing process and usability states can be seen from metric usability evaluation report.

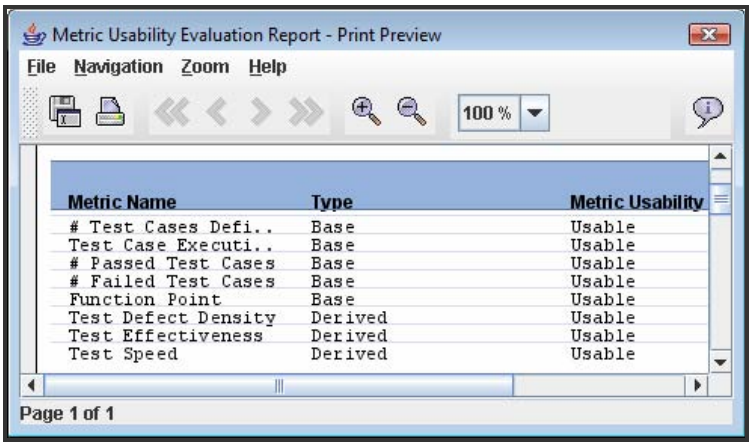
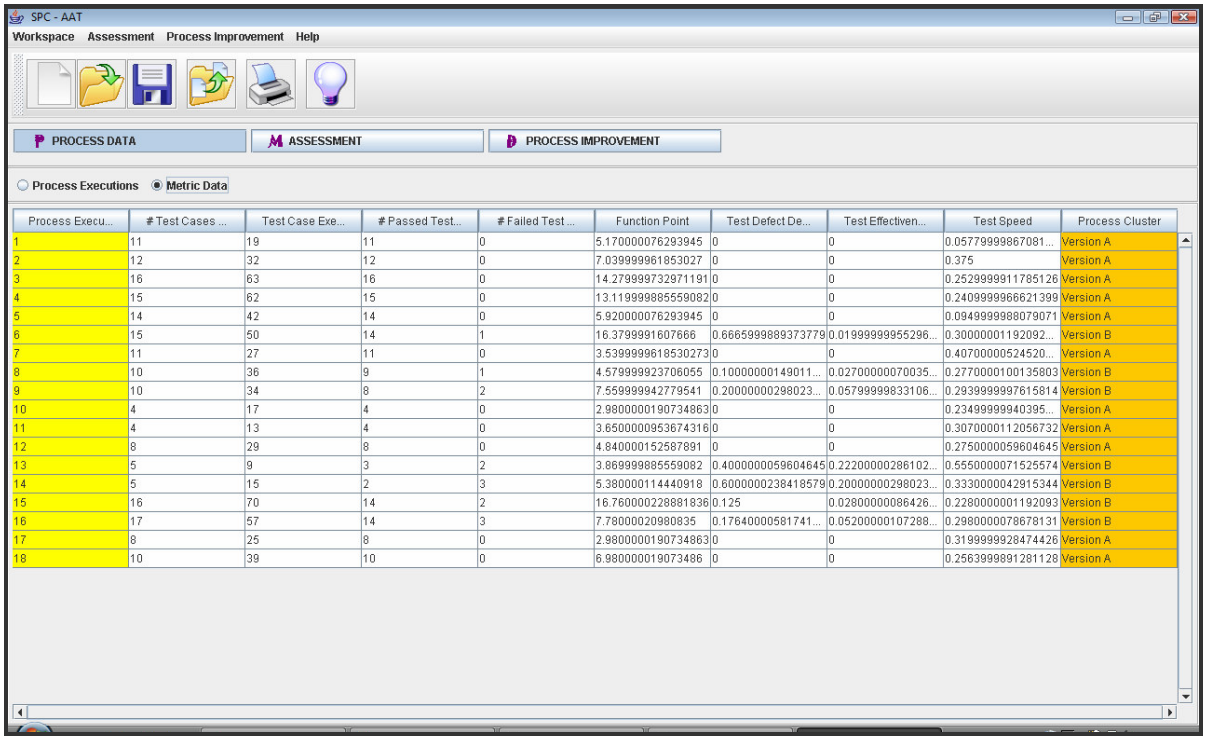


Figure 33 Metric Usability Report for Node Coverage Process

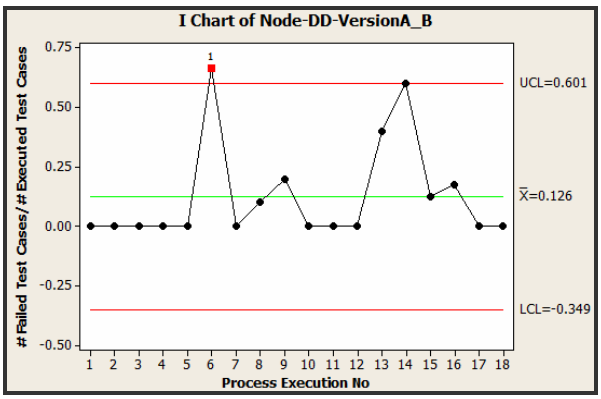
Test data for node coverage method used in Case study A was completed in SPC-AAT as shown in table 34.



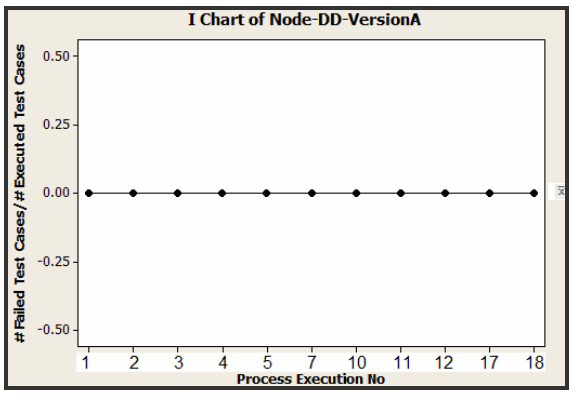
Process Execu...	# Test Cases ...	Test Case Exe...	# Passed Test...	# Failed Test ...	Function Point	Test Defect De...	Test Effectiven...	Test Speed	Process Cluster
1	11	19	11	0	5.170000076293945	0	0	0.057799998670810...	Version A
2	12	32	12	0	7.039999961853027	0	0	0.375	Version A
3	16	63	16	0	14.279999732971191	0	0	0.2529999911785126	Version A
4	15	62	15	0	13.119999885559082	0	0	0.2409999966621399	Version A
5	14	42	14	0	5.920000076293945	0	0	0.0949999988079071	Version A
6	15	50	14	1	16.3799991607666	0.666599988937377 9	0.01999999955296...	0.300000011920 92...	Version B
7	11	27	11	0	3.5399999618530273	0	0	0.407000005245 20...	Version A
8	10	36	9	1	4.5799999237060 55	0.100000001490 11...	0.027000000700 35...	0.2770000100135803	Version B
9	10	34	8	2	7.559999942779541	0.200000002980 23...	0.057999998331 06...	0.2939999997615814	Version B
10	4	17	4	0	2.980000019073486 3	0	0	0.234999999403 95...	Version A
11	4	13	4	0	3.650000095367431 6	0	0	0.307000011205 6732	Version A
12	8	29	8	0	4.840000152587891	0	0	0.275000005960 4645	Version A
13	5	9	3	2	3.869999885559082	0.400000005960 4645	0.222000002861 02...	0.5550000071525574	Version B
14	5	15	2	3	5.380000114440918	0.600000023841 8579	0.200000002980 23...	0.3330000042915344	Version B
15	16	70	14	2	16.760000228881836	0.125	0.028000000864 26...	0.2280000001192093	Version B
16	17	57	14	3	7.78000020980835	0.176400005817 41...	0.052000001072 88...	0.2980000078678131	Version B
17	8	25	8	0	2.980000019073486 3	0	0	0.319999992847 4426	Version A
18	10	39	10	0	6.980000019073486	0	0	0.256399989128 1128	Version A

Figure 34 Metric Data of Case Study A

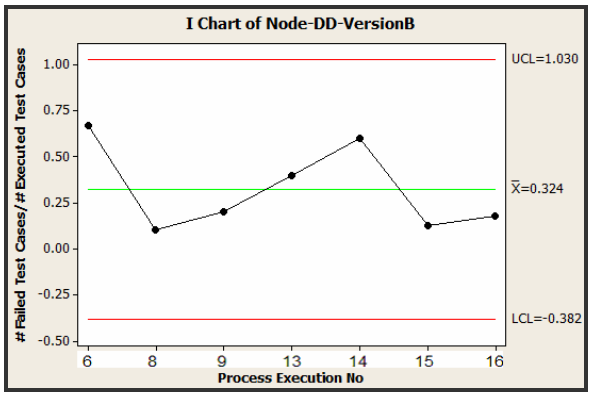
SPC tools were applied to the qualified process cluster – derived metric pairs. In this case (node coverage method), control charts drawn for process clusters - derived metric pairs are shown in the figures 35 to 37.



(a) Defect Density Metric for Version A-B



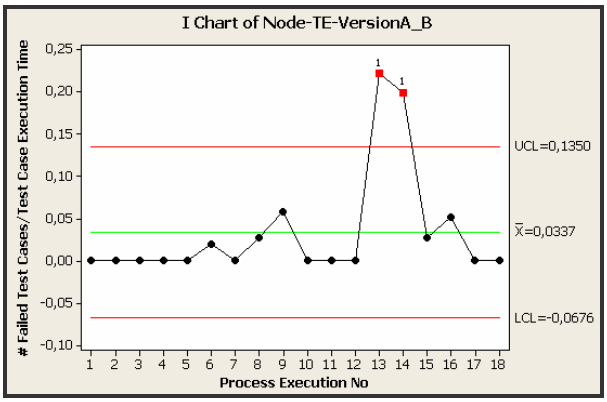
(b) Defect Density Metric for Version A



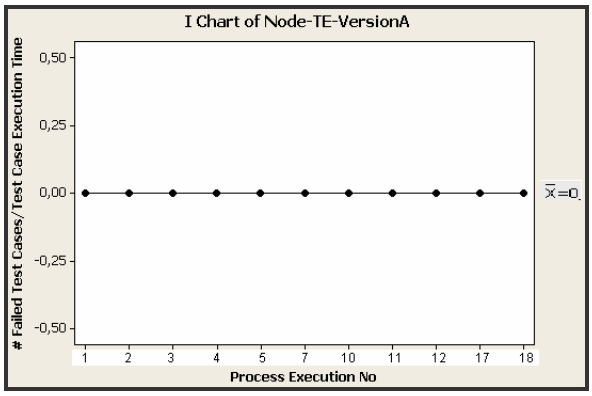
(c) Defect Density Metric for Version B

Figure 35 Individuals Charts for Derived Metrics of Test Defect Density for Node Coverage

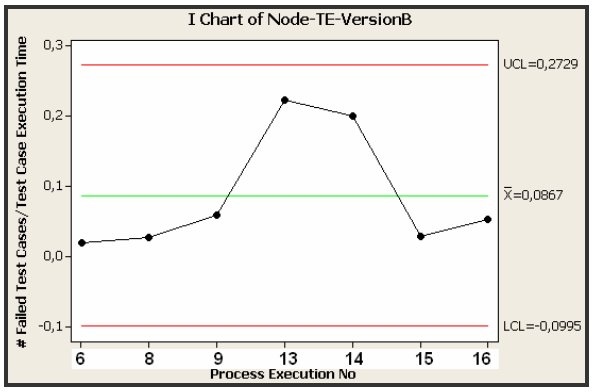
As it can be seen from Figure 35, version A indicates the process executions which do not contain any defected test cases. On the other hand, Version B indicates the process executions which contain defected test cases. Figures 35(a), 35(b) and 35(c) show that clustering worked well for defect density metric when using in node coverage method. It is because figure 35(a), version A-B, has one out of control point; where as figure 35(b), version A, and figure 35(c), version B, have no out of control points. Mean value of version A is equal to 0 (zero) because of the process executions of version A did not have failed test cases.



(a) Test Effectiveness Metric for Version A-B



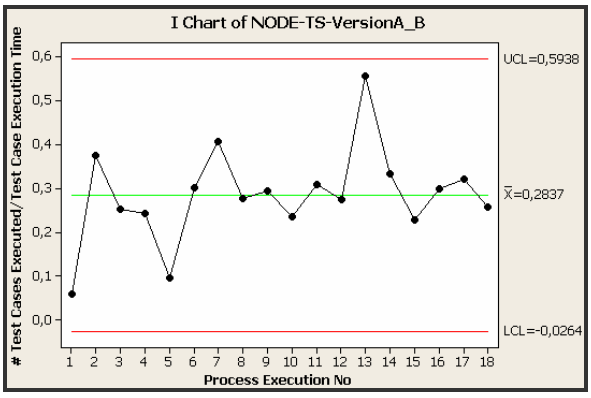
(b)Test Eff. Metric for Version A



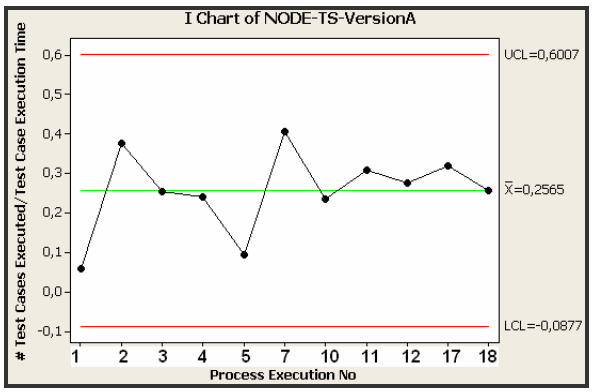
(c)Test Eff. Metric for Version B

Figure 36 Individuals Charts for Derived Metrics of Test Effectiveness for Node Coverage

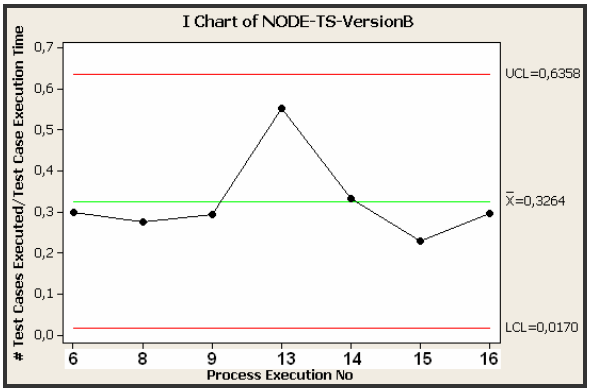
When looking at the figure 36(a) version A-B distribution by Test Effectiveness metric, there are two out of control points; where as figure 36(b), version A, and figure 36(c), version B, have no out of control points. Version A (36(b)) indicates the process executions which do not contain any defected test cases. On the other hand, Version B (36(c)) indicates the process executions which contain defective test cases. Clustering here identified and classified process executions on the basis of the similarity of the characteristics they possess. Figures 36(a), 36(b) and 36(c) show that clustering worked well for test effectiveness metric when using in node coverage method. Mean value of version A is equal to 0 (zero) because of the process executins of version A did not have failed test cases.



(a) Test Speed Metric for Version A-B



(b) Test Speed Metric for Version A



(c) Test Speed Metric for Version B

Figure 37 Individuals Charts for Derived Metrics of Test Speed for Node Coverage

For node covarage method, test speed metric for all versions is under control can be seen in figures 37(a), 37(b) and 37(c). It can be said that test speed values calculated for this case are stable. The mean value of test speed for version A was lower than the mean value of test speed for version B, because number of defined test cases have a direct ratio with test speed derived metric. Nevertheless number of test cases defined for version A are higher than number of test cases defined for version B. Besides figure 37 shows that clustering does not have remarkable effect on test speed metric when using in node coverage method. A comparison of control charts of derived metrics between Case A and Case B (that is, between node coverage and path coverage system testing techniques) is provided in the discussion section 6.1.

5.2 Case Study B

In this case, system testing of a real time project was implemented. Firstly, SRS document was written by the system analyst of the project in April 2007. STC document was written by the system tester according to SRS document in May 2007. Test cases were written per user interface defined in the SRS document. “Path Coverage” method has been implemented to 18 user interfaces, 69 paths, and 297 nodes in this case.

SPC-AAT tool was utilized for collecting data in this case. When this case was started to work, SPC-AAT was ready to use. Therefore all data were entered to SPC-AAT and Statistical Software tool (Minitab) was utilized for statistical analyses.

Process attribute values were identified to put on process similarity matrices by filling process execution records. For the path coverage case, 297 test case instances were sampled and process execution record (completed questionnaires for all metrics identified in Case Study B are also provided in appendix A) was completed for each. The information on process execution records provided typical values of process attributes, and formed an initial base for creation of the similarity matrix. There were 18 process execution records for system test path coverage method. Completed process similarity matrix for Inputs, Outputs, Activities, Roles, and Tools & Techniques of all system test process instances can be seen from Figure 38 to 42.

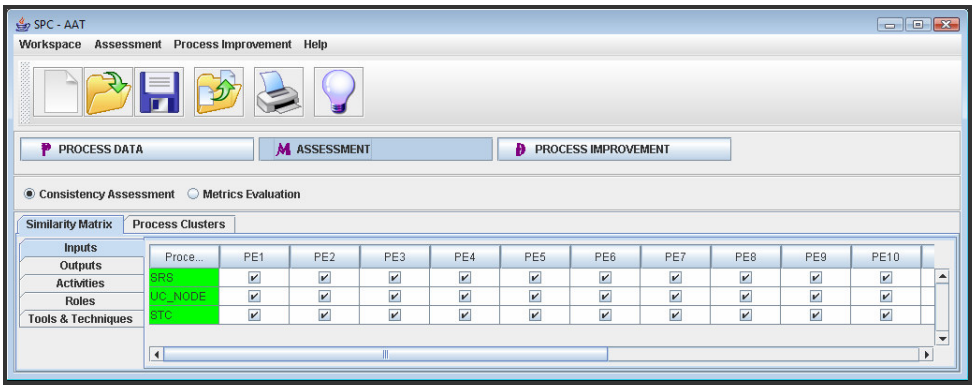


Figure 38 Process Similarity Matrixes for Inputs – Case Study B

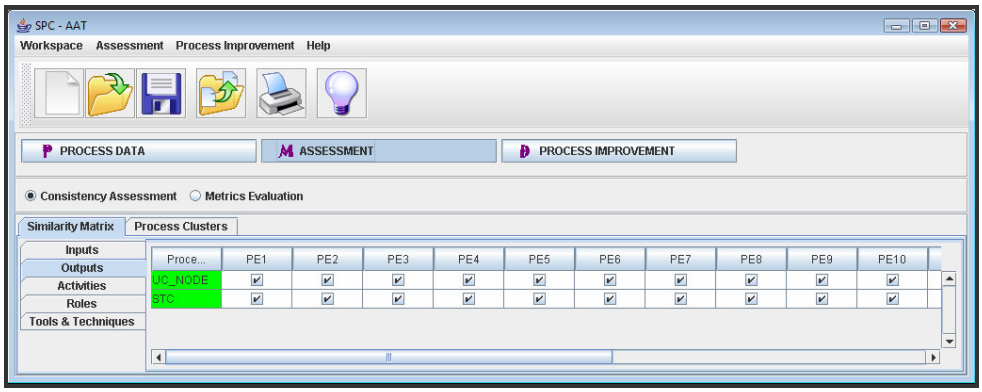


Figure 39 Process Similarity Matrixes for Outputs – Case Study B

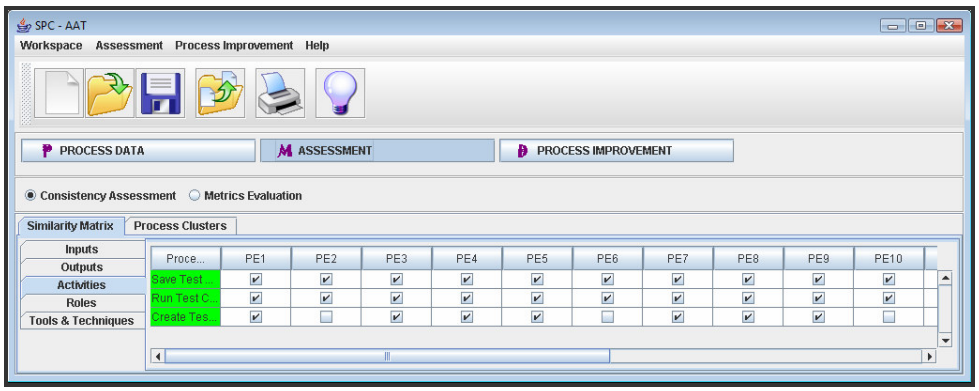


Figure 40 Process Similarity Matrixes for Activities – Case Study B

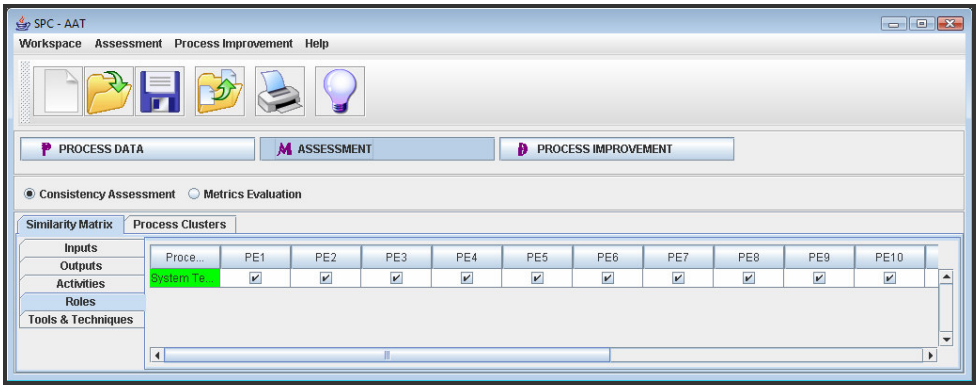


Figure 41 Process Similarity Matrixes for Roles – Case Study B

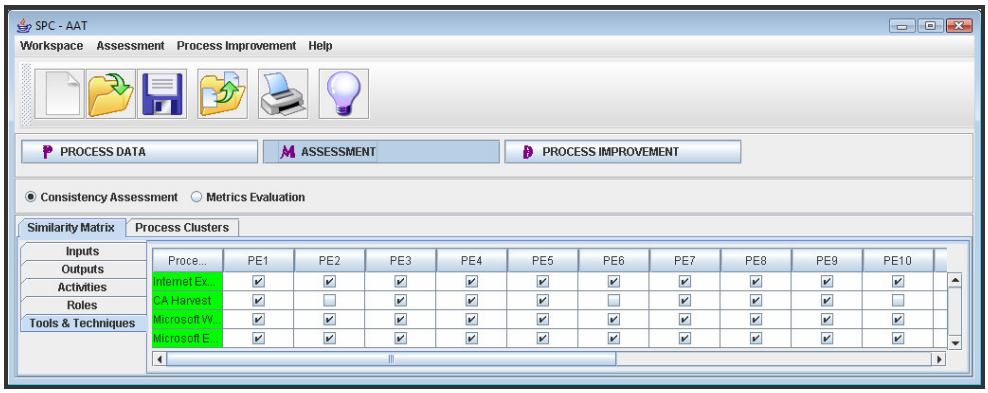


Figure 42 Process Similarity Matrixes for Tools and Techniques – Case Study B

Process similarity matrix for similarity and differences were analyzed in process executions for Case Study B. After finalizing the matrix, 2 process clusters labeled A and B as shown in Figure 43. The number of data points was not enough (at least 20) for Version A and Version B. Though, it is decided to chart data separately for these two versions to understand the effects of process clustering.

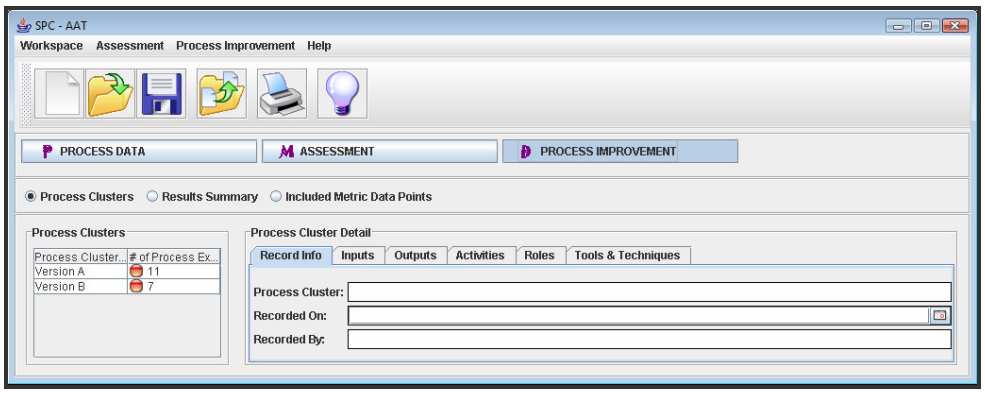


Figure 43 Base Process Clusters for System Test Process – Case Study B

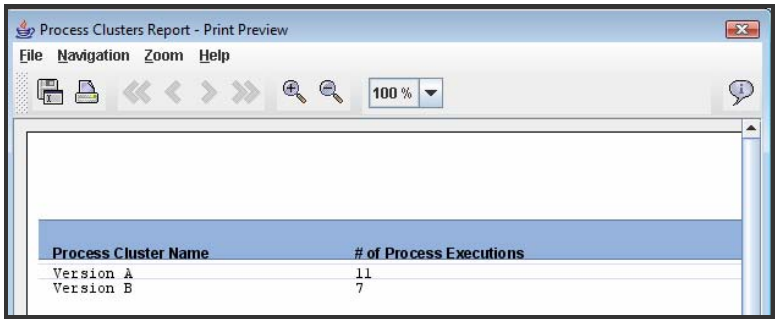


Figure 44 Process Clusters Report – Case Study B

As shown in Figure 45 distance between the process clusters is 2. This distance is based on Process attributes defined in activities and tools&techniques of Process Clusters A and B.

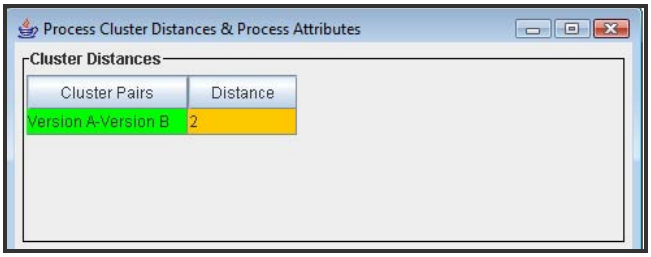


Figure 45 Process Cluster Distances & Process Attributes

Create test package activity shown in Figure and CA Harvest tool caused the difference between these two clusters shown in Figure 46 and Figure 47.

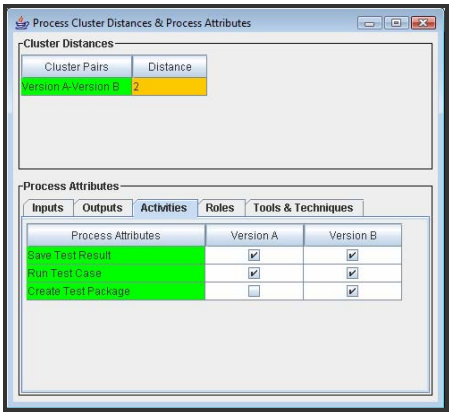


Figure 46 Process Cluster Distances & Process Attributes

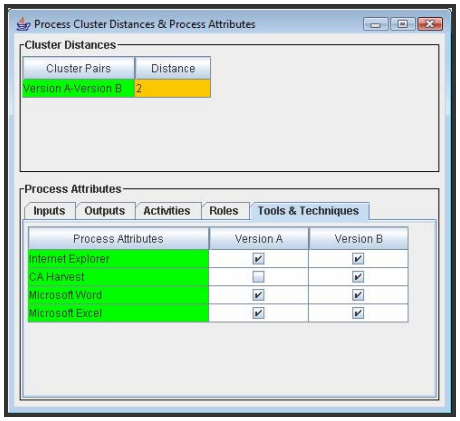
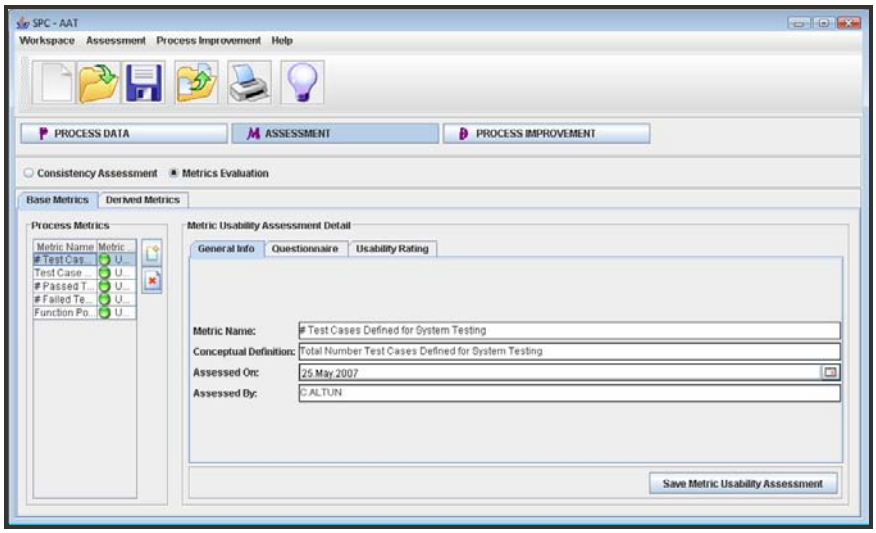


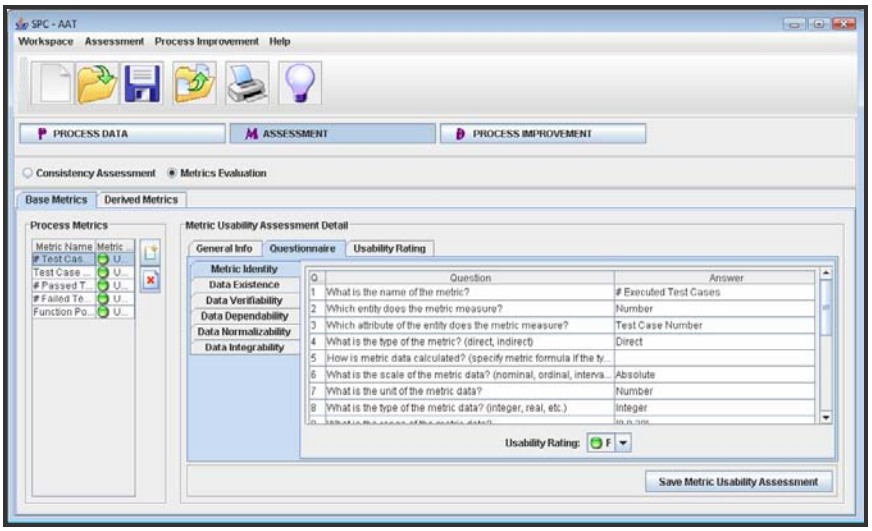
Figure 47 Process Cluster Distances & Process Attributes

After identification of initial process clusters, process metrics were utilized to evaluate their usability for statistical analysis. Number of test cases defined, test case execution time, number of passed test cases, number of failed test cases, and functional size as base metrics were identified. These were the metrics for which data was available on the tool. From the base metrics, test defect density, test effectiveness and test speed derived metrics were identified for the system test process.

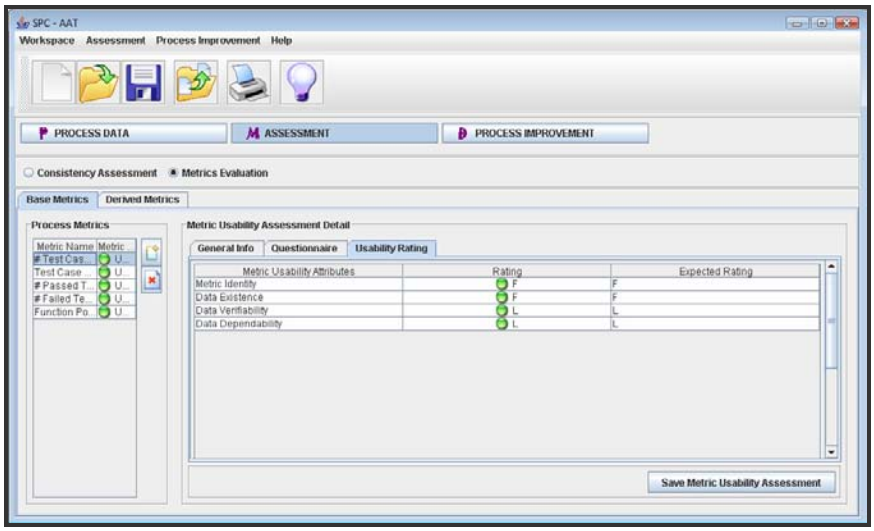
Metric Usability Questionnaires was filled for each base and derived metric from Questionnaire tab-sheet under Metric Evaluation view. Example questionnaire for "Number of Test Cases Defined" base metric with its info, questionnaire and usability ratings given are shown in Figure 48 (completed questionnaires for all metrics identified in Case Study B are provided in Appendix C).



(a) General Info Tab



(b) Questionnaire Tab



(c) Usability Rating Tab

Figure 48 Metric Usability Questionnaire and Rating for Total Number of Test Cases Defined for System Testing Process

The usability status of all base and derived metrics are listed in Figure 49. All the metrics defined at the beginning of this case are usable for the path coverage method for system testing process and usability states can be seen from metric usability evaluation report.

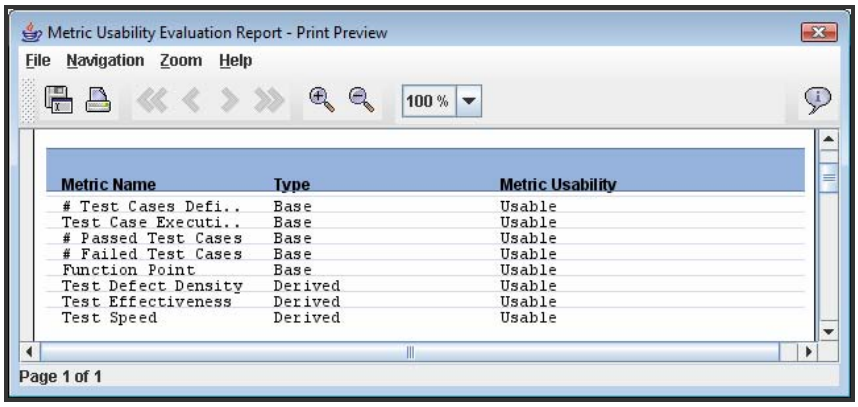
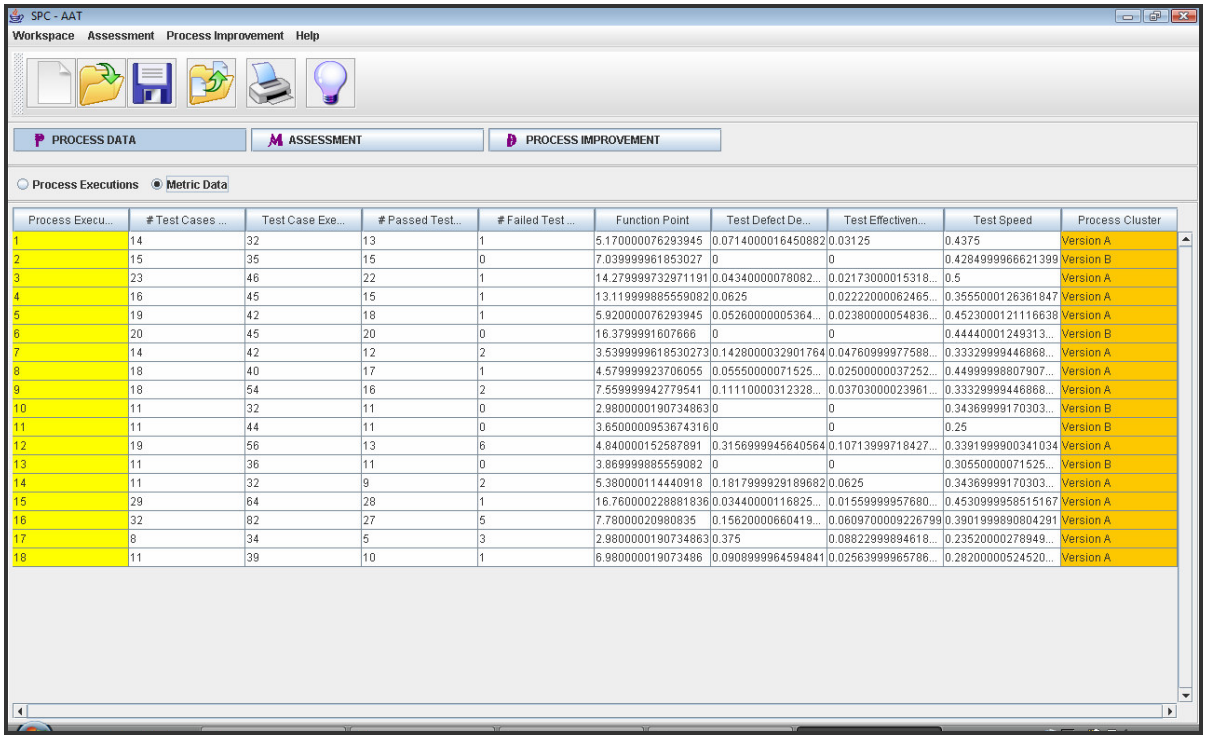


Figure 49 Metric Usability Evaluation Report – Case Study B

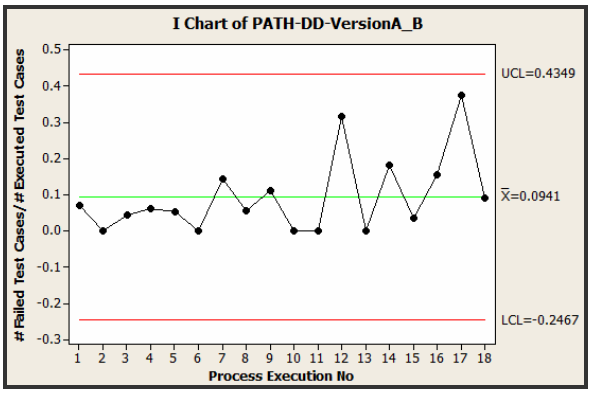
Test data for path coverage method used in Case study B was completed in SPC-AAT as shown in table 50.



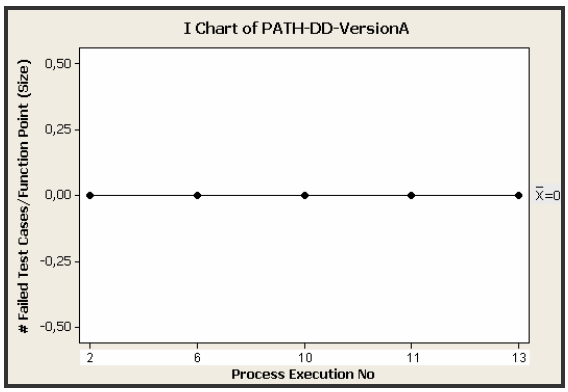
Process Execu...	# Test Cases ...	Test Case Exe...	# Passed Test...	# Failed Test ...	Function Point	Test Defect De...	Test Effectiven...	Test Speed	Process Cluster
1	14	32	13	1	5.170000076293945	0.07140000164508820	0.03125	0.4375	Version A
2	15	35	15	0	7.039999961853027	0	0	0.4284999966621399	Version B
3	23	46	22	1	14.279999732971191	0.04340000078082...	0.02173000015318...	0.5	Version A
4	16	45	15	1	13.119999885559082	0.0625	0.02222000062465...	0.3555000126361847	Version A
5	19	42	18	1	5.920000076293945	0.05260000005364...	0.02380000054836...	0.4523000121116638	Version A
6	20	45	20	0	16.3799991607666	0	0	0.44440001249313...	Version B
7	14	42	12	2	3.5399999618530273	0.14280000329017640	0.04760999977588...	0.33329999446868...	Version A
8	18	40	17	1	4.579999923706055	0.05550000071525...	0.02500000037252...	0.44999998807907...	Version A
9	18	54	16	2	7.559999942779541	0.11110000312328...	0.03703000023961...	0.33329999446868...	Version A
10	11	32	11	0	2.98000001907348630	0	0	0.34369999170303...	Version B
11	11	44	11	0	3.650000095367431	0	0	0.25	Version B
12	19	56	13	6	4.840000152587891	0.31569999456405640	0.10713999718427...	0.3391999900341034	Version A
13	11	36	11	0	3.869999885559082	0	0	0.30550000071525...	Version B
14	11	32	9	2	5.380000114440918	0.18179999291896820	0.0625	0.34369999170303...	Version A
15	29	64	28	1	16.760000228881836	0.03440000116825...	0.01559999957680...	0.4530999958515167	Version A
16	32	82	27	5	7.78000020980835	0.15620000660419...	0.06097000092267990	0.3901999890804291	Version A
17	8	34	5	3	2.98000001907348630	0.375	0.08822999894618...	0.23520000278949...	Version A
18	11	39	10	1	6.980000019073486	0.09089999645948410	0.02563999965786...	0.28200000524520...	Version A

Figure 50 Metric Data of Case Study B

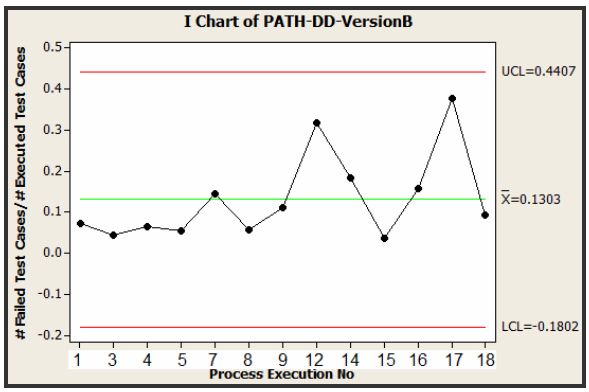
SPC tools were applied to the qualified process cluster – derived metric pairs. In this case (path coverage method), control charts drawn for process cluster – derived metric pairs are shown in the figures 51 to 53.



(a) Defect Density Metric for Version A-B



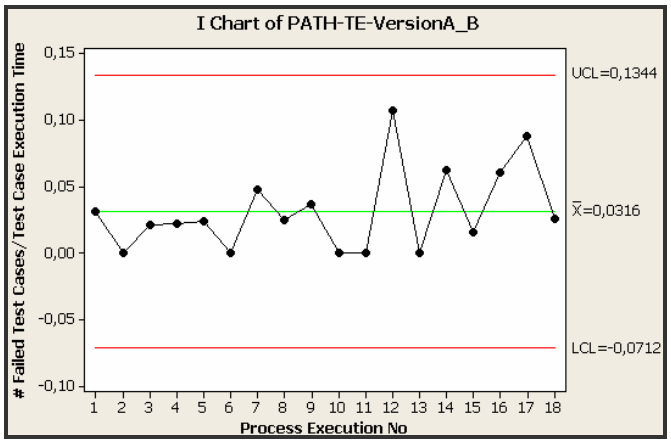
(b) Defect Density Metric for Version A



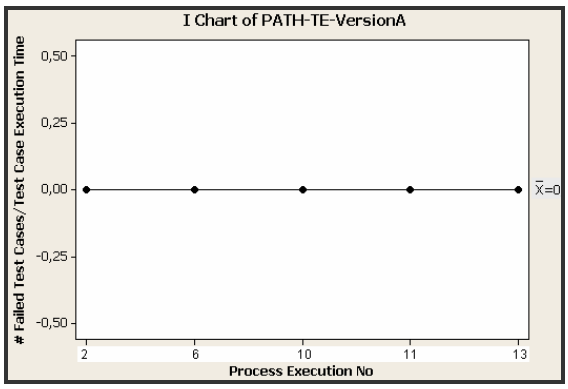
(c) Defect Density Metric for Version B

Figure 51 Individuals Charts for Derived Metrics of Test Defect Density for Path Coverage

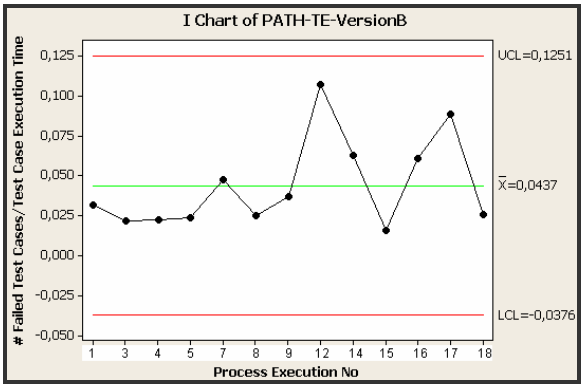
As it can be seen from Figure 51, version A indicates the process executions which do not contain any defected test cases. On the other hand, Version B indicates the process executions which contain defected test cases. Figures 51(a), 51(b) and 51(c) show that clustering does not have remarkable effect on defect density metric when using in path coverage method. It is because figure 51(a), version A-B, and figure 51(c), version B have no out of control points. For path covarage method, test defect density for version A-B, version A and version B is under control and the values are stable can be seen in Figure 51(a), 51(b), 51(c). Mean value of version A is equal to 0 (zero) because of the process executions of version A did not have failed test cases.



(a) Test Effectiveness Metric for Version A-B



(b) Test Eff. Metric for Version A

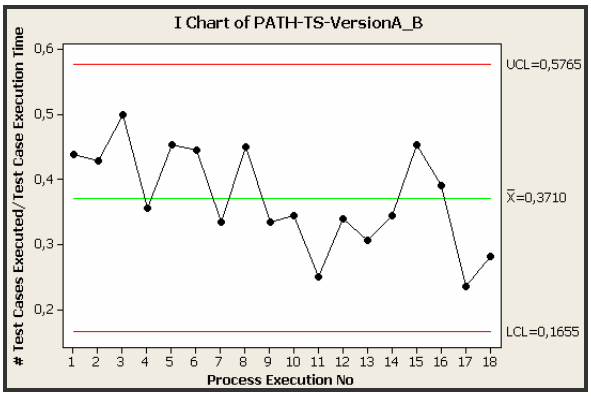


(c) Test Eff. Metric for Version B

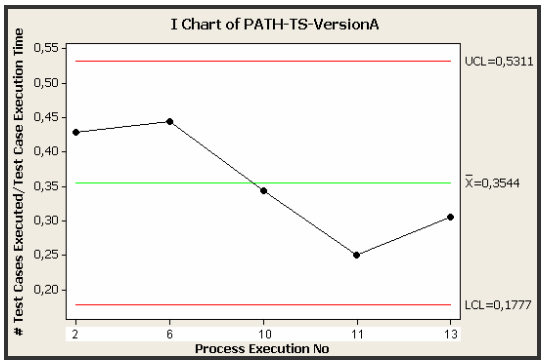
Figure 52 Individuals Charts for Derived Metrics of Test Effectiveness for Path Coverage

When looking at the figure 52(a) version A-B, figure 52(b), version A, and figure 52(c), version B distribution by Test Effectiveness metric, there are no out of control points. Version A (52(b)) indicates the process executions which do not contain any defected test cases. On the other hand, Version B (52(c)) indicates the process executions which contain defected test cases. Clustering here identified and classified process executions on the basis of the similarity of the characteristics they possess. Figures 52(a), 52(b) and 52(c) that clustering does not have remarkable effect on defect density metric when using in path coverage method. Mean value of version A is equal to 0 (zero) because of the process executions of version A did not have failed test cases. Test execution indicates high fail rate at process

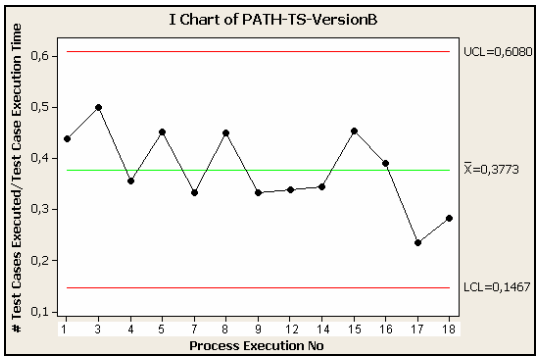
execution #9, #16 and then sharp decrease at operation #12, #17 in result of failed test case number is greater in process execution #12 and #17.



(a) Test Speed Metric for Version A-B



(b) Test Speed Metric for Version A



(c) Test Speed Metric for Version B

Figure 53 Individuals Charts for Derived Metrics of Test Speed for Path Coverage

For node covarage method, test speed metric for all versions is under control can be seen in figures 53(a), 53(b) and 53(c). It can be said that test speed values calculated for this case are stable. The mean value of test speed for version A was lower than the mean value of test speed for version B, because number of test cases defined for version A are higher than number of test cases defined for version B; although number of interfaces are less in version A.

A comparison of control charts of derived metrics between Case A and Case B (that is, between node coverage and path coverage system testing techniques) is provided in the next section.

6 SUMMARY AND CONCLUSIONS

6.1 Discussion on Case Study Results

Establishing control limits on derived metrics provides an organization the ability to predict the metrics that will be inserted into project work products, based on work product size. The use of a standard organizational software development determine readiness to move from one development stage to the next, and to predict future rework costs.

In this section, derived metrics were compared for the versions (A and B) seperately and merged clusters (A-B). One of the objective of this discussion was to understand if merging had positive or negative effects on the clusters. Derived metrics were also compared between node coverage and test coverage testing techniques, because another objective is to derive suggestions on which testing technique would be effective under specific circumstances. Negative values on derived metrics axis were not significant in result of 3 standard deviation of the mean had been implemented.

One of the variable could be utilized in these analyses are coefficient of variation (CV) which is a statistic that tells you how tightly all the various examples are clustered around the mean in a set of data.

Table 4 Comparison of Test Methods by Coefficient of Variation Values on the Basis of Clusters

Derived Metric (CV)	**Node Coverage (CV)**			**Path Coverage (CV)**		
	A_B	**A**	**B**	**A_B**	**A**	**B**
DD	1,6949	0,00	0,7197	1,1454	0,00	0,8176
TE	1,9937	0,00	0,9951	0,9888	0,00	0,6493
TS	0,3803	0,4088	0,3237	0,2074	0,2318	0,2049

In probability theory and statistics, the coefficient of variation (CV) is a measure of dispersion of a probability distribution. It is defined as the ratio of the standard deviation to the mean [34].

The coefficient of variation of the node coverage is greater than coefficient of variation of the path coverage except for test speed derived metric. The standard deviation of a normal distribution is equal to its mean, so its coefficient of variation is equal to mean values. Distributions with CV < μ are considered low-variance, while those with CV > μ are considered high-variance. Test defect density, test effectiveness and test speed derived metrics for both coverage techniques are considered high-variance except test speed derived metric of the path coverage tecnique. It is considered low-variance as can be seen from Table 4.

The other variable utilized in this study is mean (μ) which is the sum of a list of data, divided by the total number of numbers in the data. Data analyzed in this study represents population which contains all data from test cases. If sample data was using, it would be possible to do hypothesis testing with the help of 2-Sample t to analyze the mean values with the standard deviations of this data. As a result, analyzing only mean values is significant for this study because of having population data. The comparison of control charts of derived metrics between Case A and Case B are demonstrated in Table 5.

Table 5 Comparison of Test Methods by Mean Values on the Basis of Clusters

Derived Metric (μ)	**Node Coverage (μ)**			**Path Coverage (μ)**		
	A_B	**A**	**B**	**A_B**	**A**	**B**
Defect Density	0,1260	0,0000	0,3240	0,0941	0,0000	0,1303
Test Effectiveness	0,0337	0,0000	0,0867	0,0316	0,0000	0,0437
Test Speed	0,2837	0,2565	0,3264	0,3710	0,3544	0,3773

According to analysis of table 5, the path coverage and the node coverage methods in basis of clustering, it can be clearly seen that clustering is a good way to analyze with statistical methods based on test defect density, test effectiveness and test speed derived metrics. Therefore it can be stated the case studies that were worked on confirmed the SPC-AM. When looking at the Table 5, mean values of each cluster can be analyzed by means of both test techniques. The result of this cluster analysis is a number of heterogeneous versions with homogeneous contents which means that there are substantial differences between the versions, but the individuals within a single version are similar. Firstly, if defect density derived metric is analyzed for case studies A and B, it can be said that there had been found out more defects by doing system testing with node coverage technique than path coverage technique. Secondly, test effectiveness of node coverage is more than test effectiveness of path coverage. It can be said that node coverage is more effective technique than path coverage by looking at the mean values. Thirdly, by means of test speed derived metric, although number of test cases defined for path coverage technique are more than number of test cases defined for node coverage technique, test speed mean value of path coverage technique is greater than test speed mean value of node coverage technique. Using path coverage technique is less time consuming when performing the system testing.

6.2 Summary and Conclusion

Statistical Process Control (SPC) aims at quality improvement through reduction of variation. The best known tool of SPC is the control chart. After many experinces, the control charts have turned up to be a successful practical technique for monitoring process measurements.

A prospective study had been recently initiated on qualification test process for two case studies in this study. This study was performed to help the improvement of prospective studies will better capture information of process executions and data. For each of the case studies, different versions of processes (process clusters) were identified, evaluated the usability of process metrics and performed SPC analysis for the suitable process clusters and metrics. Node coverage method and path coverage method were utilized of a project at the same organization. The

organization at which case studies were performed is a software development organization having CMMI L3. In the first case, utilization of node coverage, base and derived metrics of system testing process were investigated. In the second case, utilization of path coverage and same metrics of node coverage process of the project were investigated. It was observed that the identification of process clusters is closely related to the purpose of quantitative analysis. In this study, the purpose was to understand qualification test process performance and the identified clusters were merged in such a way that the there will be no difference in testing practices in execution.

In this study, SPC - AM was used in order to test the suitability of SPC for the qualification test process and the metrics. With the help of Statistical Software tool (MINITAB), refining the product quality, improving process capability and managing projects have become pretty easy to control. The SPC–AM simply describes the way of understanding the context for identifying samples of process executions, identifying metrics for statistical analysis and also for the generated process data.

There were number of constraints related to the case studies and their applications. The first one was number of process executions for both case studies were less than the expectation of the assessment model. The clusters were merged to utilize these process executions. This was helpful to understand the benefit of merging these process executions. Second, however there were nearly 200 data points for each of case studies, metrics could just utilized on the interface based except the “test execution time” base metric. After having collected, test execution time was summed for each interface.

For qualification test (case study-A and case study-B), process clusters were identified for each of these two cases and all process metrics were evaluated as “largely usable” for statistical analysis. After control charting the data, it is observed that process clusters were under control with respect to the derived metrics of software test process for both case studies. If there is not adequate time to test all of the nodes that are covered by the test cases, path coverage method is a better way to find the failed functions in the system. Besides, second and third passes of these test cases should be evaluated to make consistent analyses for node coverage model.

According to analysis of table 5, the path coverage and the node coverage methods by mean values in basis of clustering, it can be clearly seen that clustering is a good way to analyze with statistical methods based on test defect density, test effectiveness and test speed metrics. Therefore it can be stated the case studies that were worked on confirmed the SPC-AM. The suggestions of this study are: 1. More test cases should be written to find out clearer analyses results and 2. Path coverage method for system testing should be preferred if there is time constraint on the system testing phase.

MINITAB Statistical Software tool was used to extract control charts in result of SPC-AAT was not beneficial enough to extract control charts in detail of having some information about statistical methods. On the other hand SPC-AAT reduced the time required for statistical analysis by providing a focal point to analyze the metric data besides collecting, organizing and assessing.

SPC-AAT was successful to ease rational sampling process. The attributes of process executions (inputs, outputs, activities, roles, tools & techniques) were entered and SPC-AAT automatically identified the process clusters.

SPC-AAT enhanced defining derived metrics and reduced the time required for calculation. Defining new derived metrics by using existing base or derived metrics was easy. New metrics were defined by just typing the name and the formula of the new derived metric and SPC-AAT calculated metric values for all process executions automatically.

It is obvious that the use of SPC (control charts) and other statistical methods can easily and effectively be used in a software testing in case studies implemented in this study. SPC can identify undesirable trends and can point out fixable problems and potential process improvements. Control charts can show the capability of the process, so achievable goals can be set. They can provide evidence of process stability, which can justify predicting process performance. SPC analysis can provide valuable information used in defect prevention and for lessons learned. SPC is relatively new to software development but after working on this study our observation is that SPC can support software process improvement and improve the quality of software products.

REFERENCES

[1] CAIVANO, D., Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on Volume , Issue , 21-23 March 2005 Page(s): 288 – 293.

[2] ROBESON, K., PROBERT, R. L., CHEN, Y., Effective Test Metrics for Test Strategy Evolution.

[3] OLSSON, F., LUDBERG, H., 2003. Automated Module Testing of Embedded Software Systems. Lund University.

[4] SEDIGH, S., GHAFOOR, A., GHAFOOR, A., 2002. Temporal Modeling of Software Test Coverage. 26 th Annual International Computer Software and Applications Conference.

[5] DUMKE, R., BRAUNGARTEN, R., BLAZEY, M., HEGEWALD, H., REITZ, D., RICHTER, K., 2006. Software Process Measurement and Control. Otto-von-Guericke University.

[6] DEMMY, W. S., PETRINI, A. B., Statistical Process Control In Software Quality Assurance.

[7] WIDARA, M., Data Flow Coverage for Testing Erlang Programs, pp. 151-166.

[8] Wang, Y., King, G., 2000. Software Engineering Processes – Principles and Applications. CRC Press, Boca Raton London New York.

[9] GRESSE, C., HOISLE, B., WÜRST, J., 1995. A process model for GQM-Based Measurement, STTI Report , University of Kaiserslautern, s.229.

[10] LEWIS, R. O., Independent Verification and Validation, IEEE Transactions on Vehicular Technology, vol.40, no.4, s.708-713, 1991.

[11] FLORAC, W. A., PARK, R. E., CARLETON, A. D., 1997. Practical Software Measurement: Measuring for Process Management and Improvement.

[12] PUSALA, R., 2006. Operational Excellence through Efficicient Software Testing Metrics.

[13] YU, Y. T., TANG, S. F., POON, P. L., CHEN, T. Y., 2001. “A study on a path-based strategy for selecting black-box generated test cases, International Journal of Software Engineering and Knowledge Engineering.

[14] KRISHNAMURTHY, S., and MATHUR, A.P., 1997. On the Estimation of Reliability of a Software System Using Reliabilities of its Components, Proceedings 8th International Symposium of Software Reliability Engineering (ISSRE), pp. 146-155.

[15] Carleton, A. D., Paulk, M. C., 1997. Statistical Process Control (SPC) for Software.

[16] Sommerville, I. , 2001. Software Engineering, sixth edition, Pearson Education, Essex.

[17] 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.

[18] FENTON, N. E., 1991. Software Metrics: A Rigorous Approach. London: Chapman & Hall.

[19] http://www.micquality.com/six_sigma_glossary/measurement_scales.htm

[20] OLSSON, T., RUNESON, P., 2001. V-GQM: A Feed-Back Approach to Validation of a GQM Study, Metrics ’01 - International Software Metrics Symposium.

[21] PARK, R. E., GOETHERT, W. B., FLORAC, W. A., 1996. Goal Driven Software Measurement, Software Engineering Institute, Carnegie Mellon University.

[22] Montgomery, D. C., Introduction to Statistical Quality Control, 4th Edition. Arizona State University.

[23] JALOTE, P., SAXENA, A, 2002. Optimum Control Limits for Employing Statistical Process Control in Software Process. IEEE Transactions on Software Engineering Volume 28, s: 1126 – 1134.

[24] FLORAC, W. A., CARLETON, A. D., 1999. Measuring the Software Process – Statistical Process Conrol for Software Process Improvement.

[25] LIPKE, W., 2002. "Statistical Process Control of Project Performance." CrossTalk.

[26] PIWOWARSKI, P., MITSURU O., CARUSO, J., Coverage Measurement Experience During Function Test, International Business Machines Corporation.

[27] BRIAND, L.C., LABICHE, Y., 2004. Improving State-Based Coverage Criteria Using Data Flow Information, Carleton University.

[28] KANER, C., 2001. Measurement Issues and Software Testing. Florida Instıtude of Technology.

[29] NIESSINK, F., VLIET, H., 2001. Measurement Program Success Factors Revisited. Information and Software Technology, Volume 43.

[30] TARHAN ,A., DEMİRÖRS, O., Assessment of Software Process and Metrics to Support Quantitative Understanding.

[31] TARHAN ,A., An assessment model for the applicability of statistical process control for software processes, 2006.

[32] KIRBAŞ, S., TARHAN, A., DEMİRÖRS, O., An Assessment and Analysis Tool for Statistical Process Control of Software Processes.

[33] KIRBAŞ, S., An assessment and analysis tool for statistical process control of software processes, 2007.

[34] http://en.wikipedia.org/wiki/Coefficient_of_variation

APPENDICES

A. SPC-AM ASSETS

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
1		

2. Outputs: Please list the outputs from the process execution.

No	Name	Description
1		
2		

3. Activities: Please list in sequence the activities that were performed while executing the process.

No	Name	Description
1		
2		
3		
4		

4. Roles: Please list the roles that were allocated responsibilities in process execution.

No	Name	Description
1		
2		

5. Tools and Techniques: Please list the tools and techniques that are used to support process execution.

No	Name	Description

Figure 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 2 Process Execution Questionnaire

PROCESS SIMILARITY MATRIX

Process Name:
Recorded On:
Recorded By:

		Process Executions																		
Process Attributes		PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8	PE9	PE10	.	.	.	.	.	.	.	.	PEX
1	**Inputs**																			
	1.1																			
2	**Outputs**																			
	2.1																			
3	**Activities**																			
	3.1																			
	Activities in this sequence?																			
4	**Roles**																			
	4.1																			
5	**Tools and Techniques**																			
	5.1																			
Process Cluster																				

* Please verify each process execution against process attributes. Insert an "O" into each cell if applicable (leave blank if not applicable).

Figure 3 Process Similarity Matrix

Metric Name:
Conceptual Definition:
Assessed On:
Assessed By:

Please rate each attribute in four scales, based on asnwers to questions as indicators:
F : Indicatiors of the atribute are fully satisfied (%86-100)
L : Indicatiors of the atribute are largely satisfied (%51-85)
P : Indicatiors of the atribute are largely satisfied (%16-50)
N : Indicatiors of the atribute are not satisfied (%0-15)

Attributes			Answers	Rating	Expected Answers
	Indicators				
Metric Identity			MUF-1	F	
	Q1	Which entity does the metric measure?			
	Q2	Which attribute of the entity does the metric measure?			
	Q3	What is the scale of the metric data? (nominal, ordinal, interval, ratio, absolute)			Ratio, Absolute
	Q4	What is the unit of the metric data?			
	Q5	What is the type of the metric data? (integer, real, etc.)			
	Q6	What is the range of the metric data?			
Data Existence			MUF-2	F	
	Q7	Is metric data existent?			Available > 20
	Q8	What is the amount of overall observations?			
	Q9	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			MUF-3	F	
	Q12	When is metric data recorded in the process? (at start, middle, end, later, etc.)			
	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?			
	Q15	Is all metric data recorded by the responsible body?			Yes
	Q16	How is metric data recorded? (on a form, report, tool, etc.)			
	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.)			
	Q19	Is all metric data stored in the same place? (in a file, database, etc.)			Yes
Data Dependability			MUF-4	F	
	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 4 Metric Usability Questionnaire for Base Metrics

				Please rate each attribute in four scales, based on asnwers to questions as indicators:	
Metric Name:				F : Indicatiors of the atribute are fully satisfied (%86-100)	
Conceptual Definition:				L : Indicatiors of the atribute are largely satisfied (%51-85)	
Assessed On:				P : Indicatiors of the atribute are largely satisfied (%16-50)	
Assessed By:				N : Indicatiors of the atribute are not satisfied (%0-15)	
Attributes			**Answers**	**Rating**	Expected Answers
	Indicators				
Metric Identity			**MUF-1**	**F**	
	Q1	What is the 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 5 Metric Usability Questionnaire for Derived Metrics

B. DETAILS OF CASE STUDIES A, B

SPC-AM Assets

Process Execution Records of Case Study A

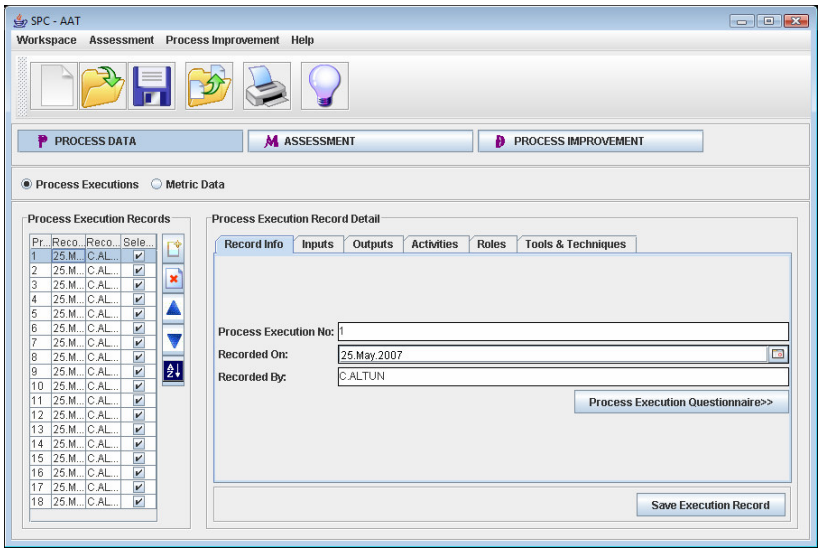


Figure 6 Process Execution Record of Process Execution #1

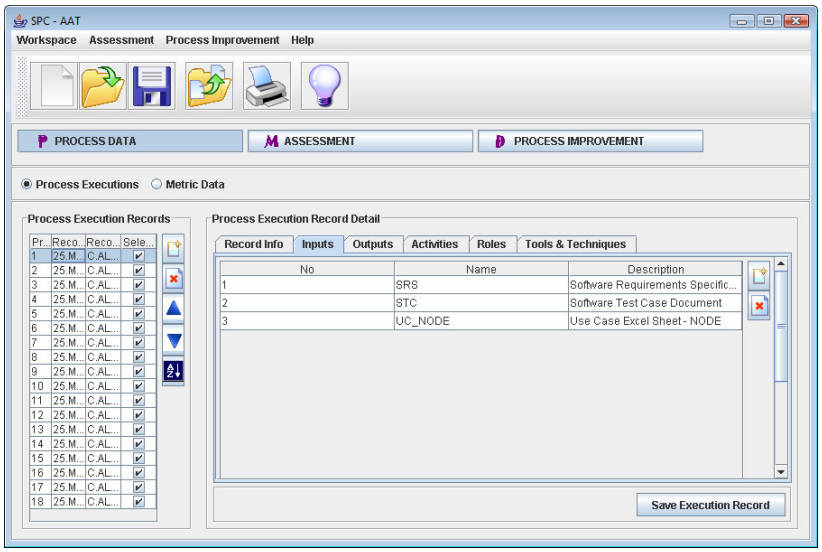


Figure 7 Process Execution Record of Process Execution #1

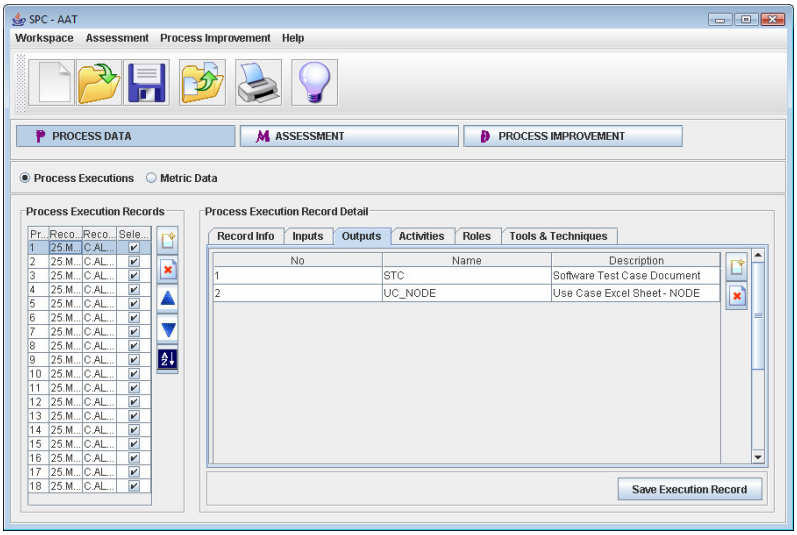


Figure 8 Process Execution Record of Process Execution #1

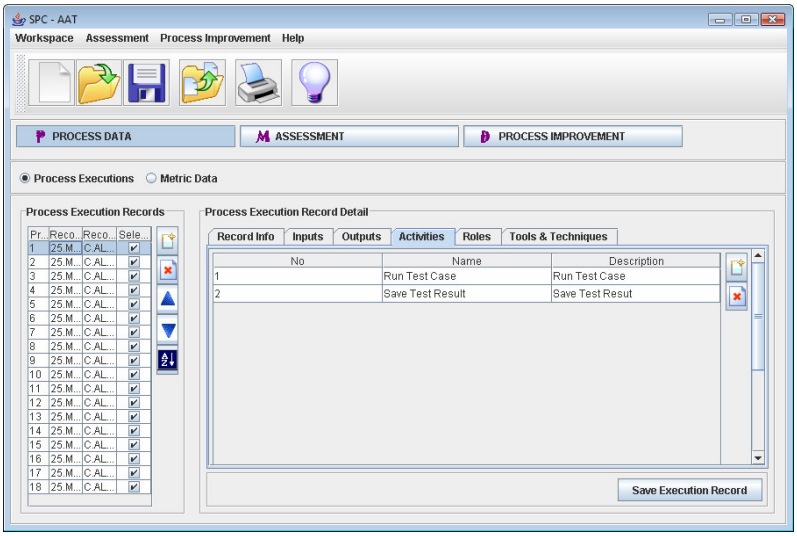


Figure 9 Process Execution Record of Process Execution #1

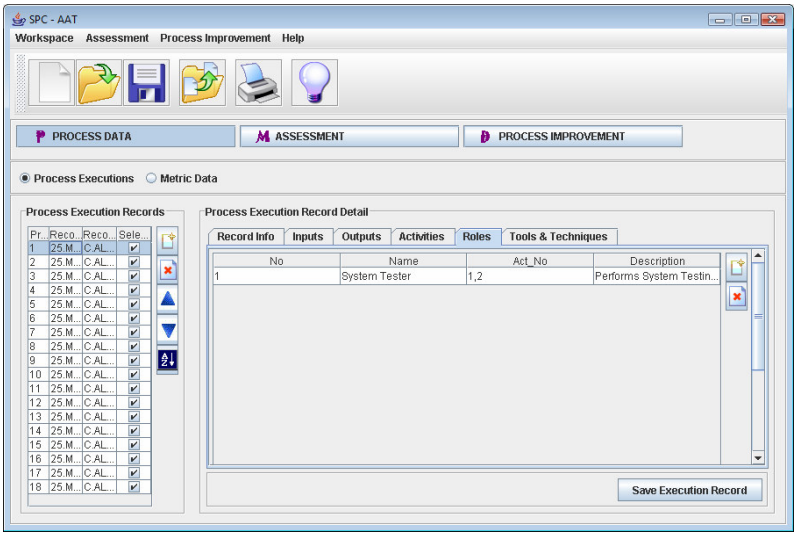


Figure 10 Process Execution Record of Process Execution #1

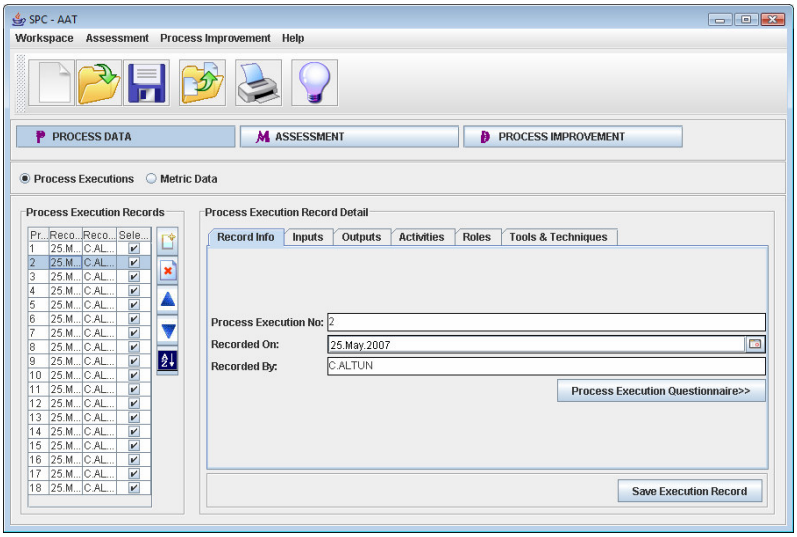


Figure 11 Process Execution Record of Process Execution #2

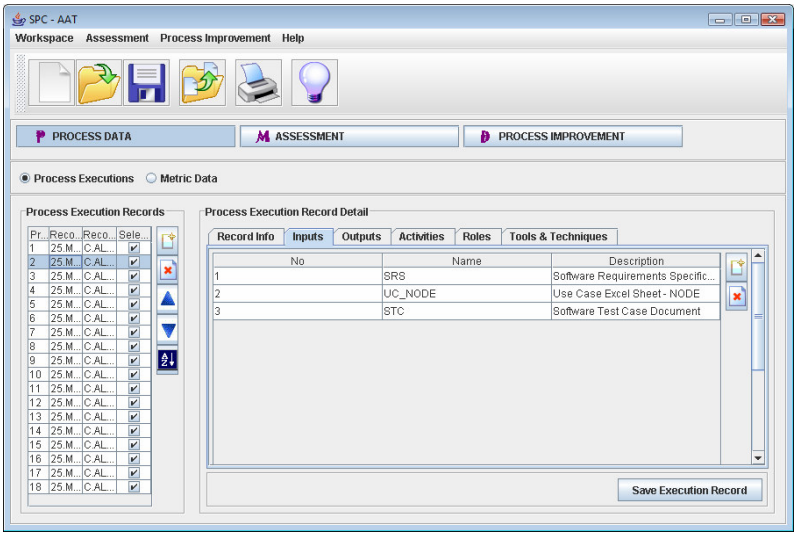


Figure 12 Process Execution Record of Process Execution #2

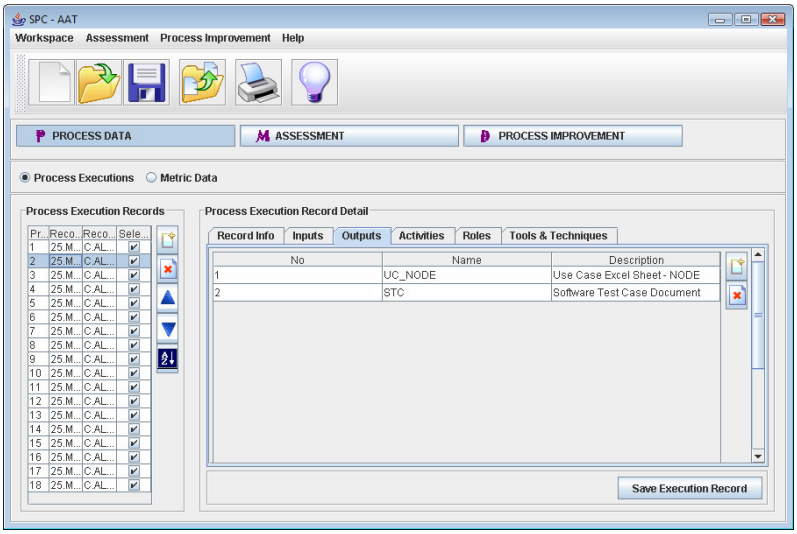


Figure 13 Process Execution Record of Process Execution #2

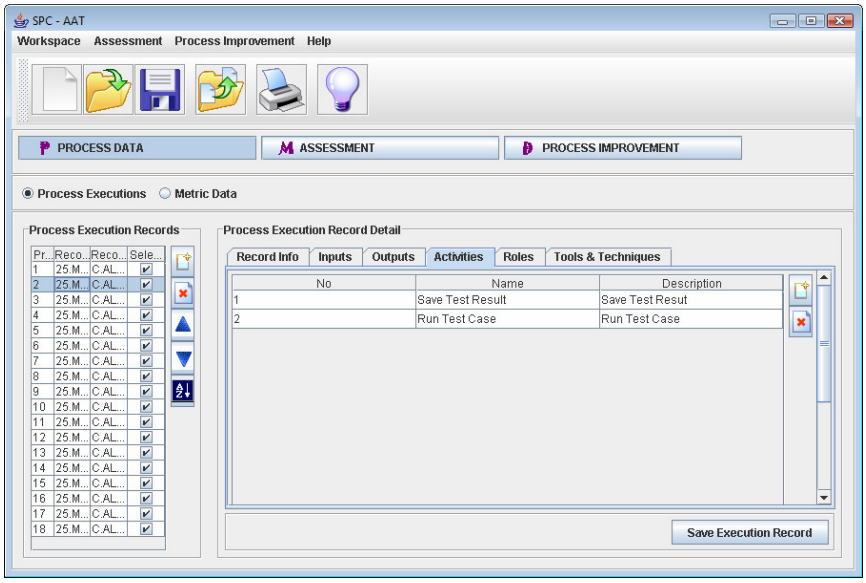


Figure 14 Process Execution Record of Process Execution #2

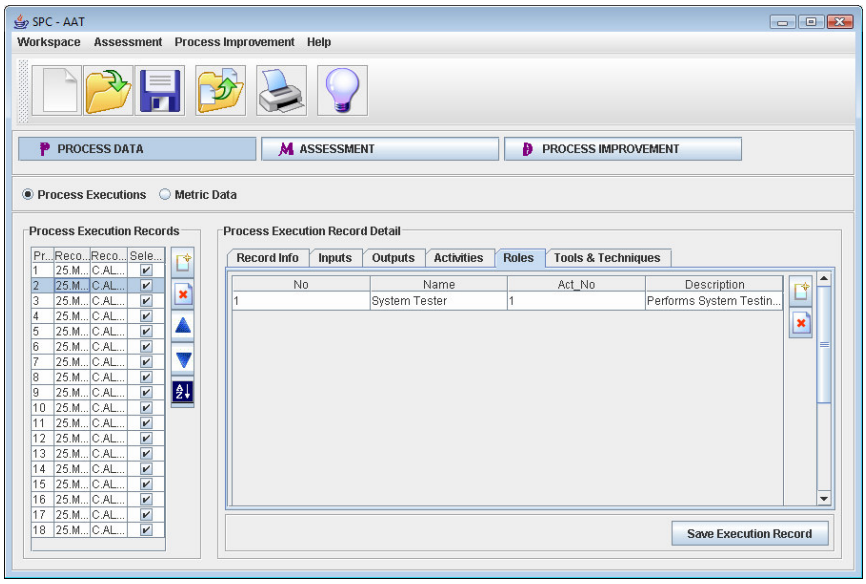


Figure 15 Process Execution Record of Process Execution #2

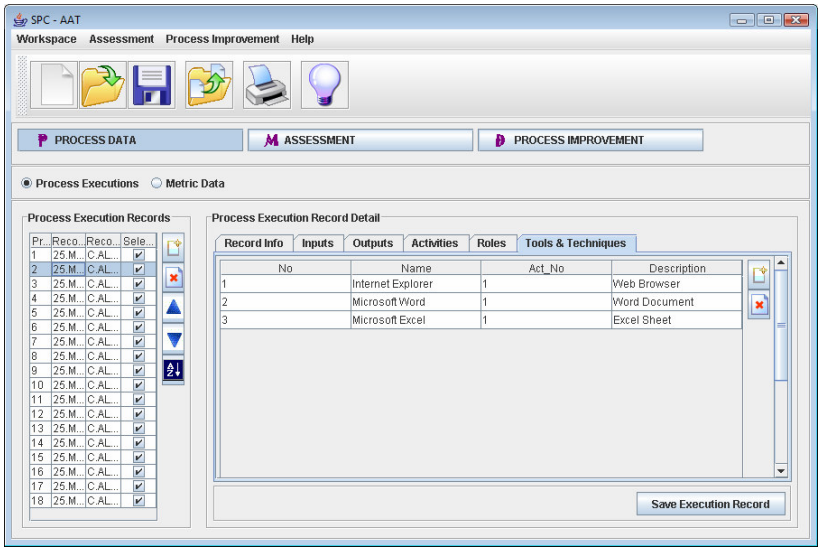


Figure 16 Process Execution Record of Process Execution #2

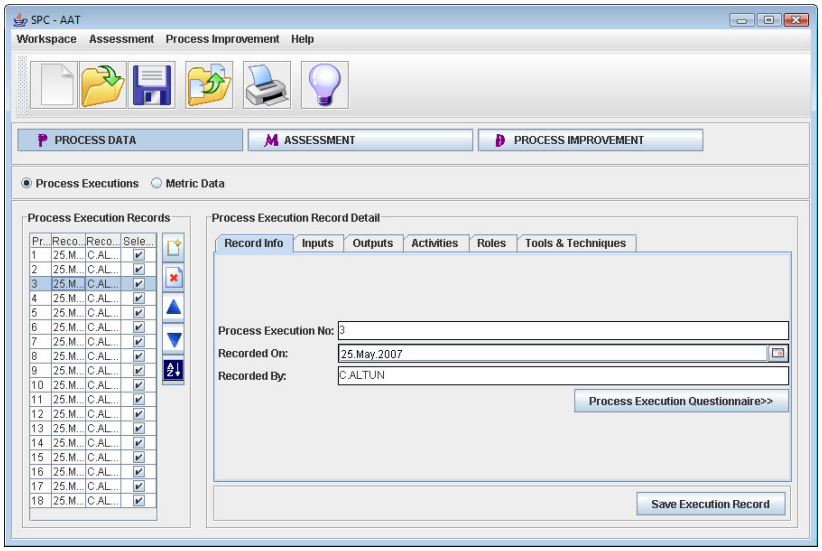


Figure 17 Process Execution Record of Process Execution #3

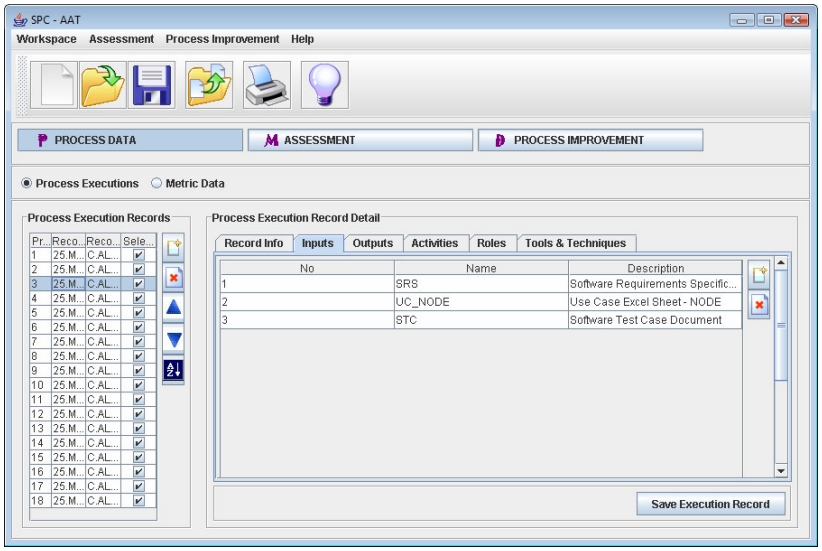


Figure 18 Process Execution Record of Process Execution #3

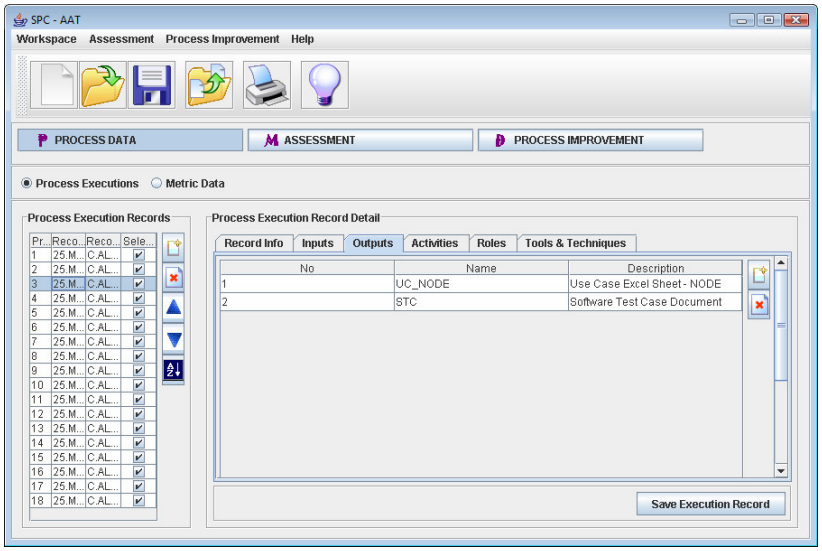


Figure 19 Process Execution Record of Process Execution #3

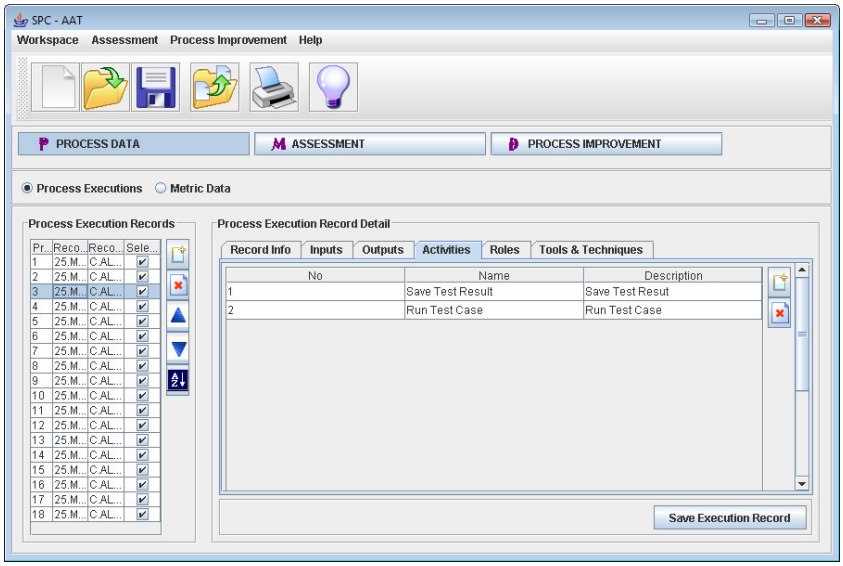


Figure 20 Process Execution Record of Process Execution #3

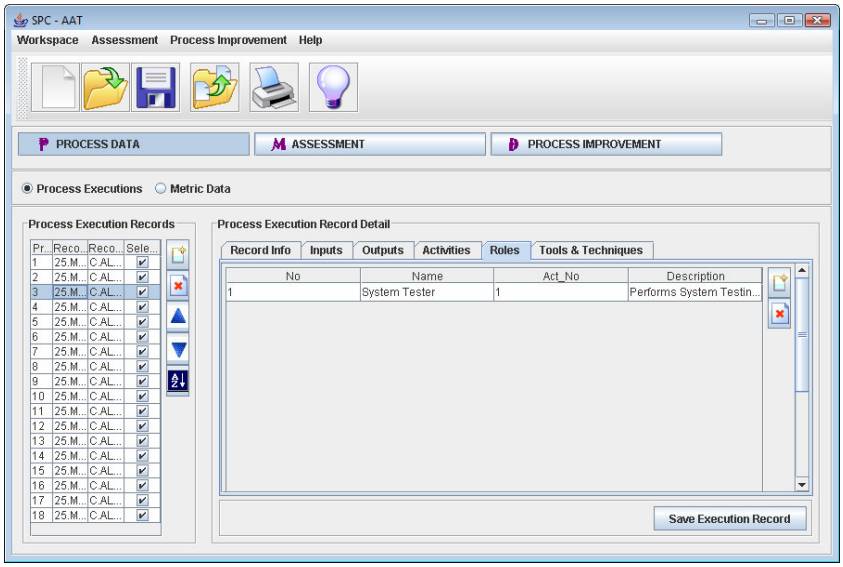


Figure 21 Process Execution Record of Process Execution #3

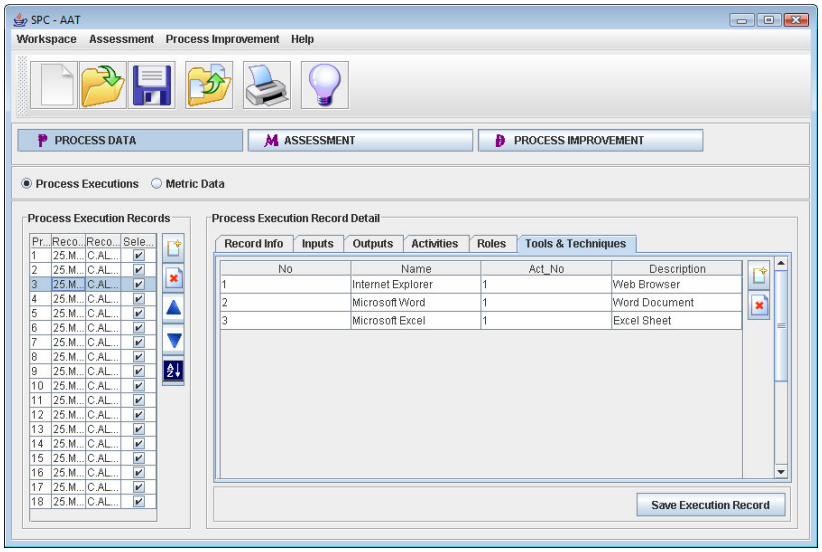


Figure 22 Process Execution Record of Process Execution #3

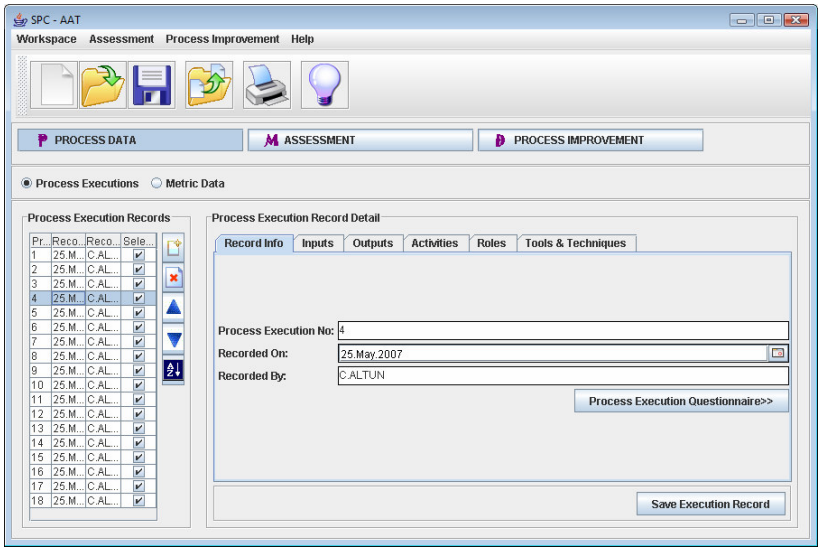


Figure 23 Process Execution Record of Process Execution #4

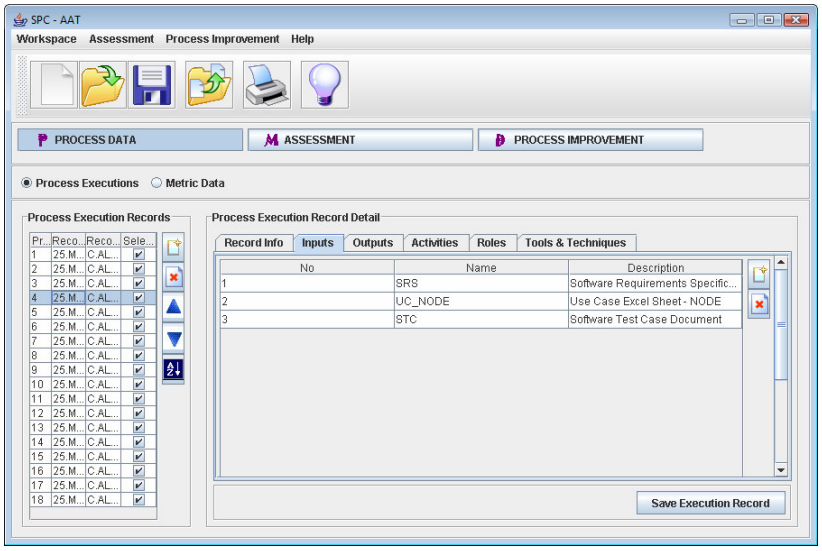


Figure 24 Process Execution Record of Process Execution #4

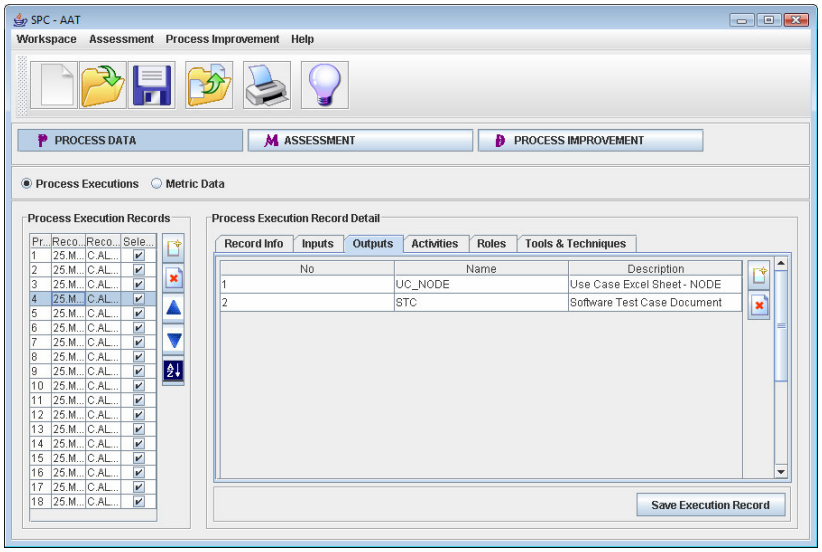


Figure 25 Process Execution Record of Process Execution #4

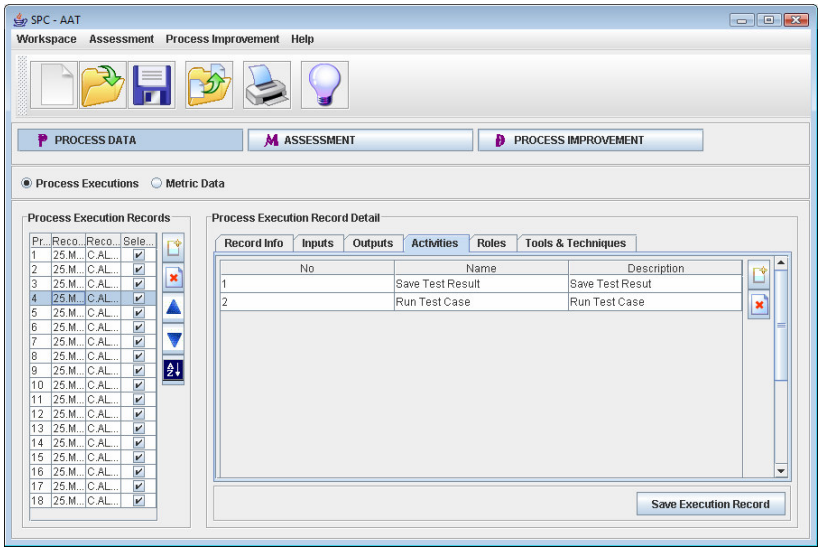


Figure 26 Process Execution Record of Process Execution #4

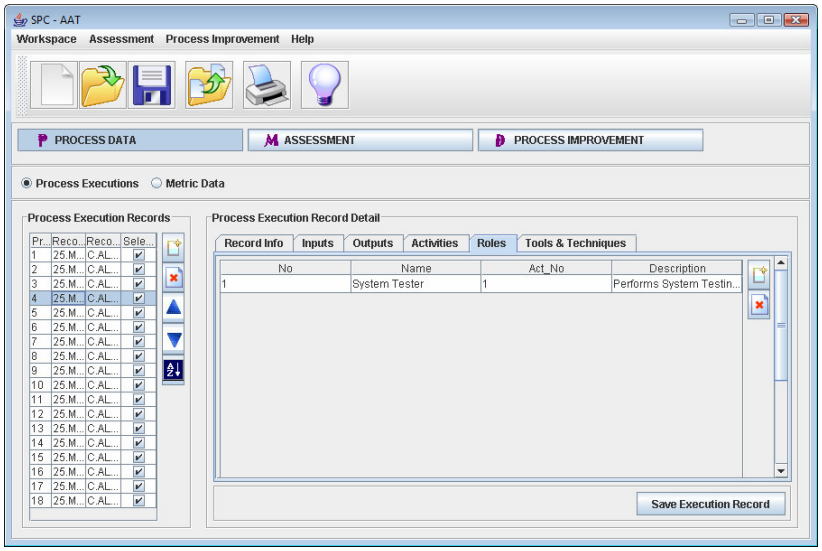


Figure 27 Process Execution Record of Process Execution #4

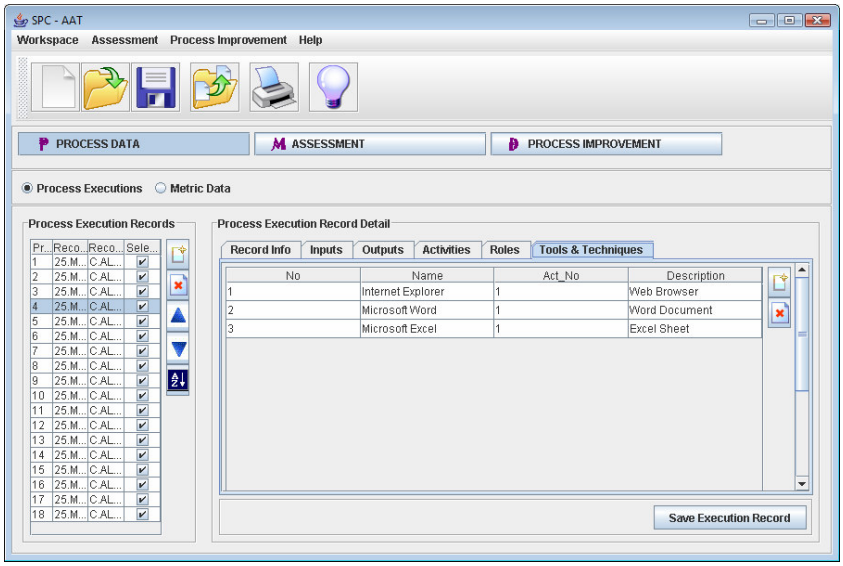


Figure 28 Process Execution Record of Process Execution #4

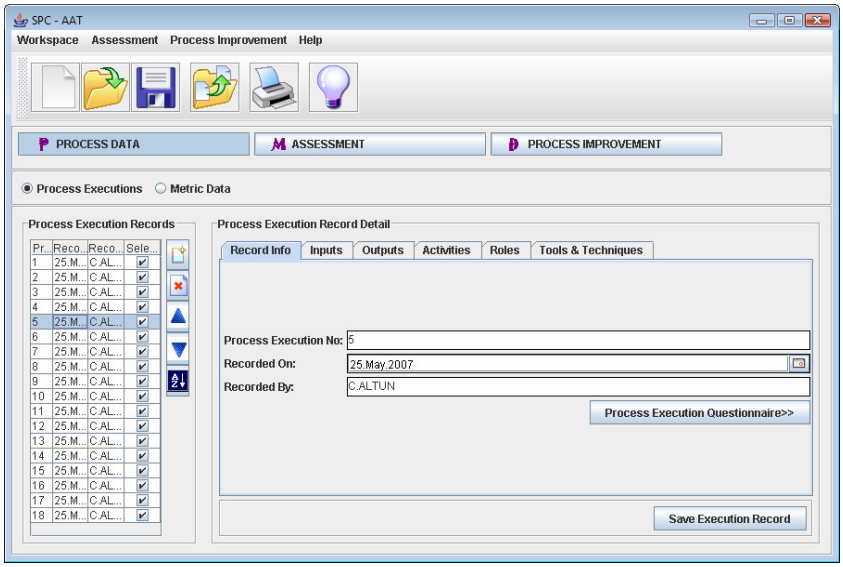


Figure 29 Process Execution Record of Process Execution #5

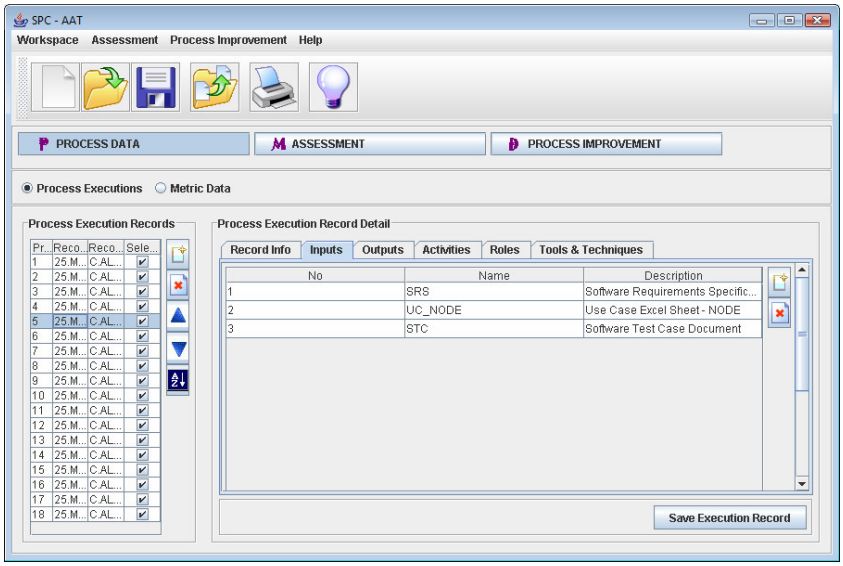


Figure 30 Process Execution Record of Process Execution #5

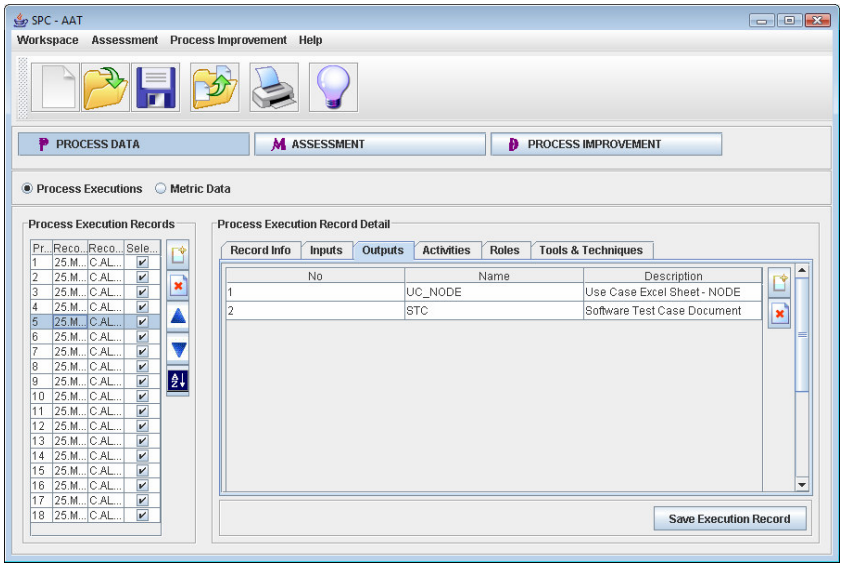


Figure 31 Process Execution Record of Process Execution #5

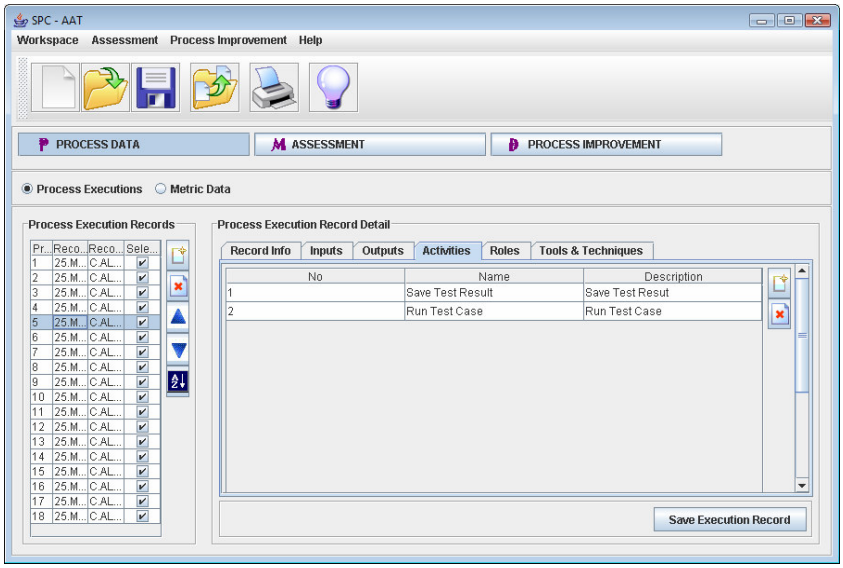


Figure 32 Process Execution Record of Process Execution #5

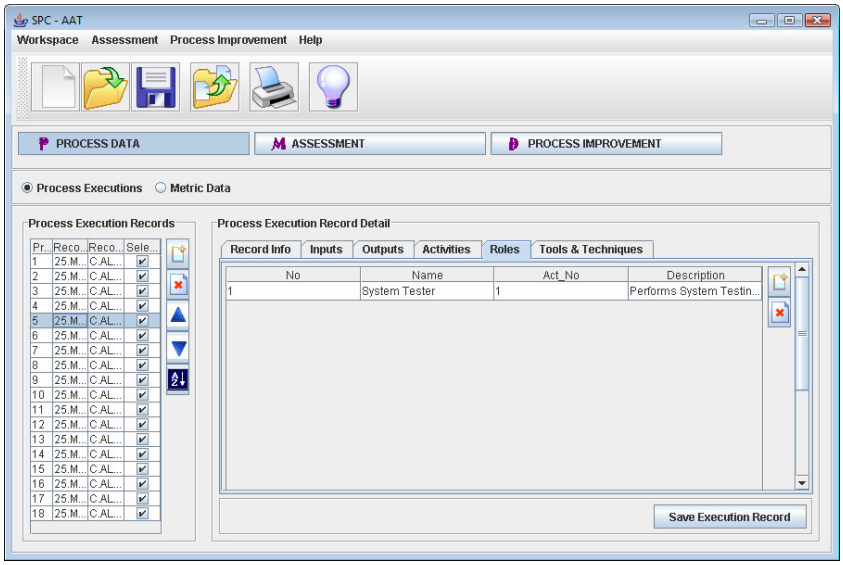


Figure 33 Process Execution Record of Process Execution #5

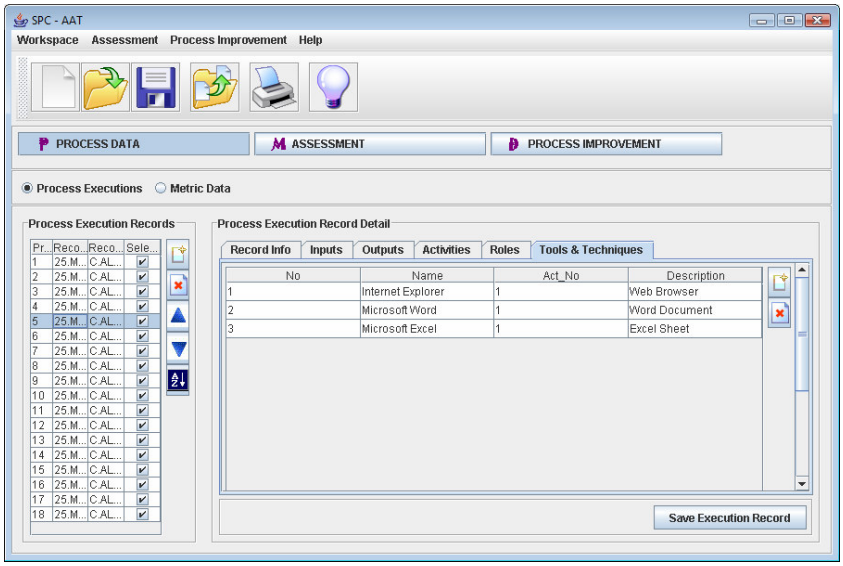


Figure 34 Process Execution Record of Process Execution #5

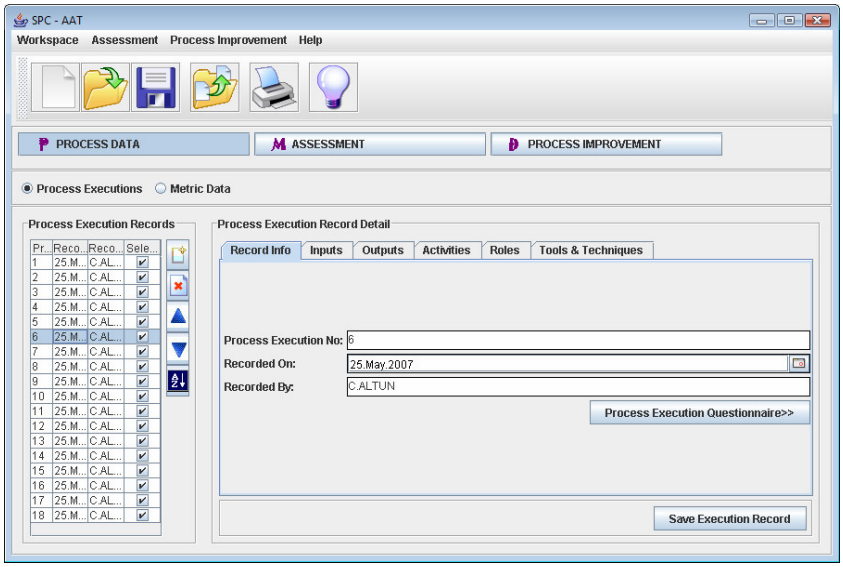


Figure 35 Process Execution Record of Process Execution #6

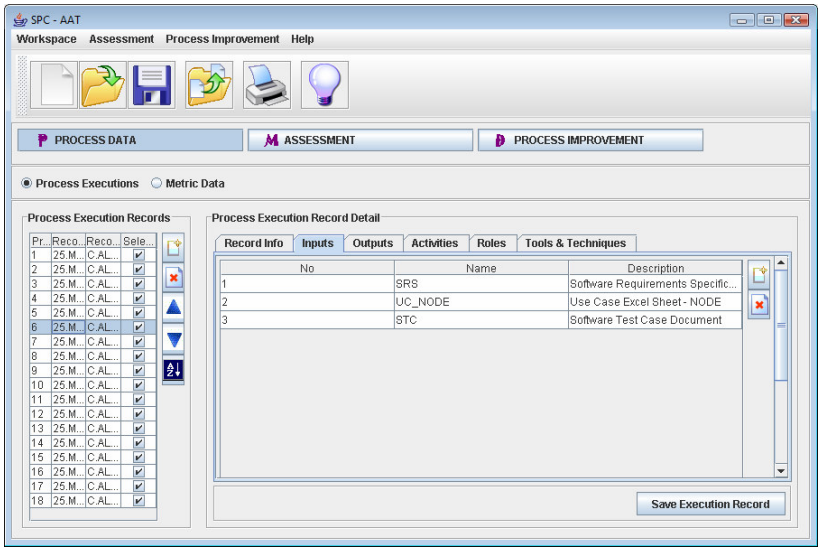


Figure 36 Process Execution Record of Process Execution #6

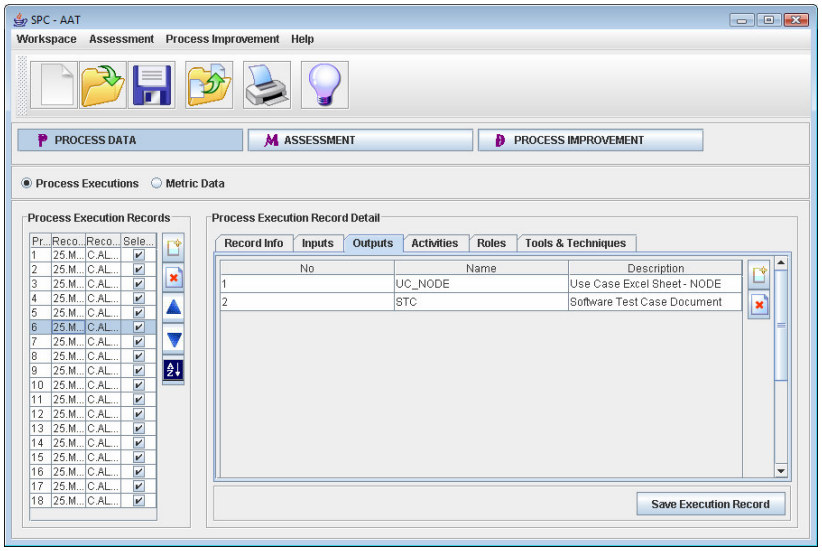


Figure 37 Process Execution Record of Process Execution #6

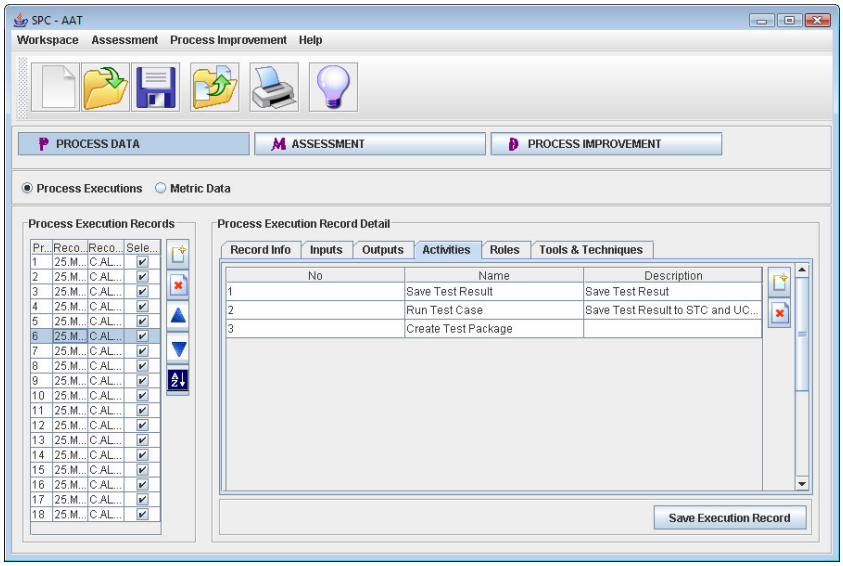


Figure 38 Process Execution Record of Process Execution #6

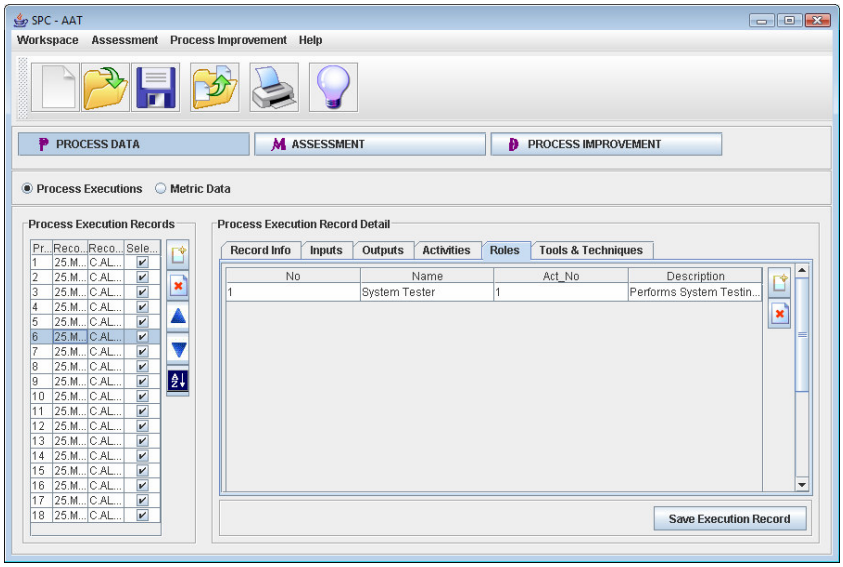


Figure 39 Process Execution Record of Process Execution #6

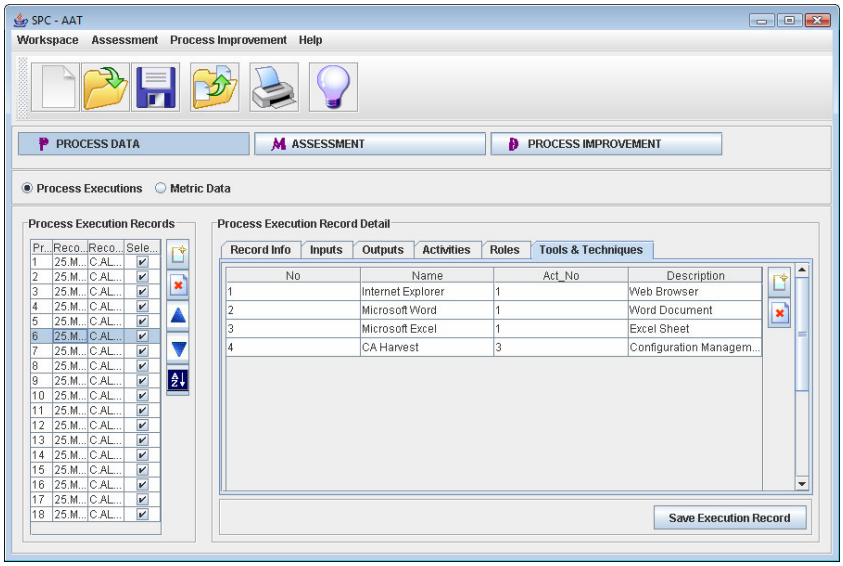


Figure 40 Process Execution Record of Process Execution #6

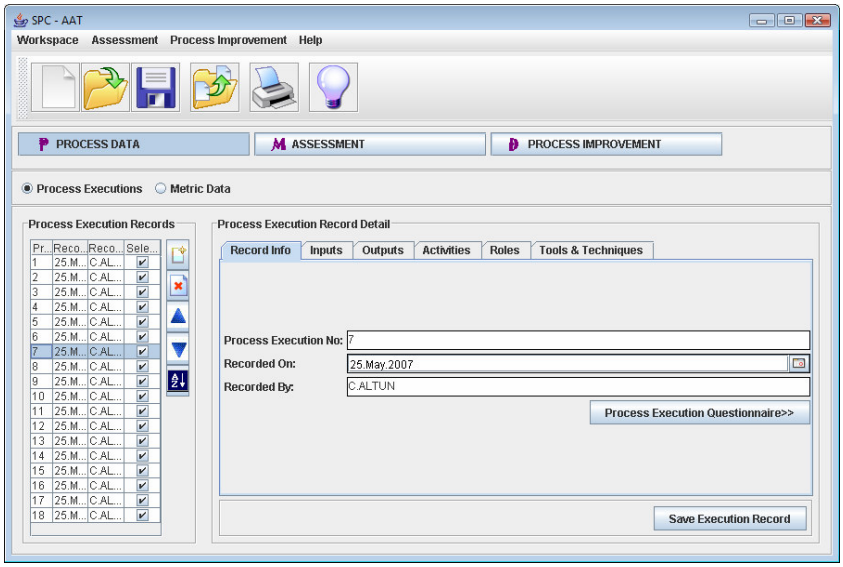


Figure 41 Process Execution Record of Process Execution #7

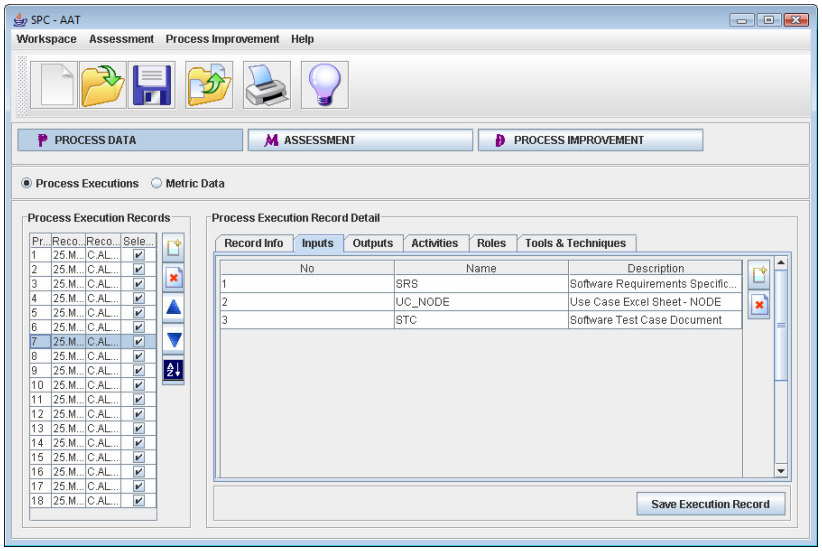


Figure 42 Process Execution Record of Process Execution #7

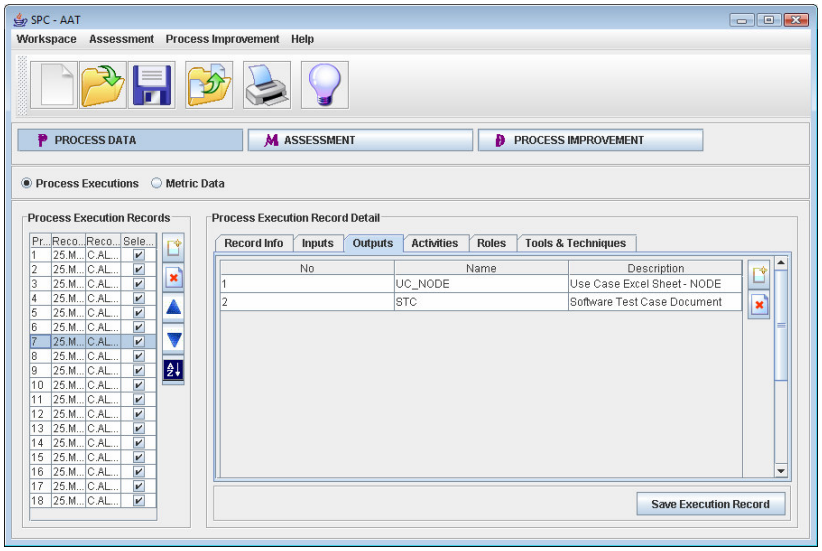


Figure 43 Process Execution Record of Process Execution #7

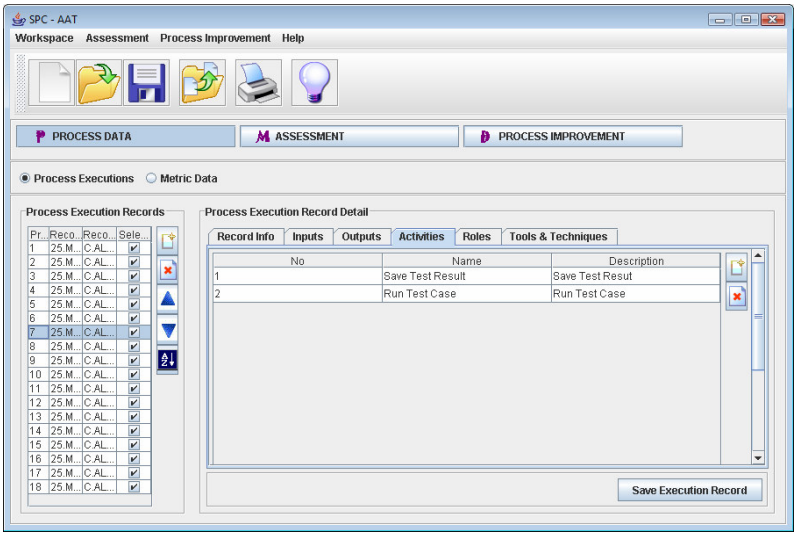


Figure 44 Process Execution Record of Process Execution #7

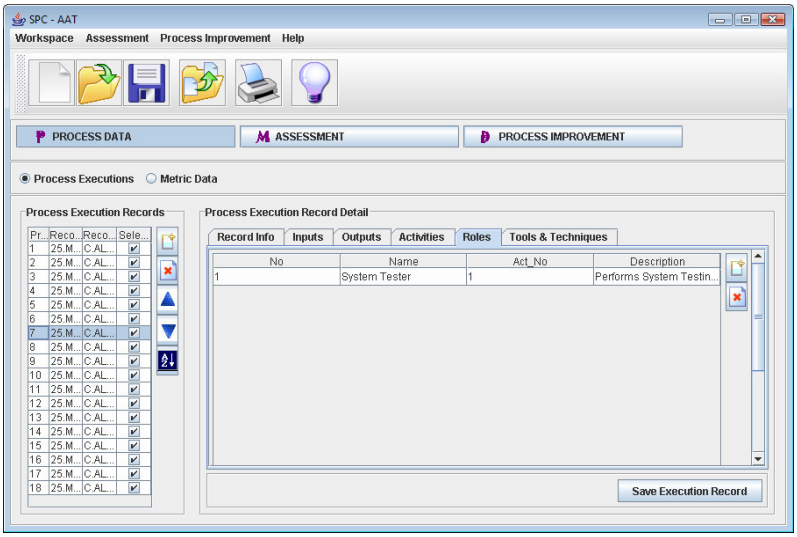


Figure 45 Process Execution Record of Process Execution #7

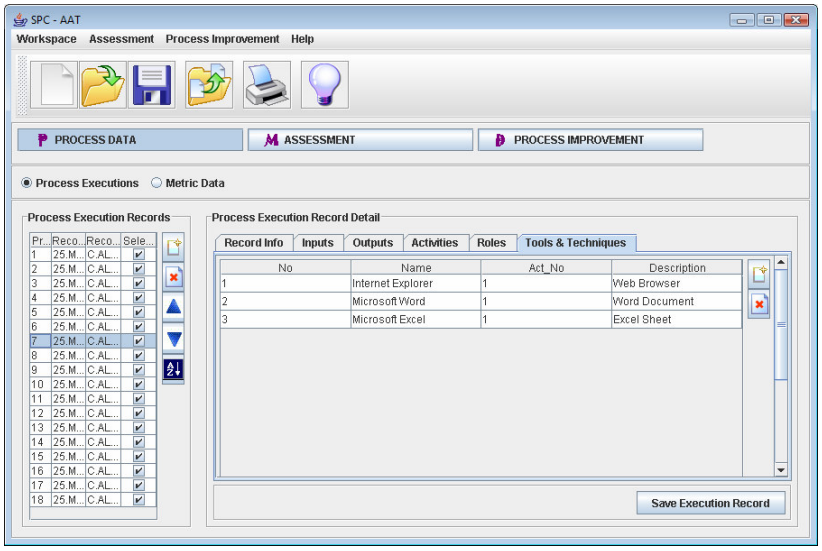


Figure 46 Process Execution Record of Process Execution #7

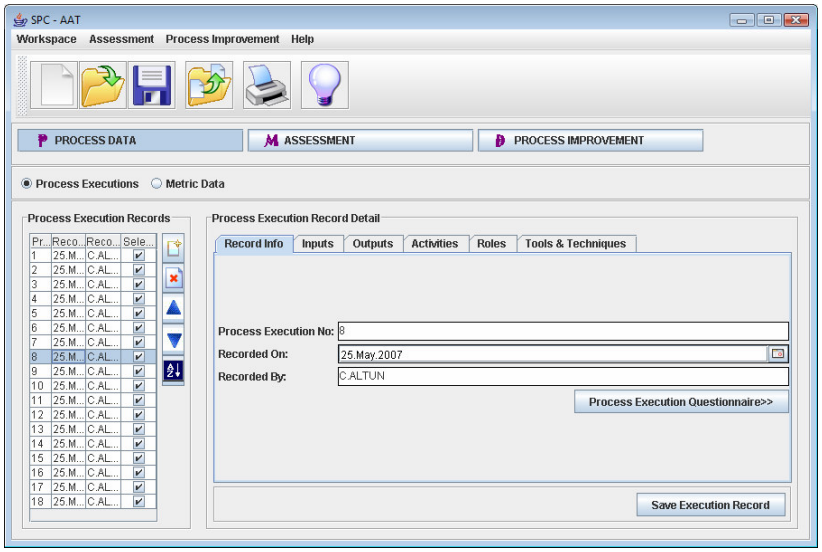


Figure 47 Process Execution Record of Process Execution #8

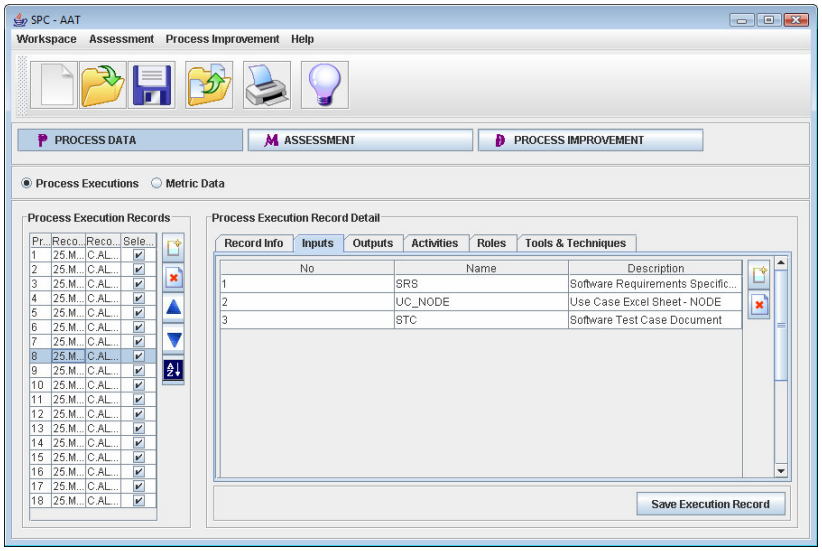


Figure 48 Process Execution Record of Process Execution #8

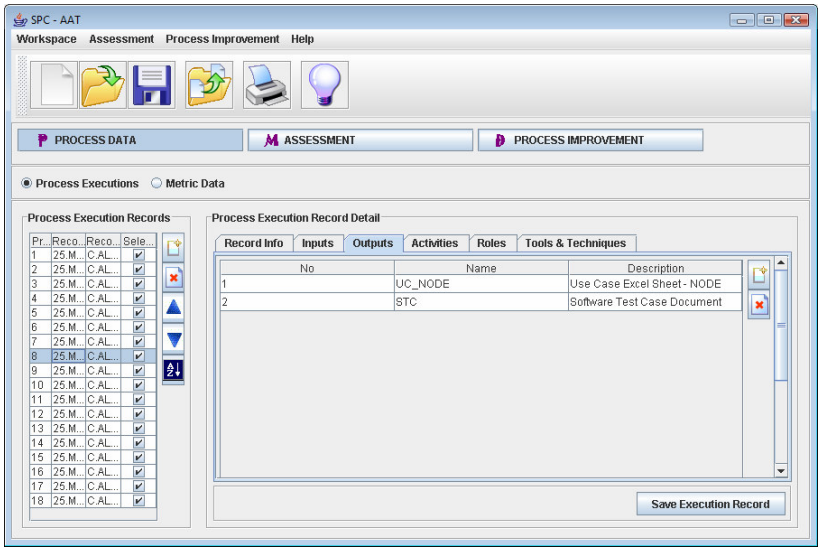


Figure 49 Process Execution Record of Process Execution #8

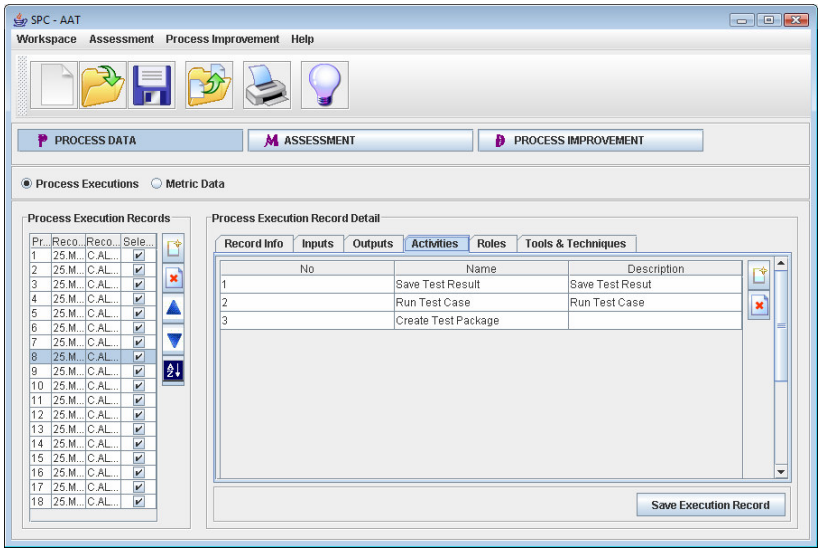


Figure 50 Process Execution Record of Process Execution #8

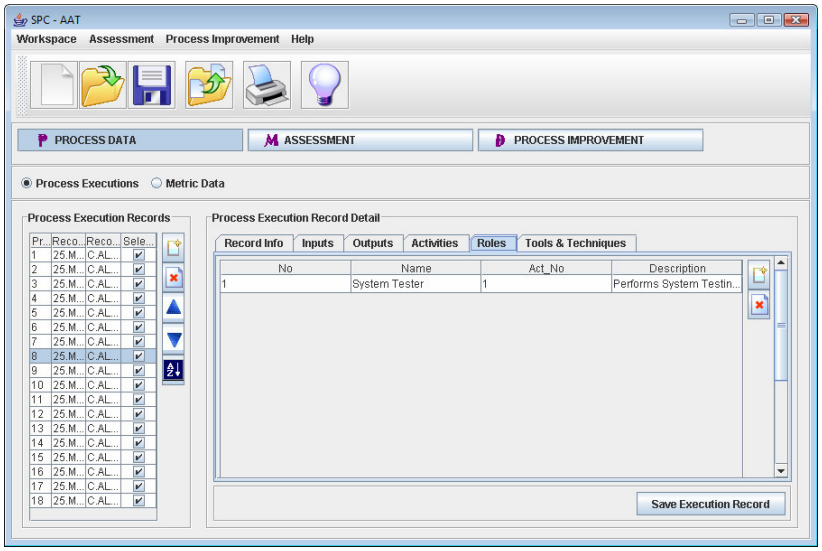


Figure 51 Process Execution Record of Process Execution #8

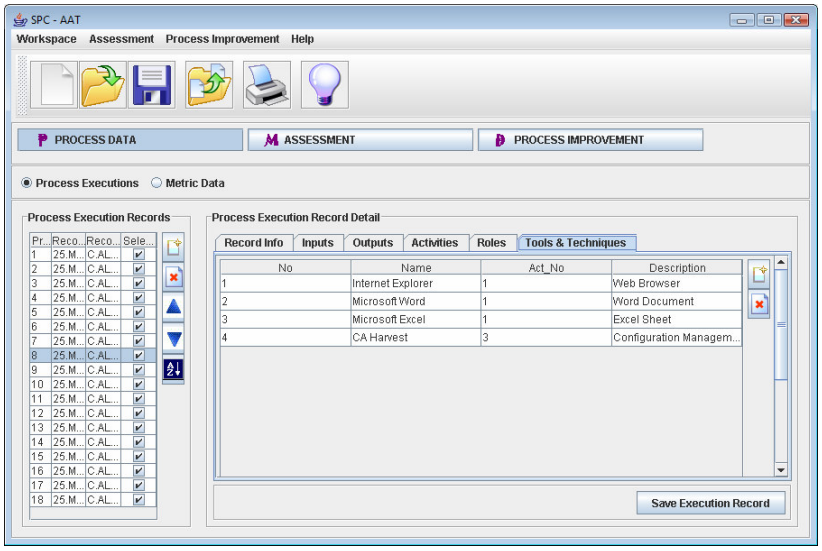


Figure 52 Process Execution Record of Process Execution #8

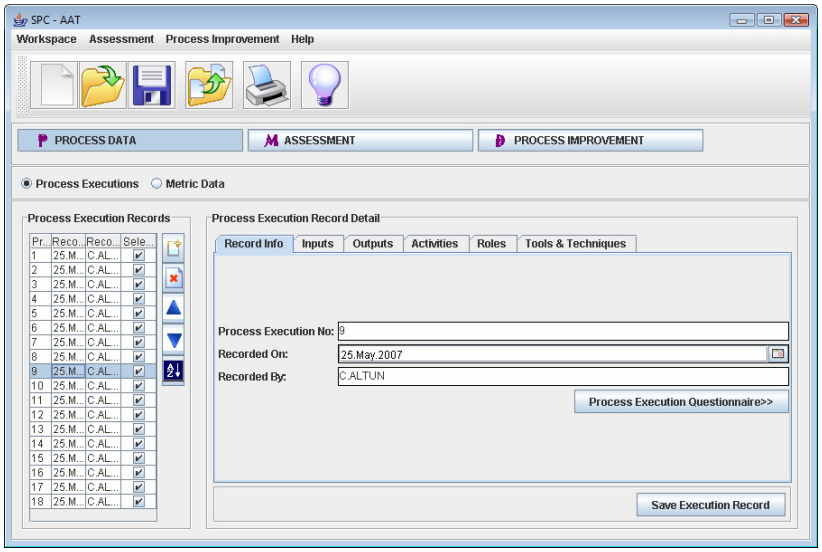


Figure 53 Process Execution Record of Process Execution #9

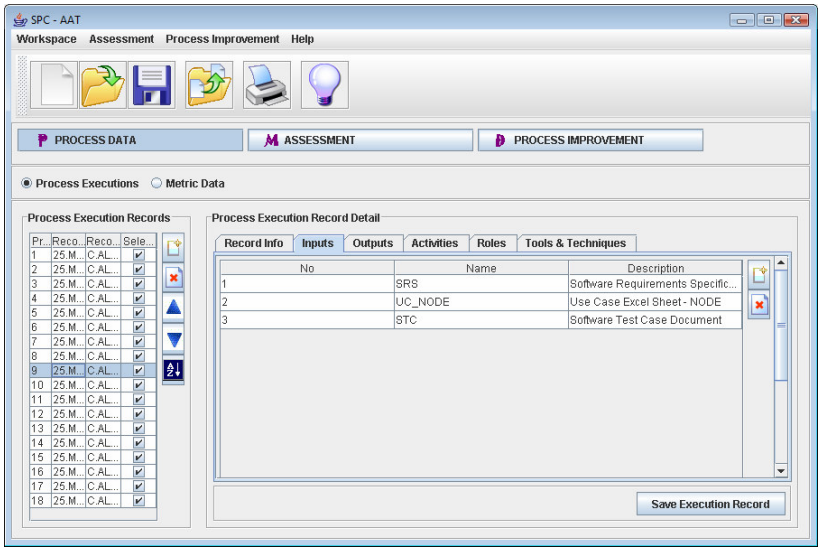


Figure 54 Process Execution Record of Process Execution #9

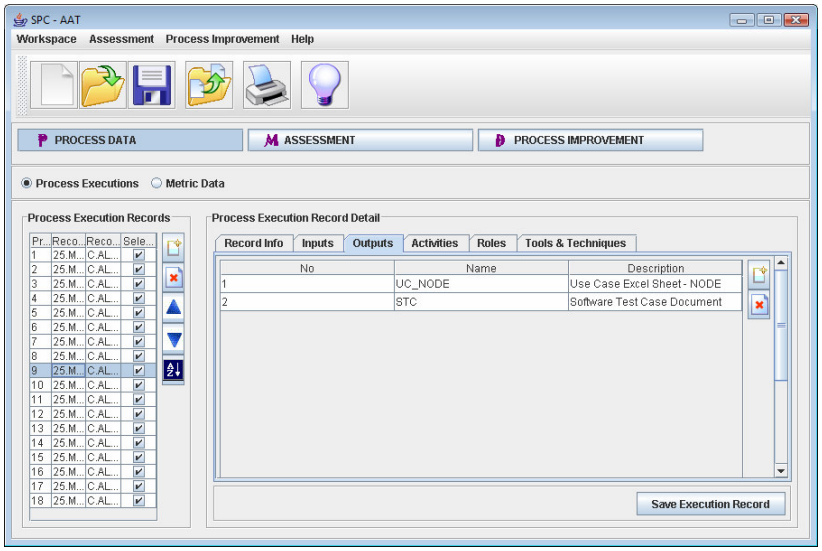


Figure 55 Process Execution Record of Process Execution #9

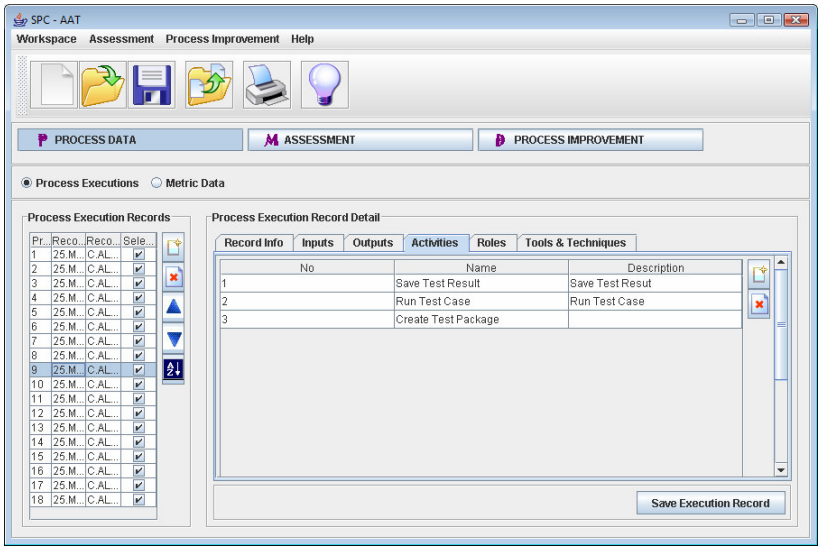


Figure 56 Process Execution Record of Process Execution #9

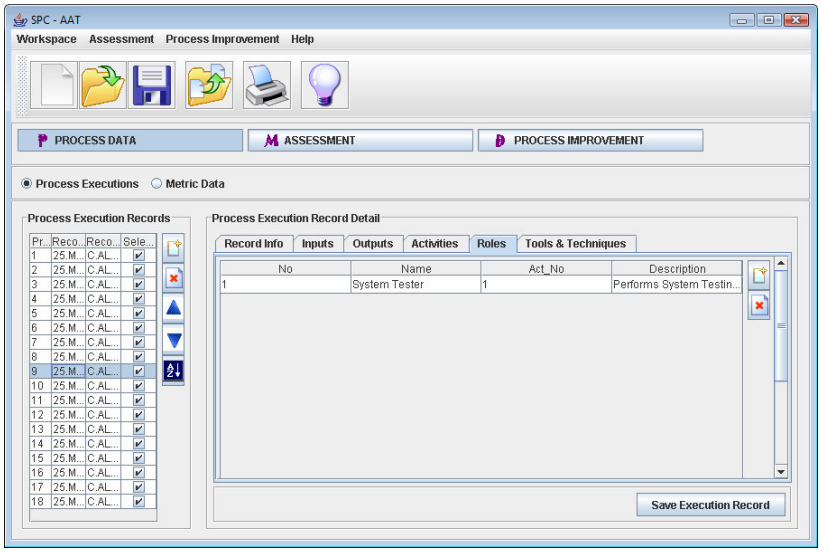


Figure 57 Process Execution Record of Process Execution #9

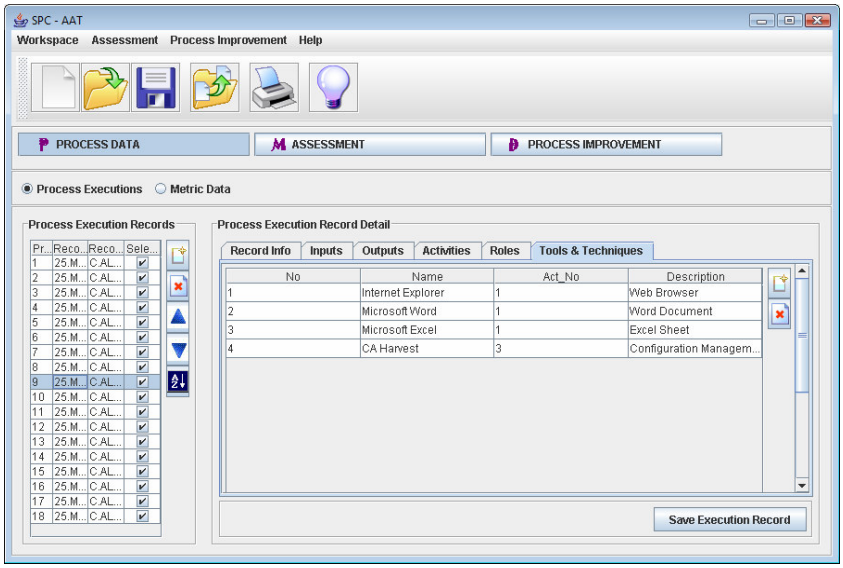


Figure 58 Process Execution Record of Process Execution #9

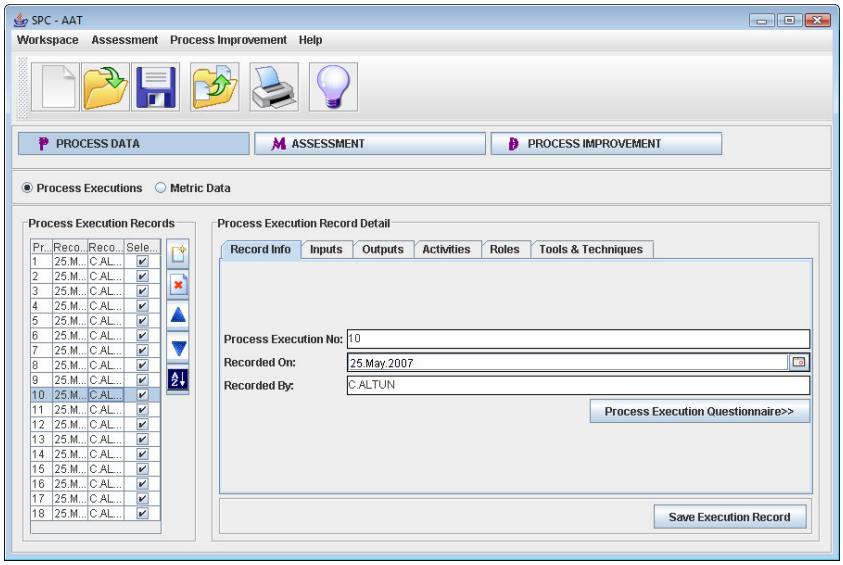


Figure 59 Process Execution Record of Process Execution #10

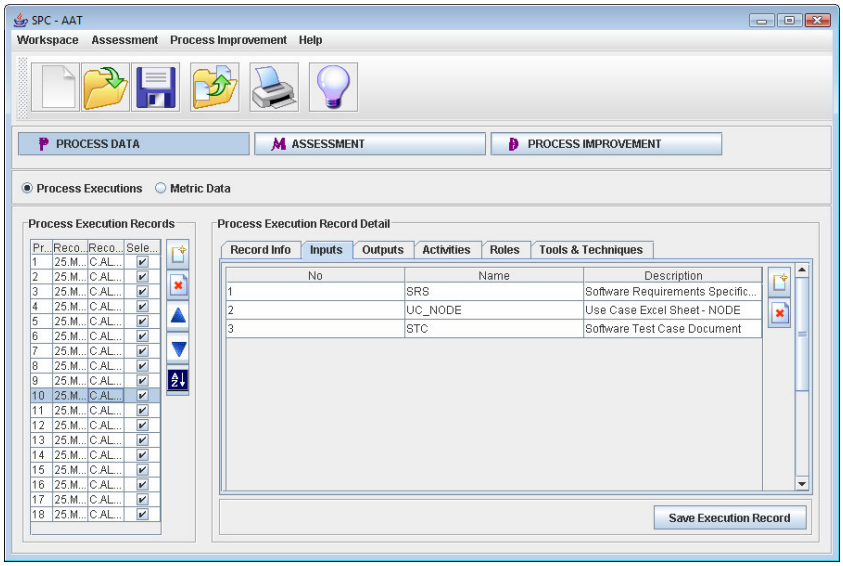


Figure 60 Process Execution Record of Process Execution #10

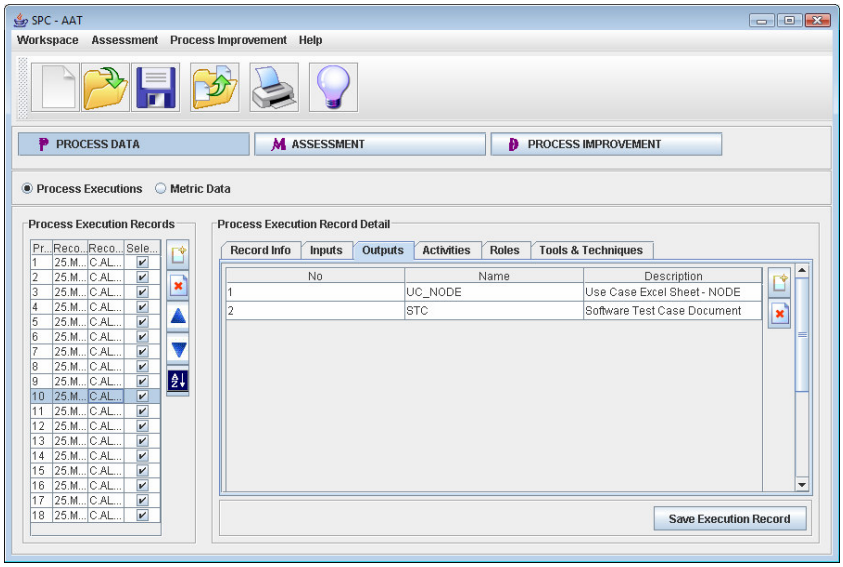


Figure 61 Process Execution Record of Process Execution #10

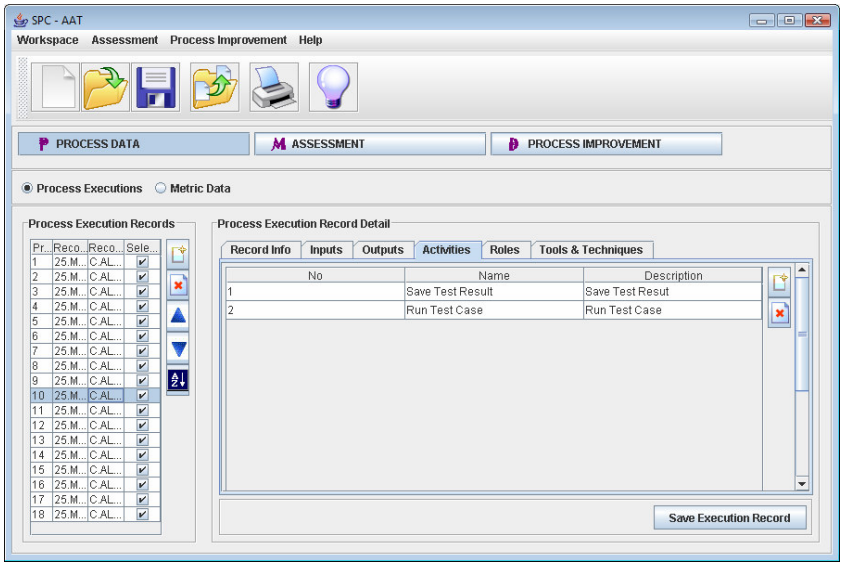


Figure 62 Process Execution Record of Process Execution #10

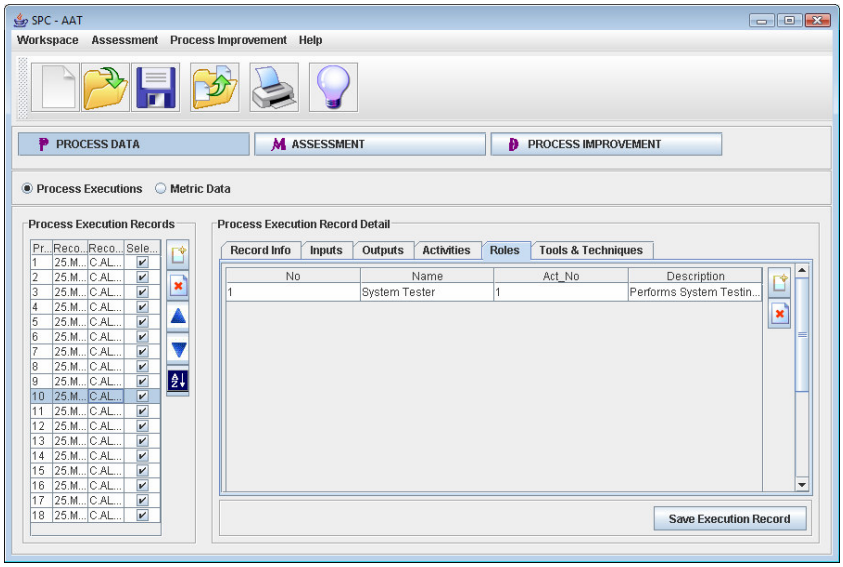


Figure 63 Process Execution Record of Process Execution #10

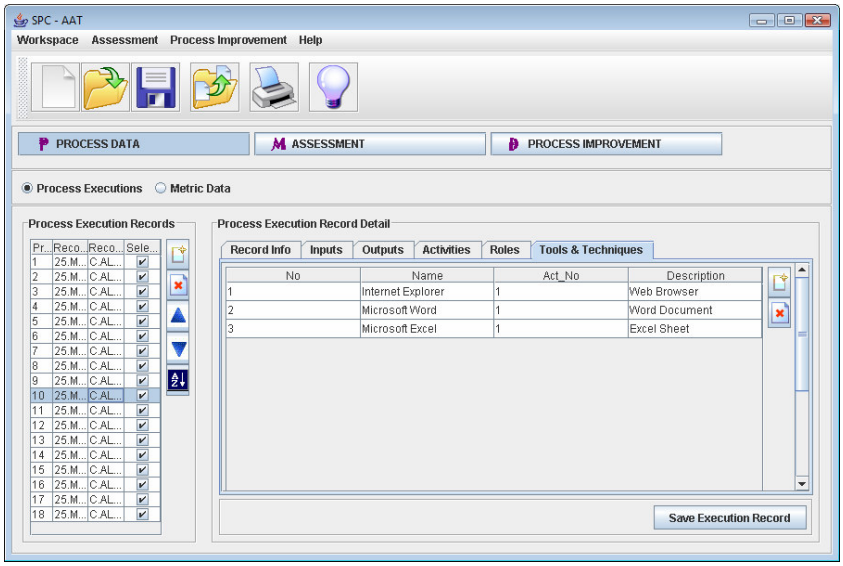


Figure 64 Process Execution Record of Process Execution #10

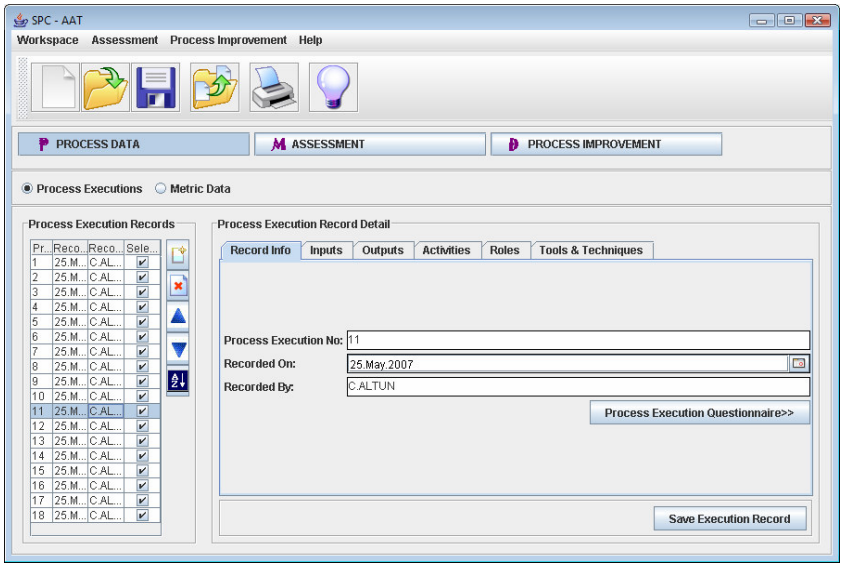


Figure 65 Process Execution Record of Process Execution #11

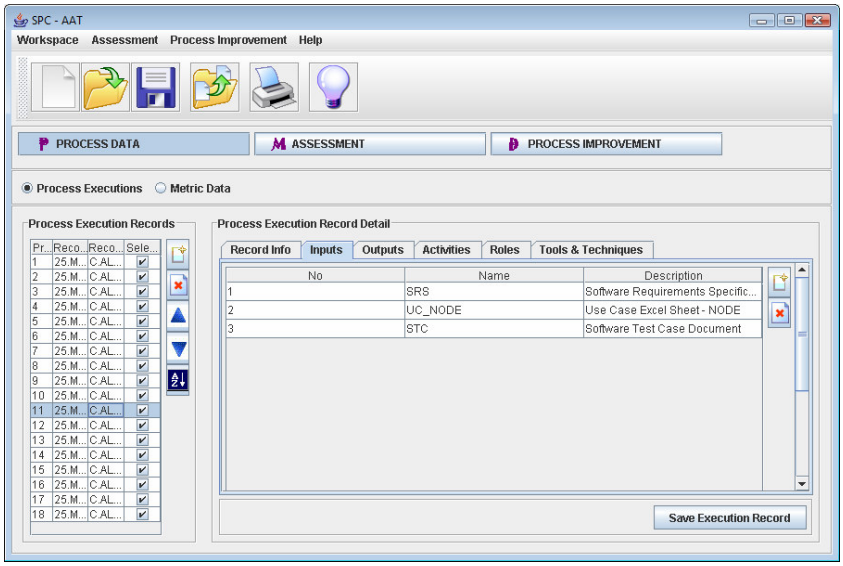


Figure 66 Process Execution Record of Process Execution #11

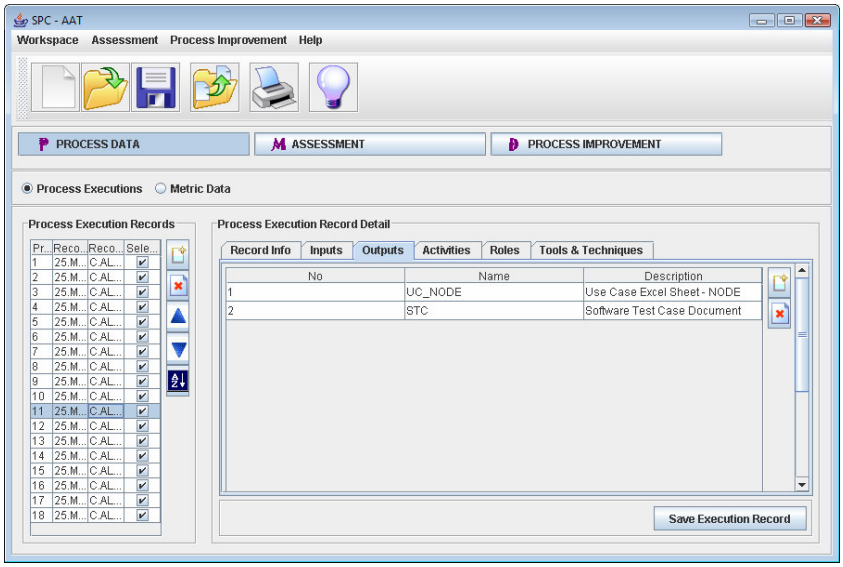


Figure 67 Process Execution Record of Process Execution #11

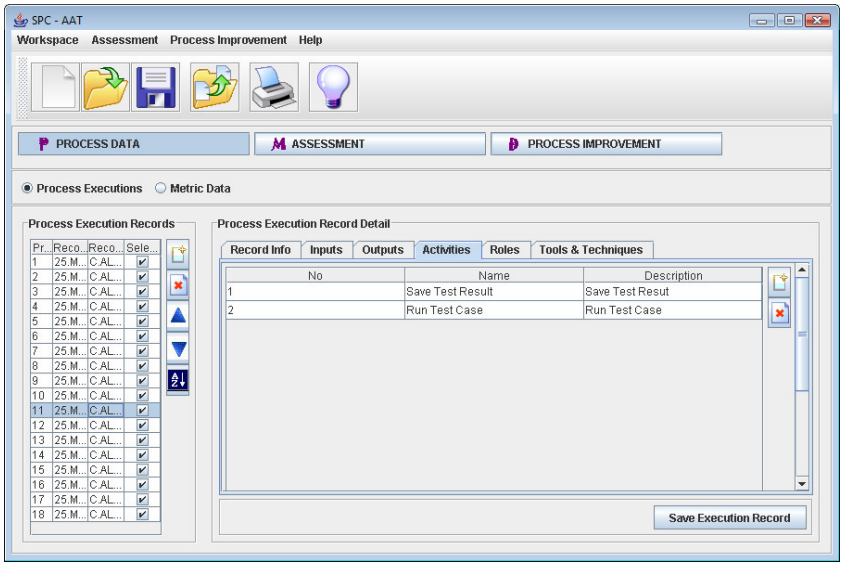


Figure 68 Process Execution Record of Process Execution #11

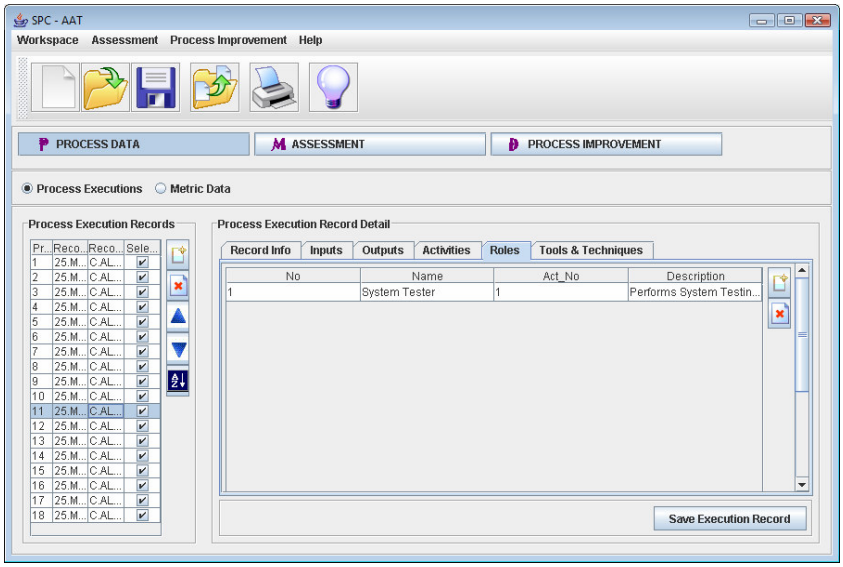


Figure 69 Process Execution Record of Process Execution #11

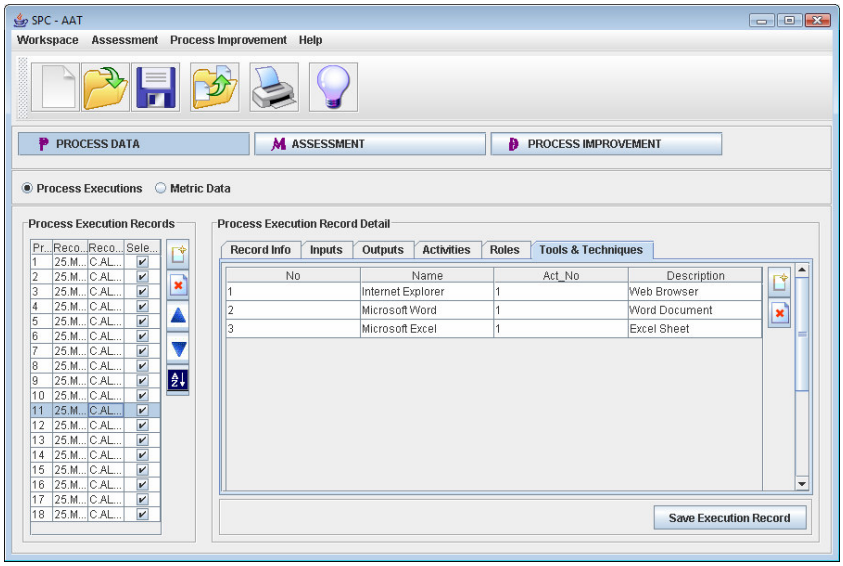


Figure 70 Process Execution Record of Process Execution #11

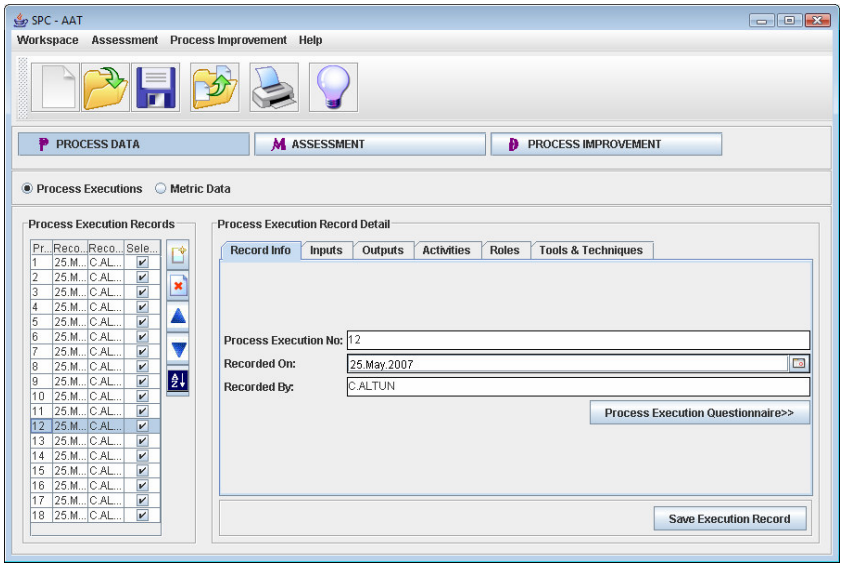


Figure 71 Process Execution Record of Process Execution #12

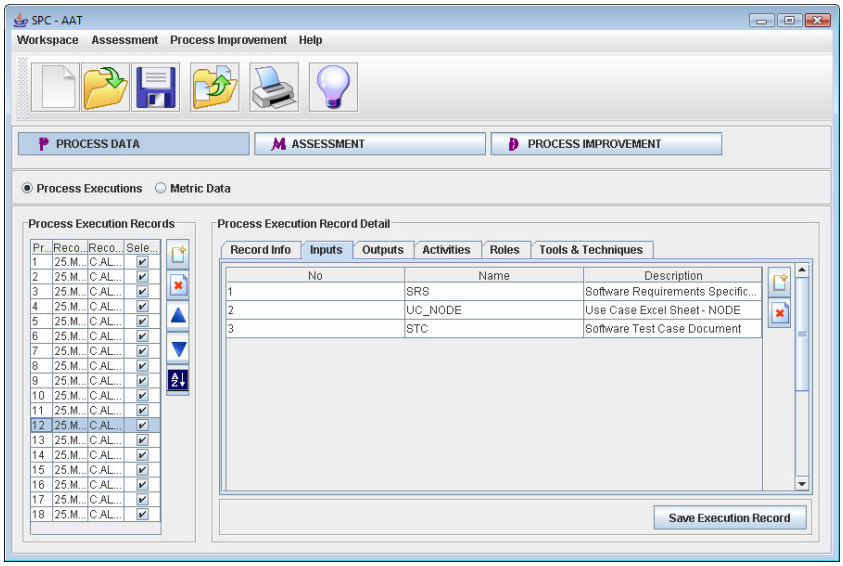


Figure 72 Process Execution Record of Process Execution #12

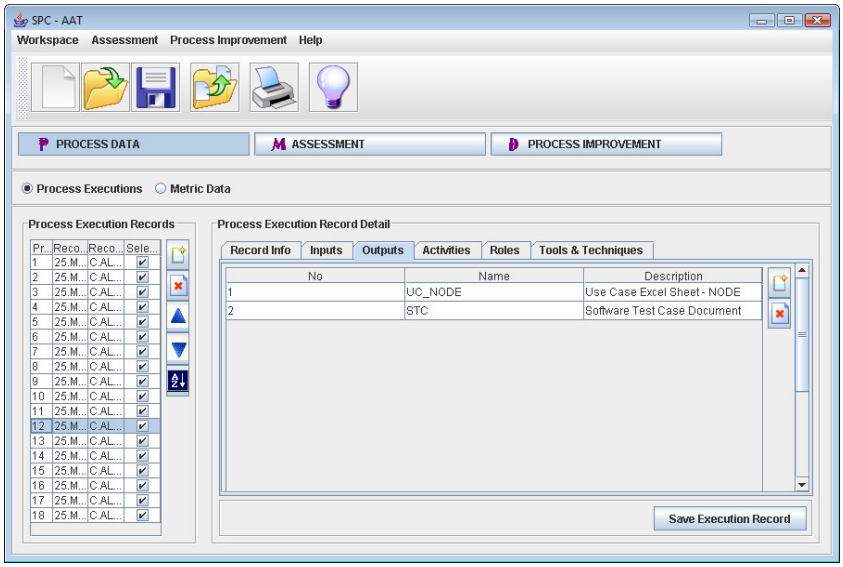


Figure 73 Process Execution Record of Process Execution #12

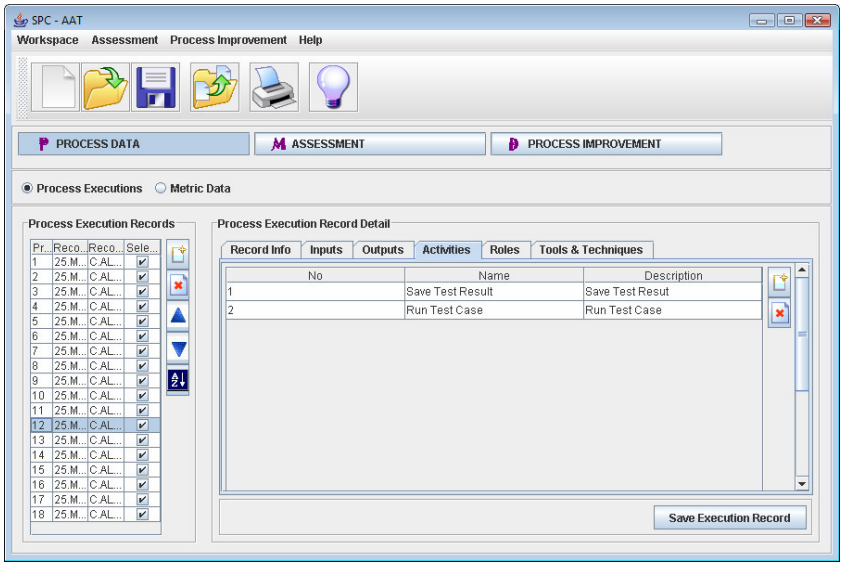


Figure 74 Process Execution Record of Process Execution #12

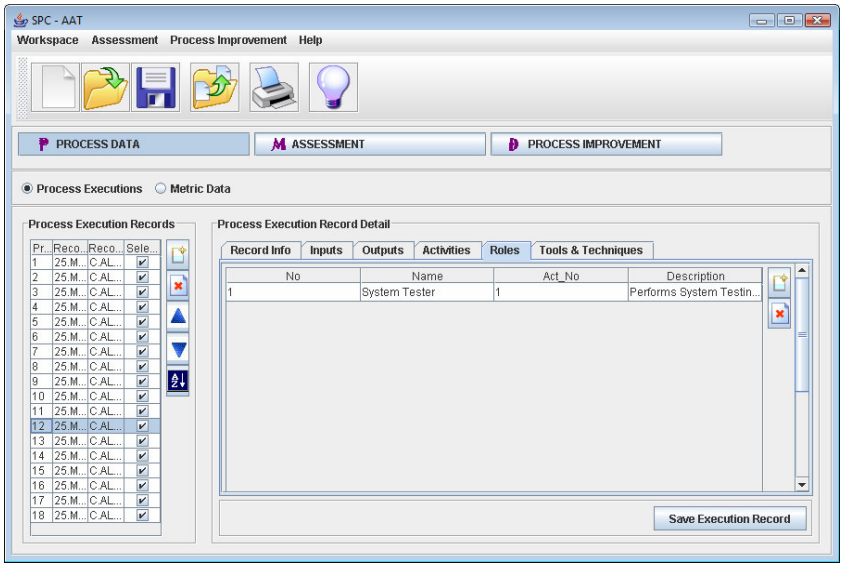


Figure 75 Process Execution Record of Process Execution #12

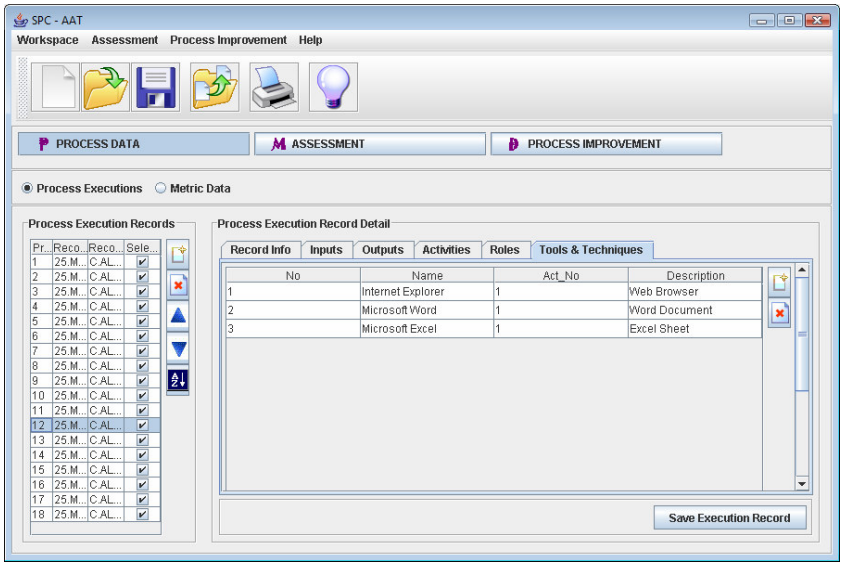


Figure 76 Process Execution Record of Process Execution #12

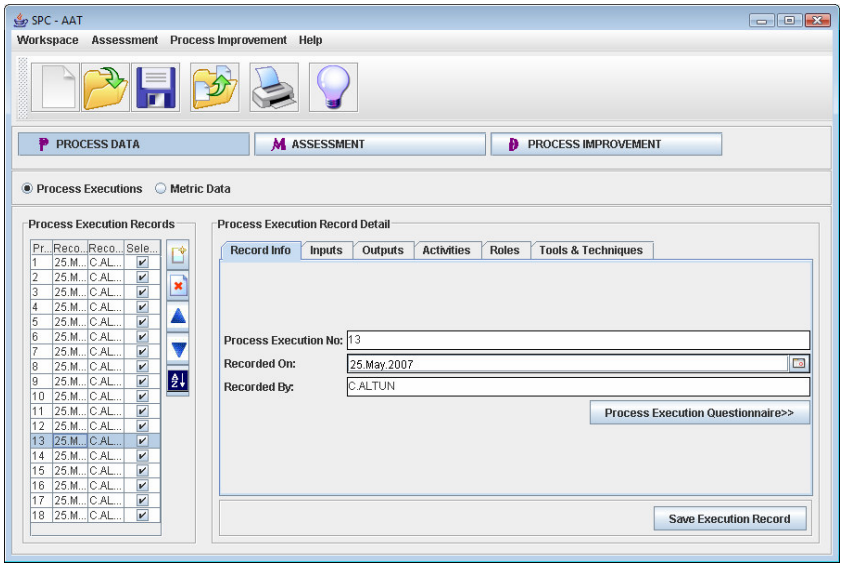


Figure 77 Process Execution Record of Process Execution #13

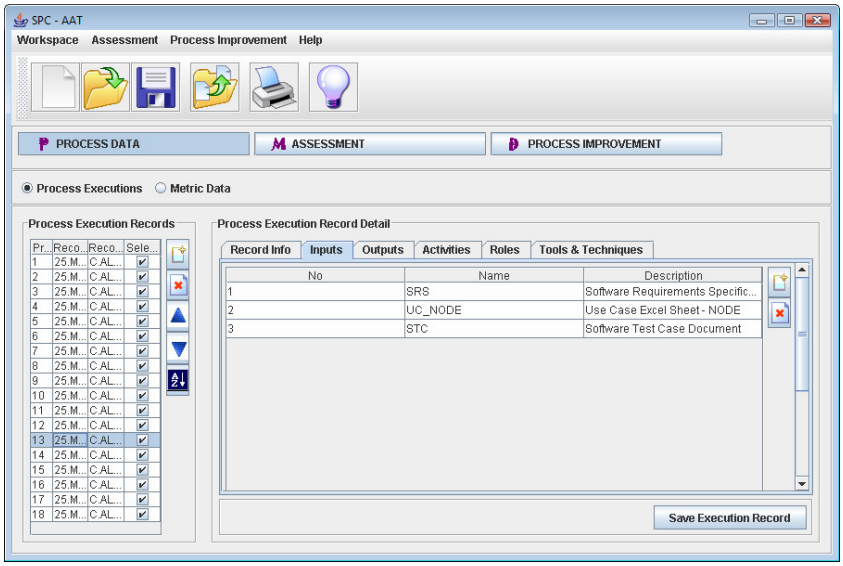


Figure 78 Process Execution Record of Process Execution #13

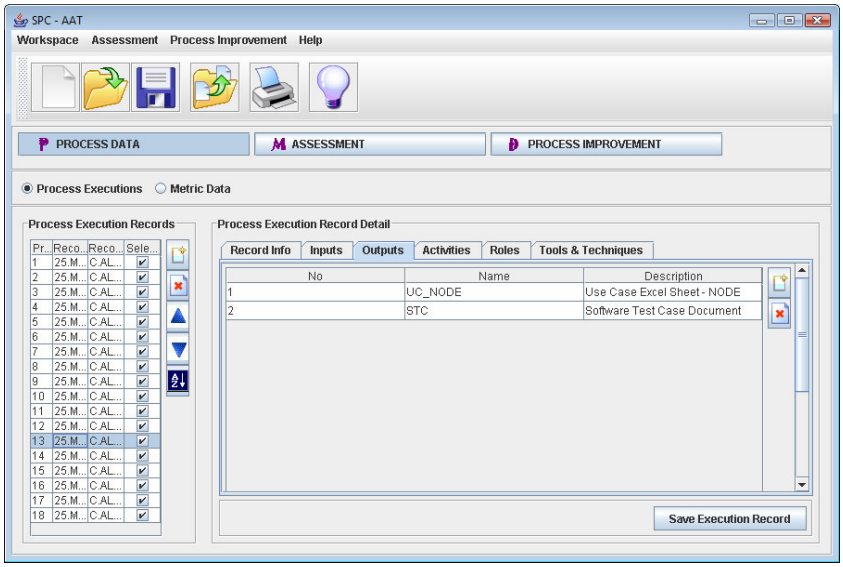


Figure 79 Process Execution Record of Process Execution #13

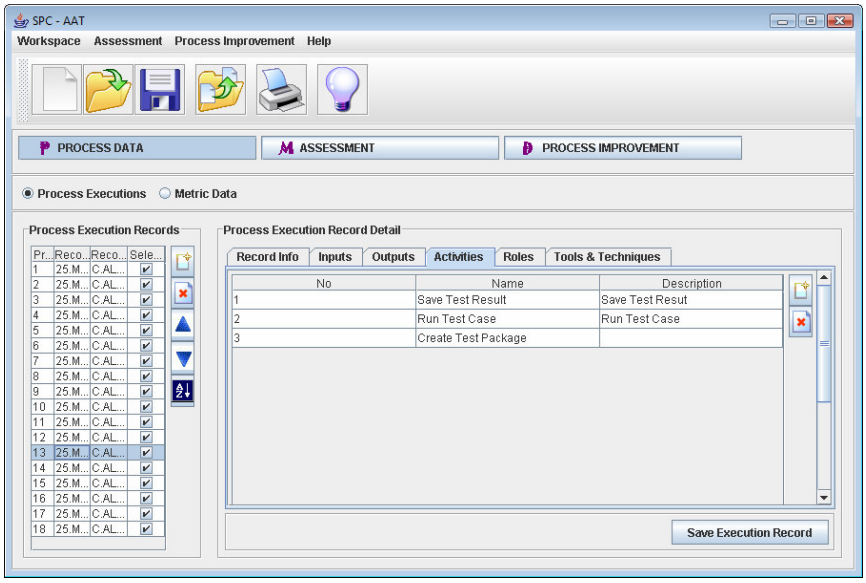


Figure 80 Process Execution Record of Process Execution #13

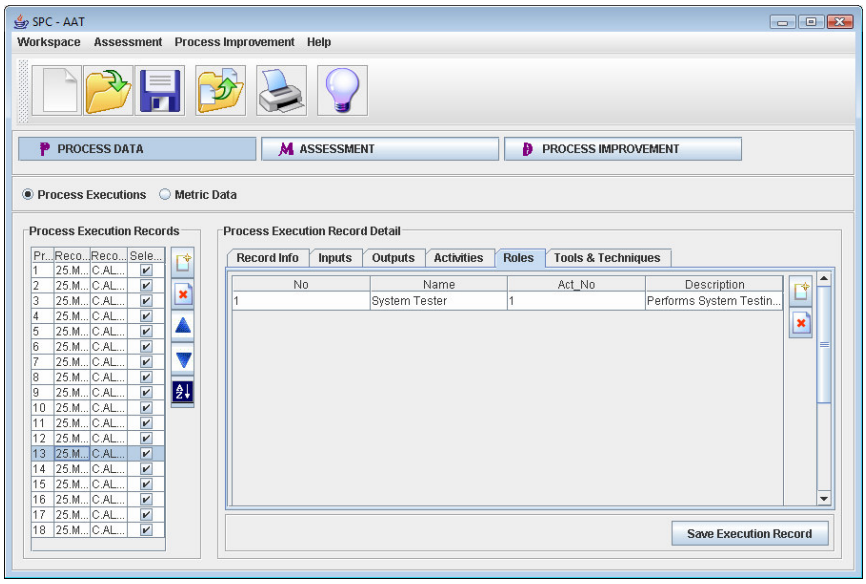


Figure 81 Process Execution Record of Process Execution #13

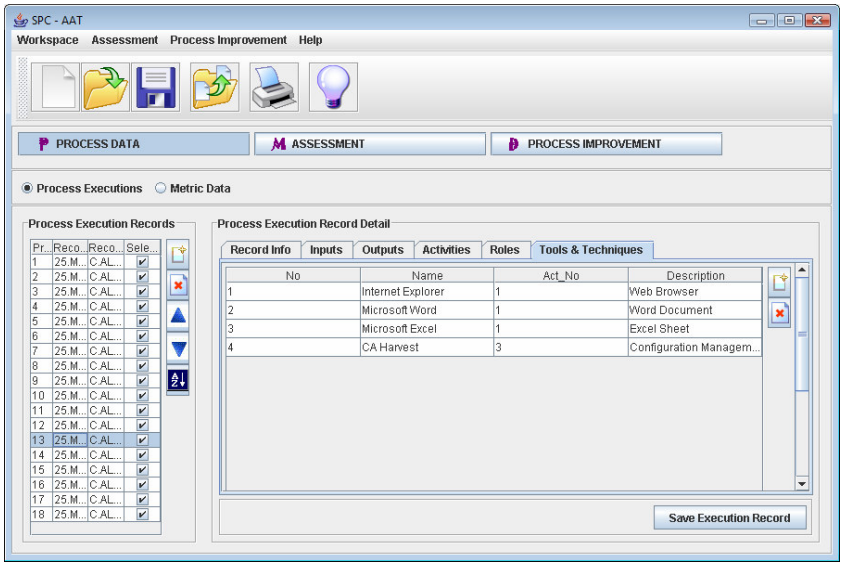


Figure 82 Process Execution Record of Process Execution #13

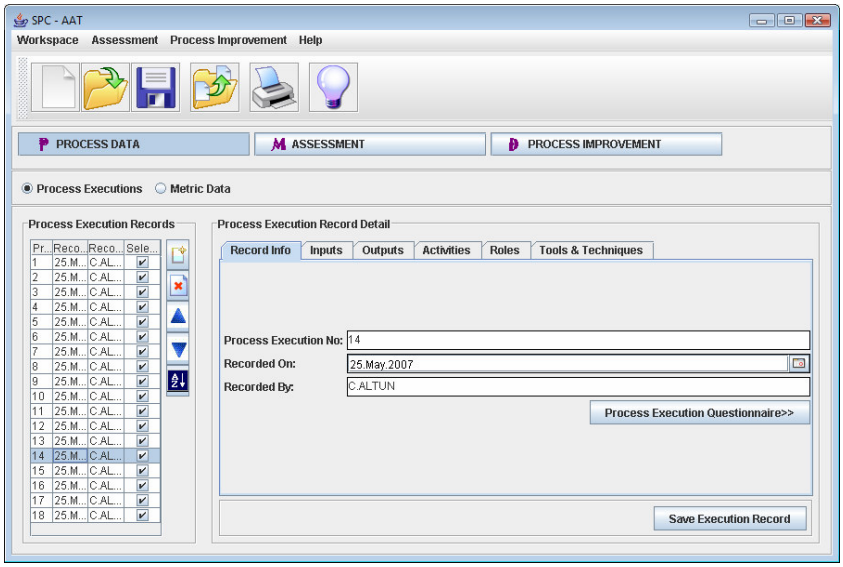


Figure 83 Process Execution Record of Process Execution #14

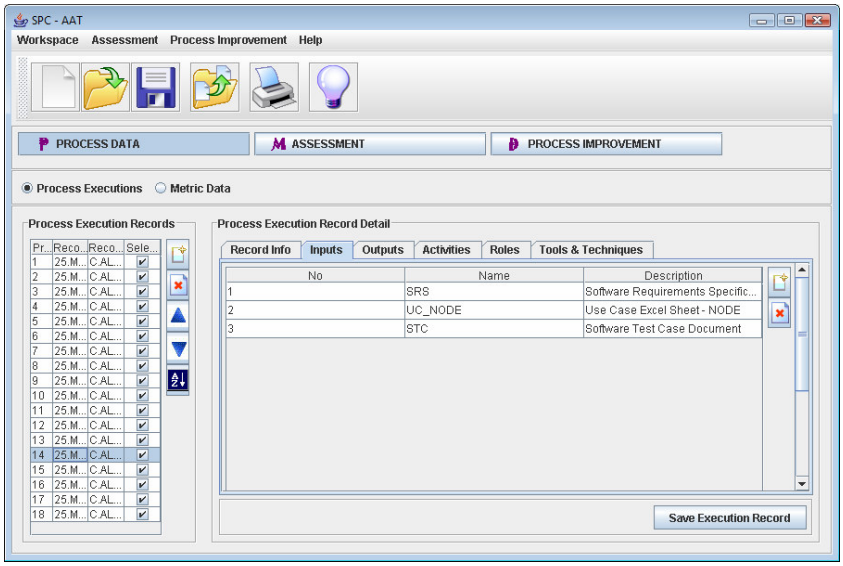


Figure 84 Process Execution Record of Process Execution #14

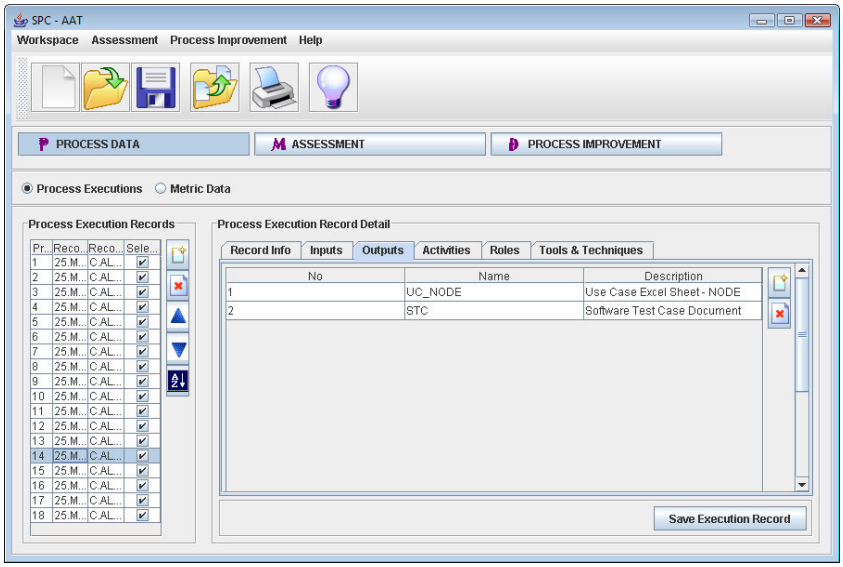


Figure 85 Process Execution Record of Process Execution #14

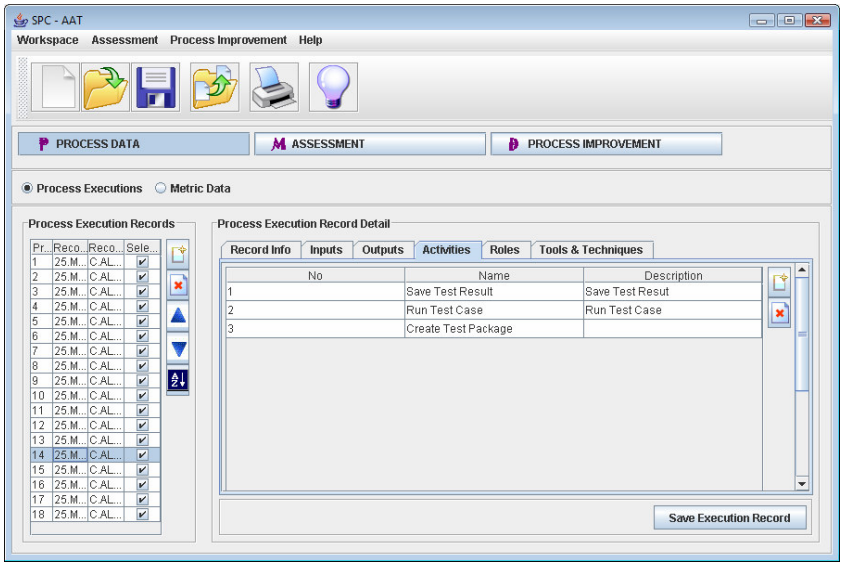


Figure 86 Process Execution Record of Process Execution #14

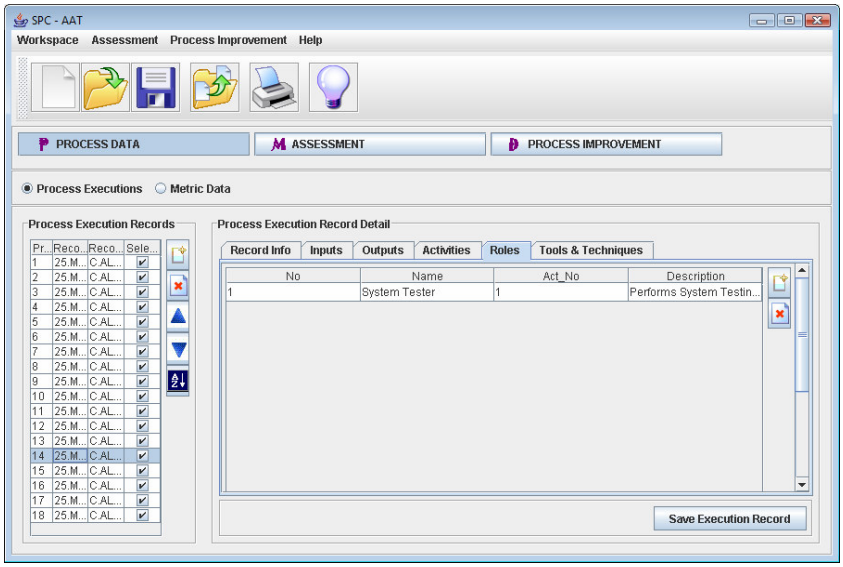


Figure 87 Process Execution Record of Process Execution #14

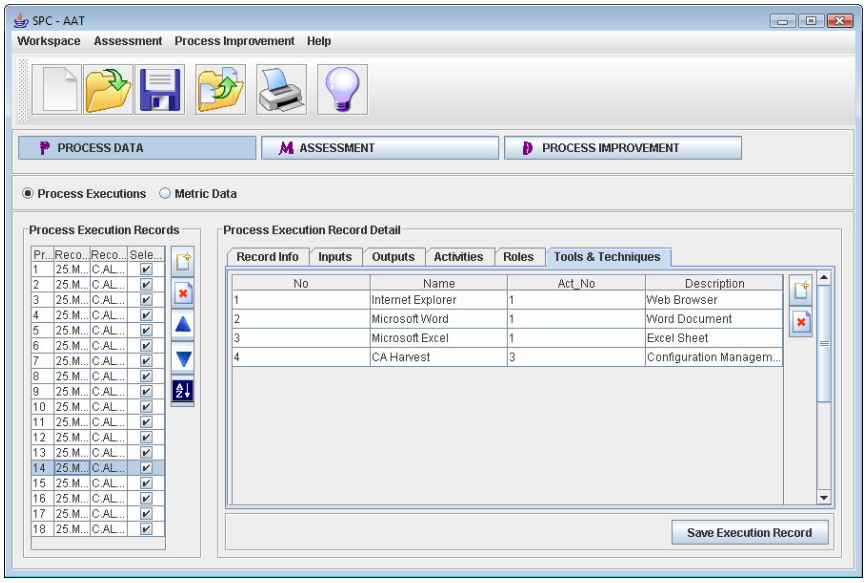


Figure 88 Process Execution Record of Process Execution #14

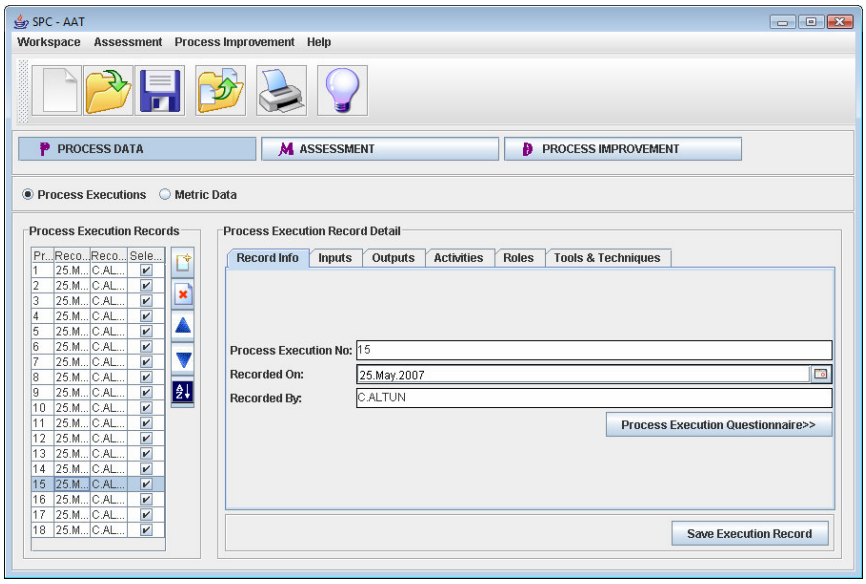


Figure 89 Process Execution Record of Process Execution #15

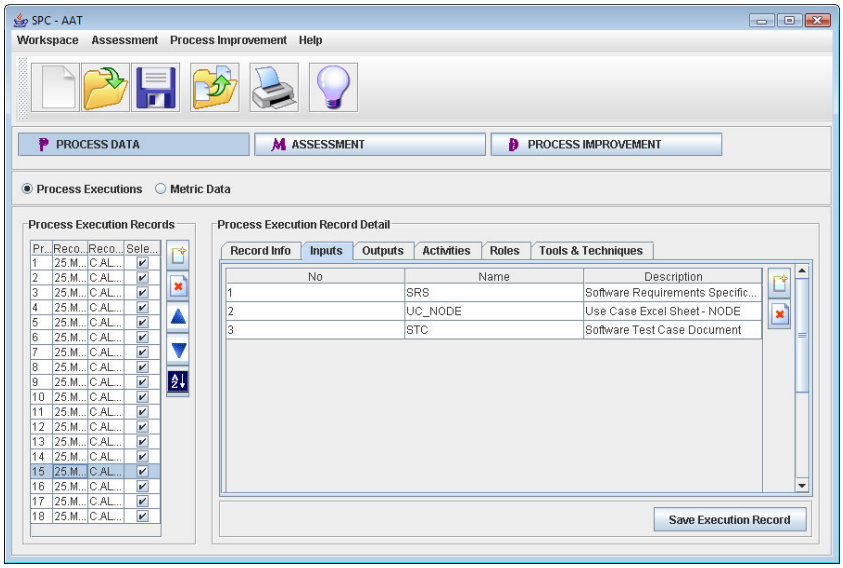


Figure 90 Process Execution Record of Process Execution #15

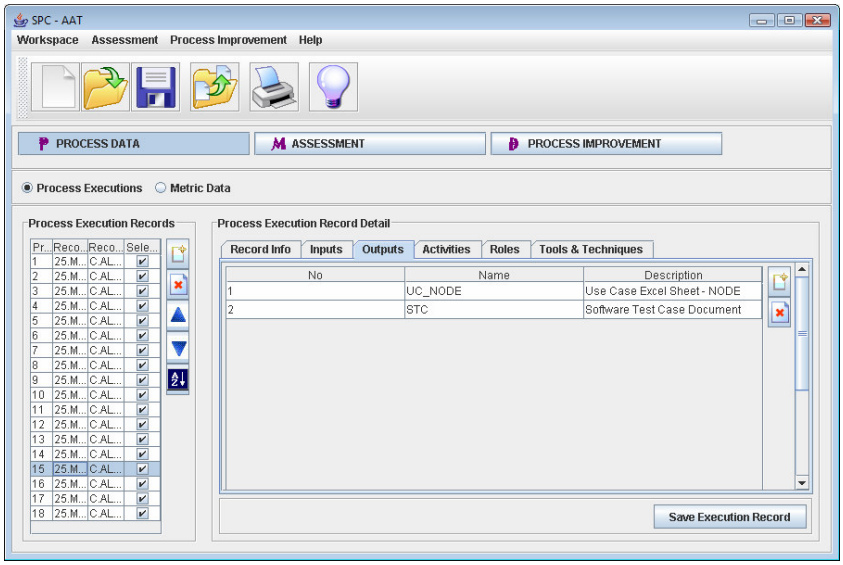


Figure 91 Process Execution Record of Process Execution #15

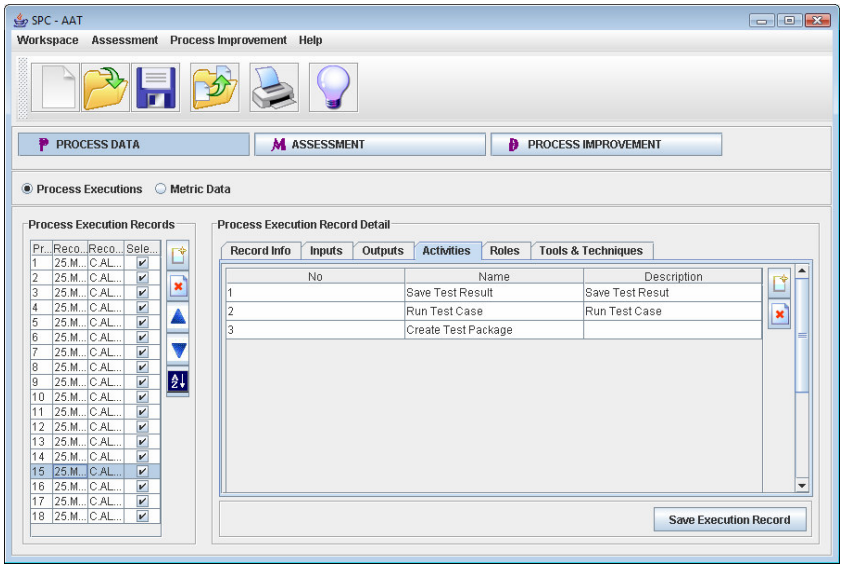


Figure 92 Process Execution Record of Process Execution #15

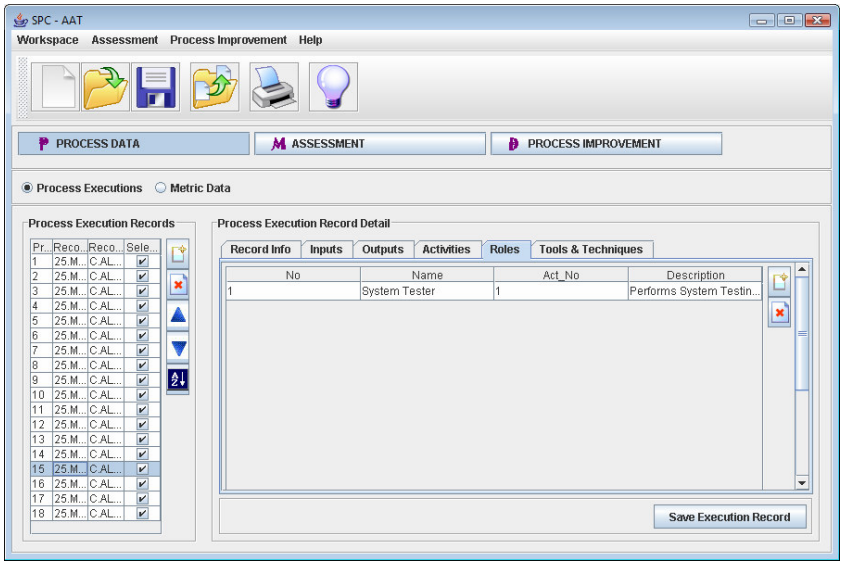


Figure 93 Process Execution Record of Process Execution #15

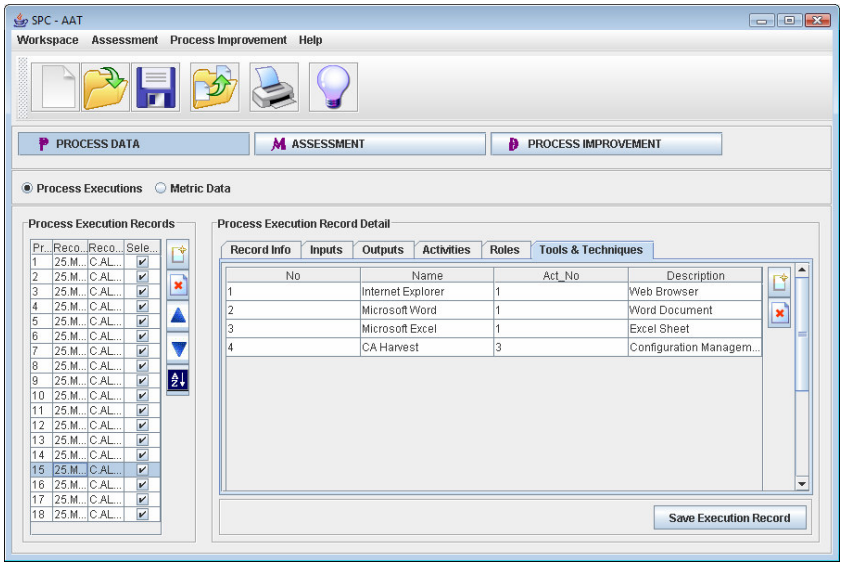


Figure 94 Process Execution Record of Process Execution #15

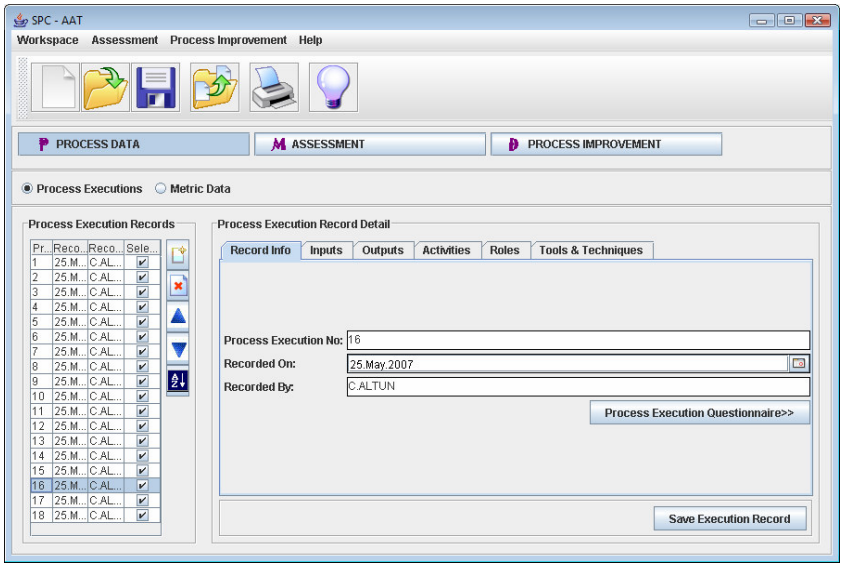


Figure 95 Process Execution Record of Process Execution #16

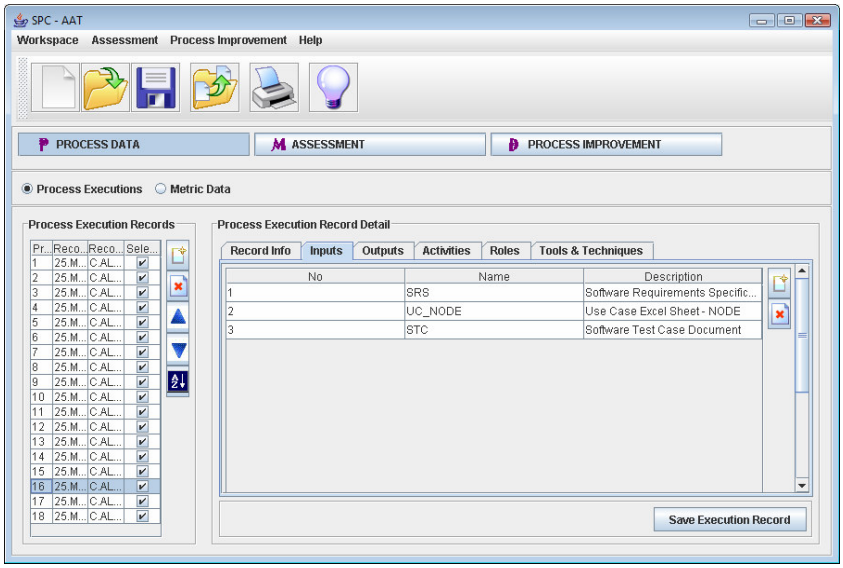


Figure 96 Process Execution Record of Process Execution #16

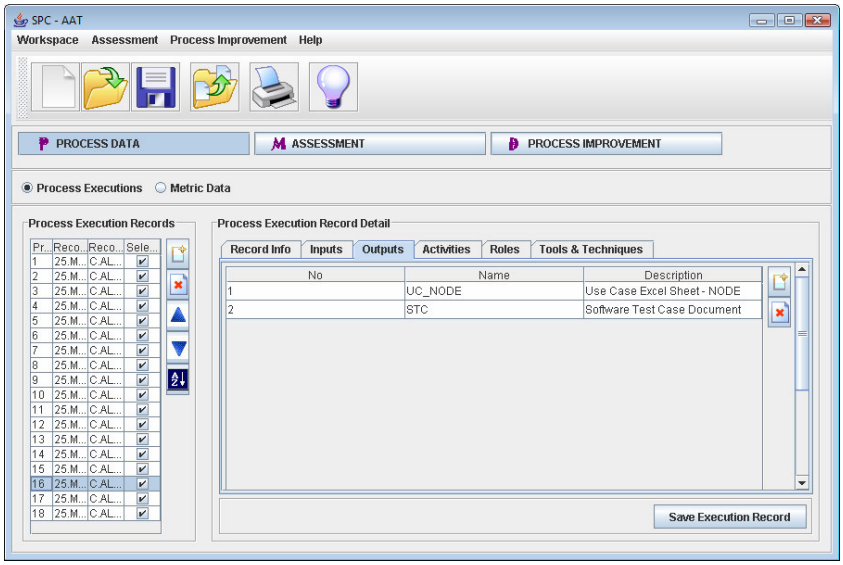


Figure 97 Process Execution Record of Process Execution #16

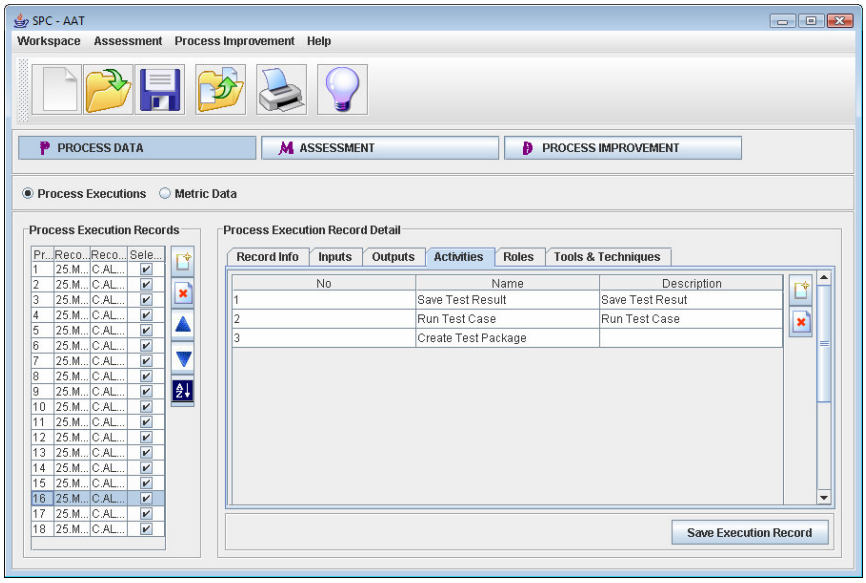


Figure 98 Process Execution Record of Process Execution #16

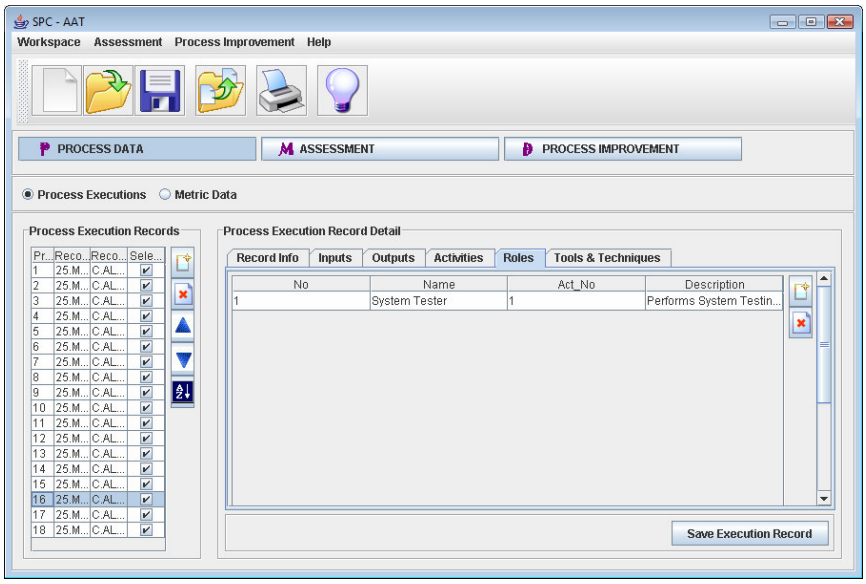


Figure 99 Process Execution Record of Process Execution #16

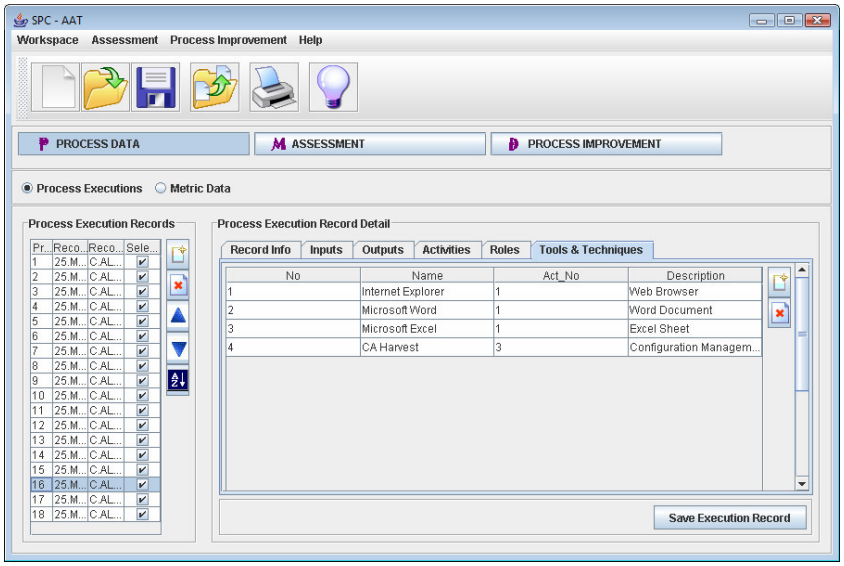


Figure 100 Process Execution Record of Process Execution #16

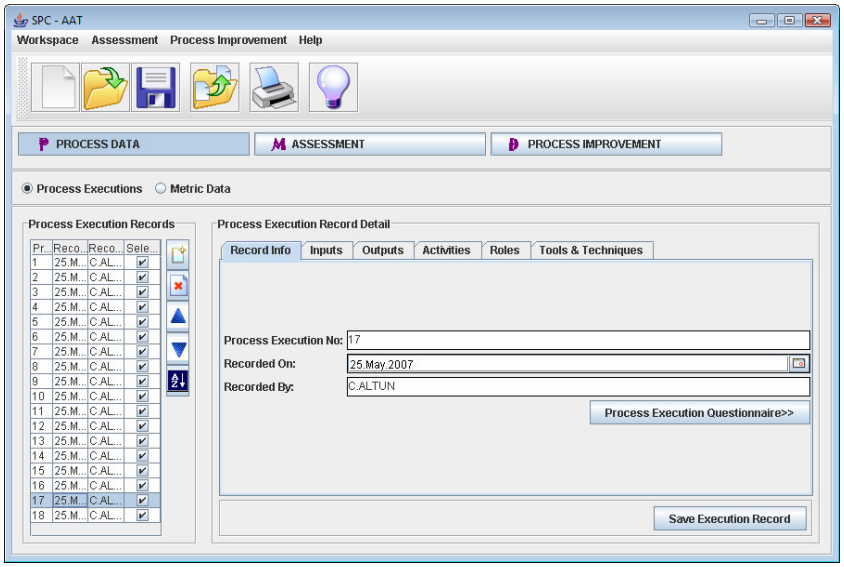


Figure 101 Process Execution Record of Process Execution #17

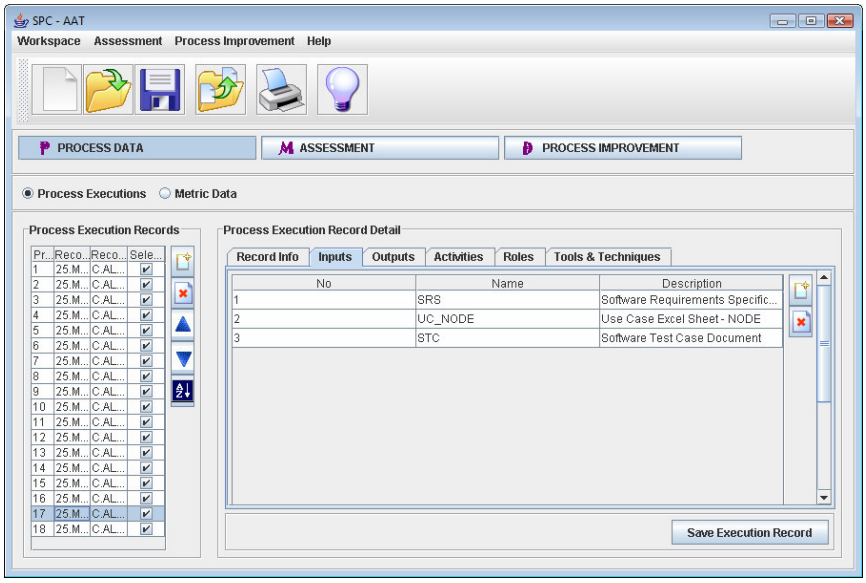


Figure 102 Process Execution Record of Process Execution #17

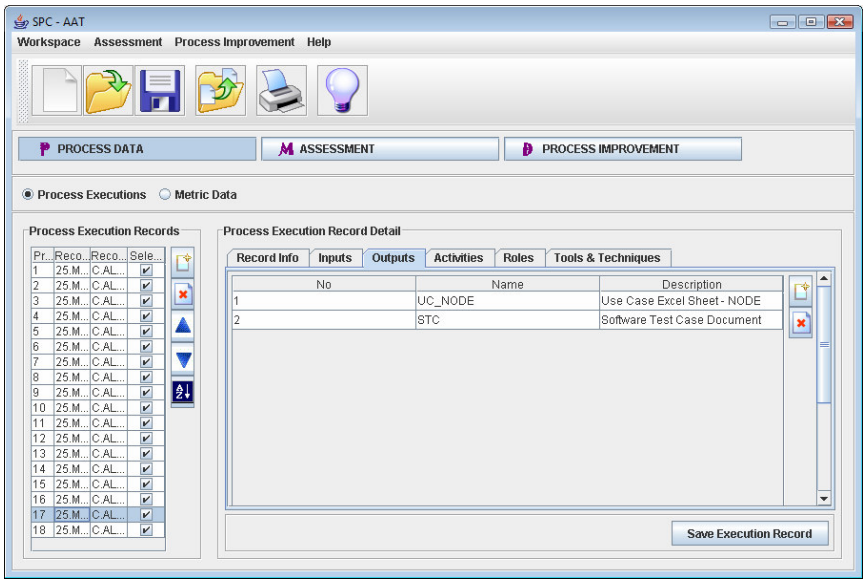


Figure 103 Process Execution Record of Process Execution #17

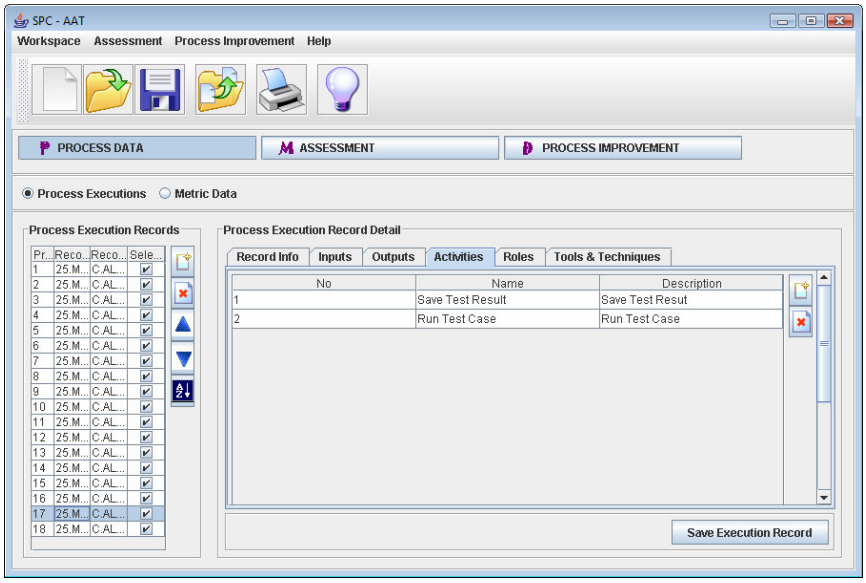


Figure 104 Process Execution Record of Process Execution #17

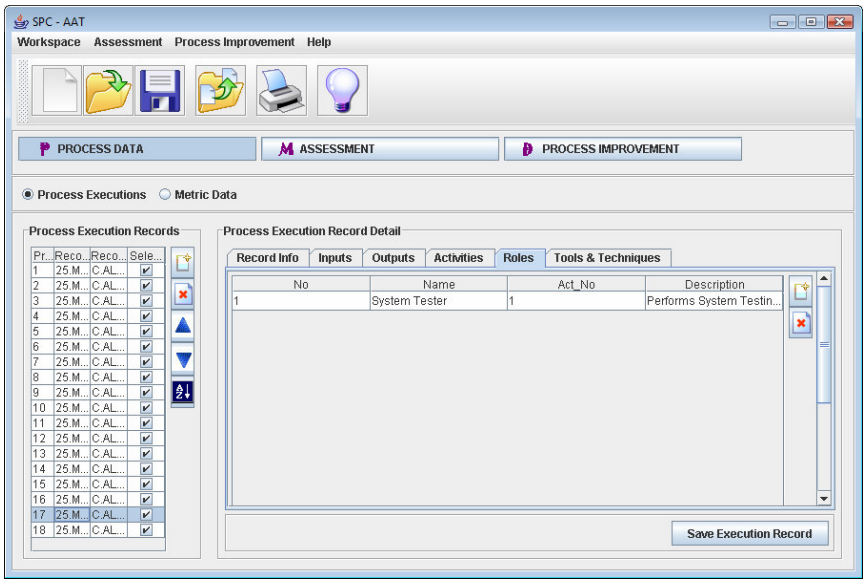


Figure 105 Process Execution Record of Process Execution #17

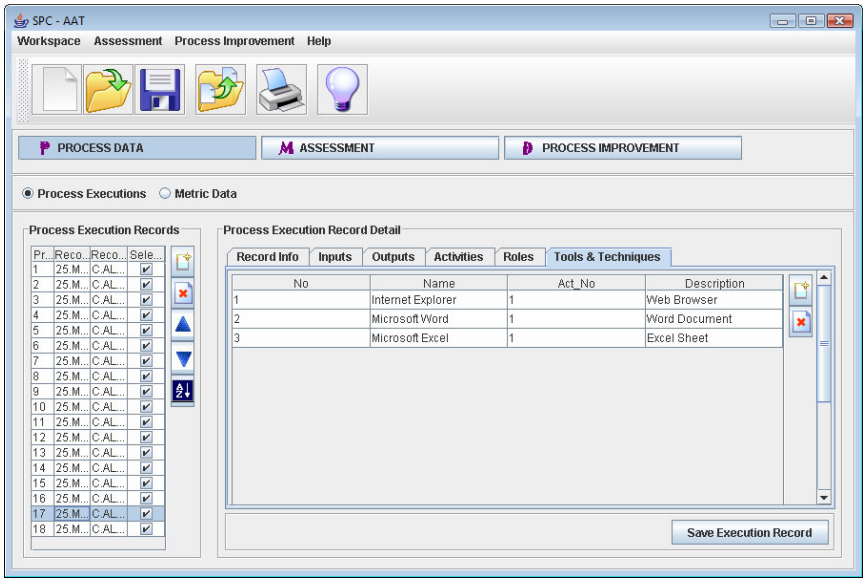


Figure 106 Process Execution Record of Process Execution #17

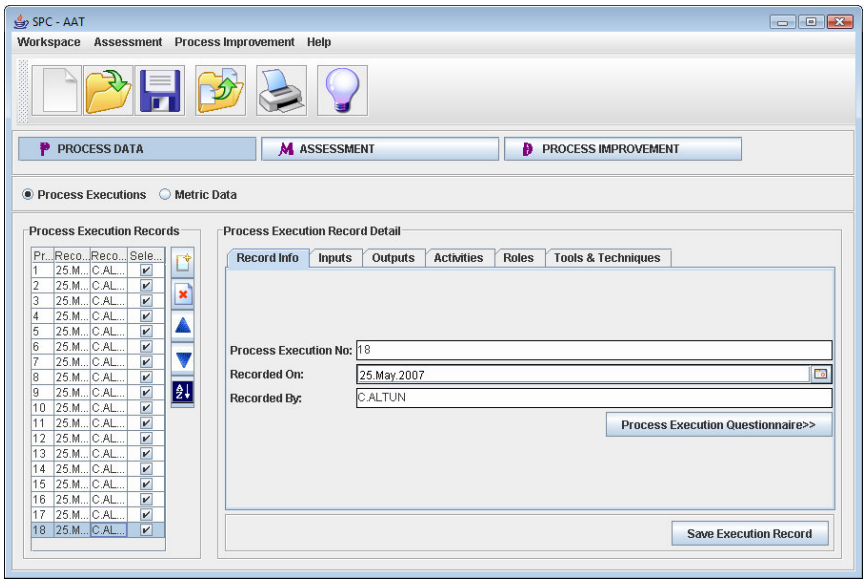


Figure 107 Process Execution Record of Process Execution #18

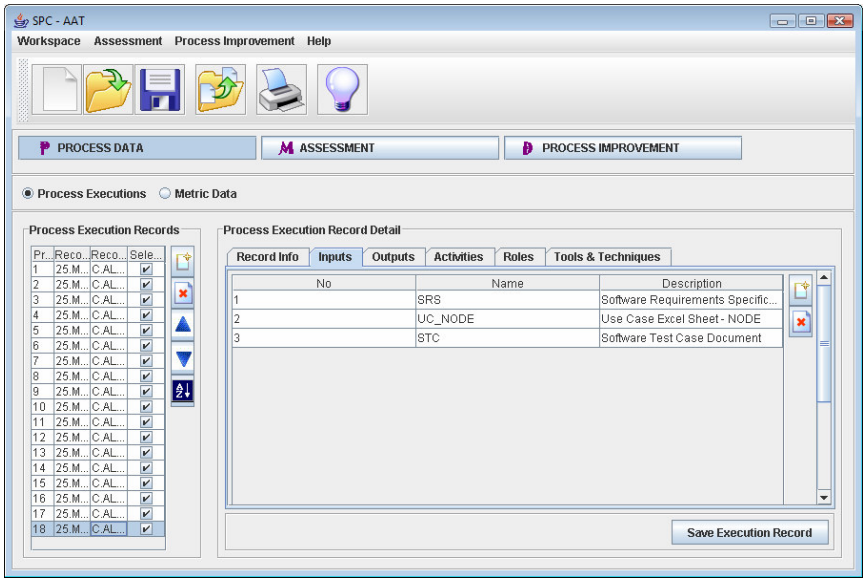


Figure 108 Process Execution Record of Process Execution #18

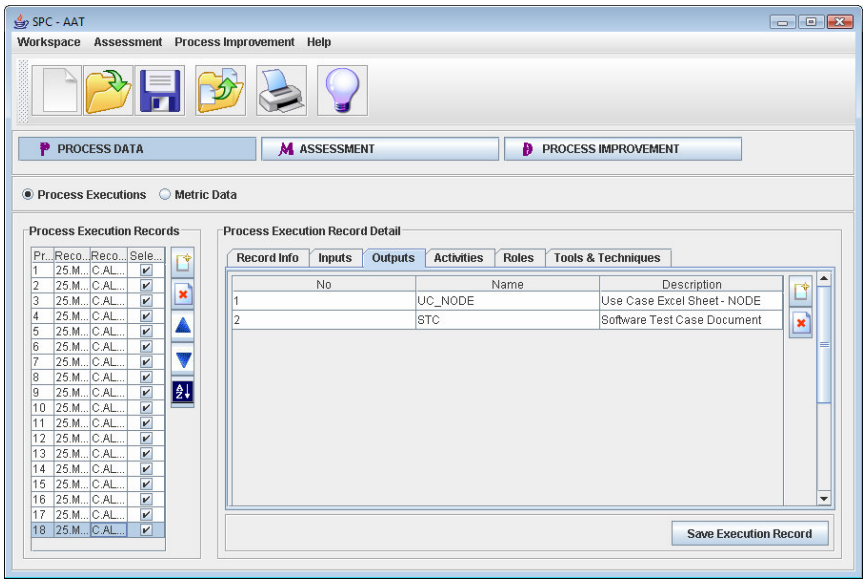


Figure 109 Process Execution Record of Process Execution #18

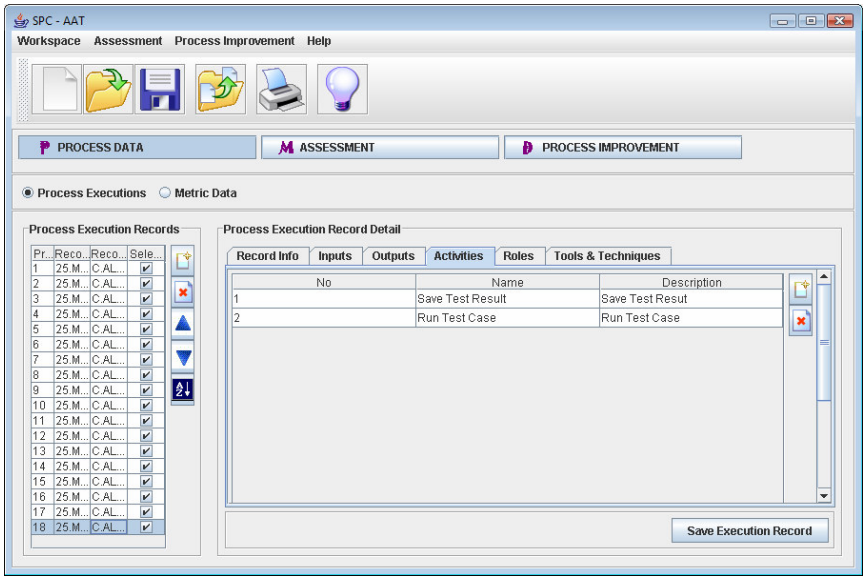


Figure 110 Process Execution Record of Process Execution #18

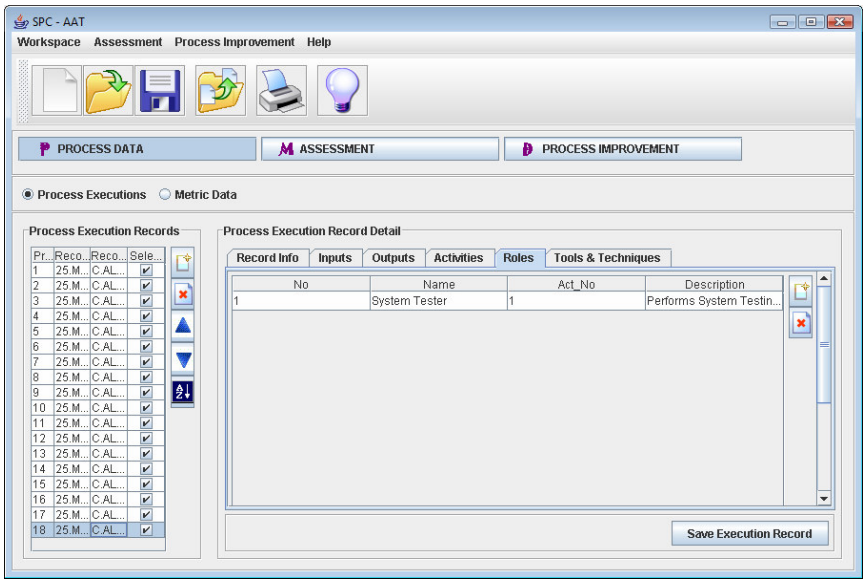


Figure 111 Process Execution Record of Process Execution #18

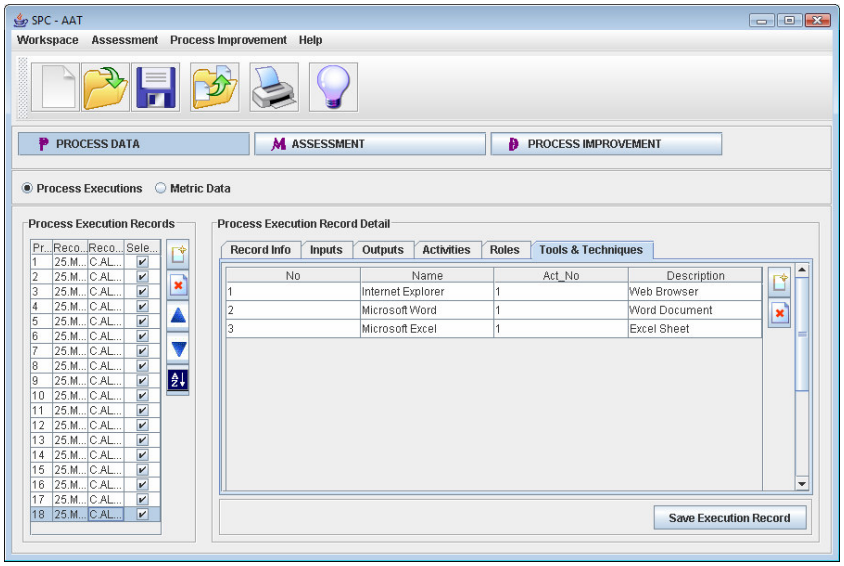


Figure 112 Process Execution Record of Process Execution #18

Process Execution Records of Case Study B

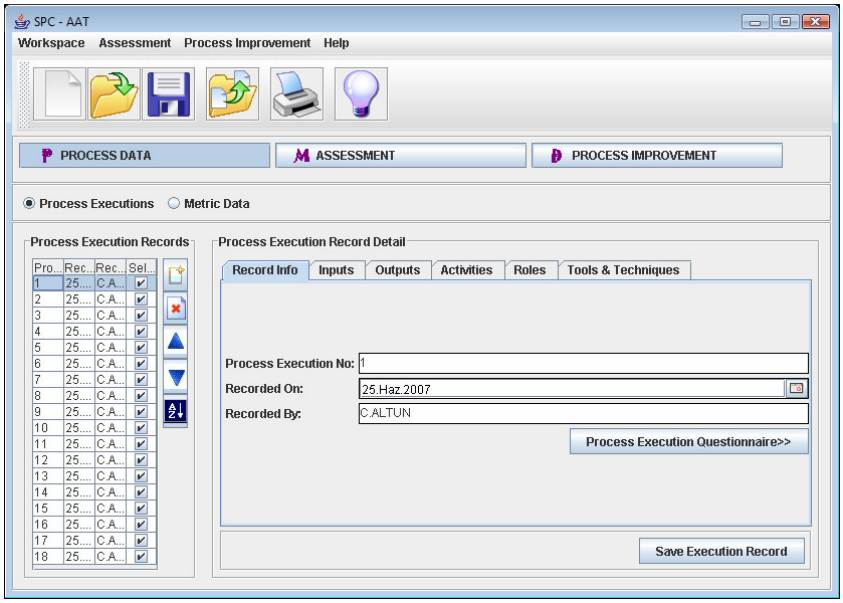


Figure 113 Process Execution Record of Process Execution #1

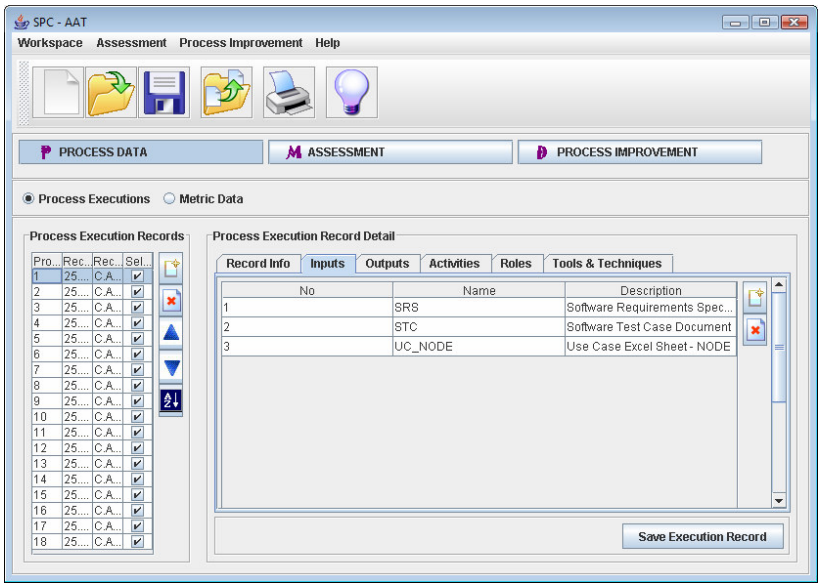


Figure 114 Process Execution Record of Process Execution #1

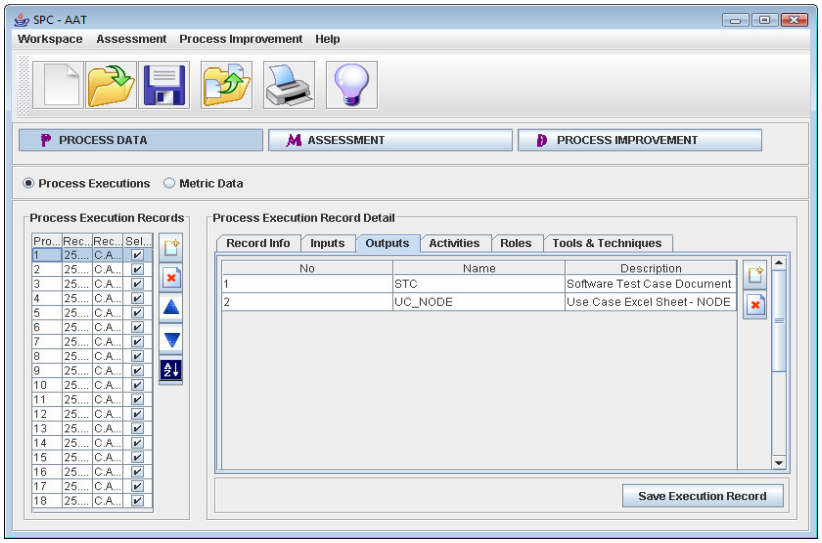


Figure 115 Process Execution Record of Process Execution #1

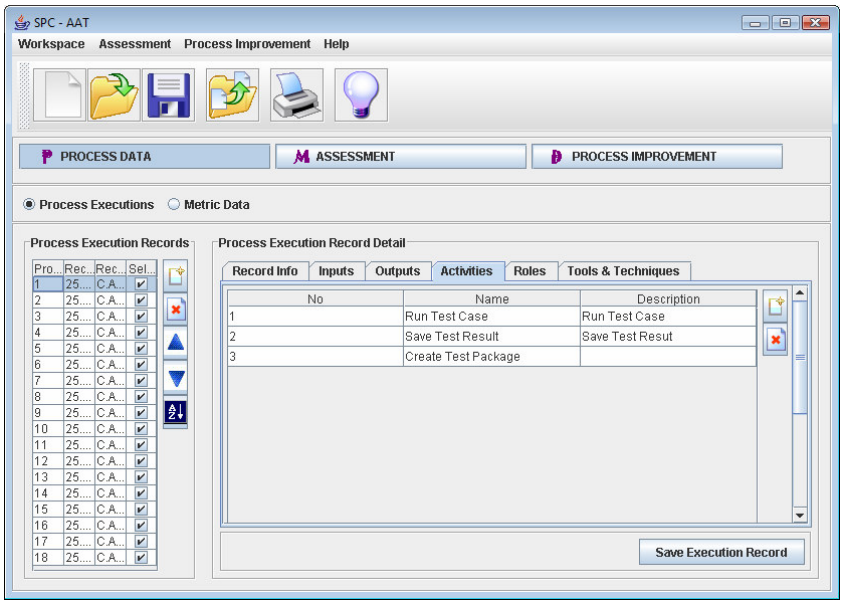


Figure 116 Process Execution Record of Process Execution #1

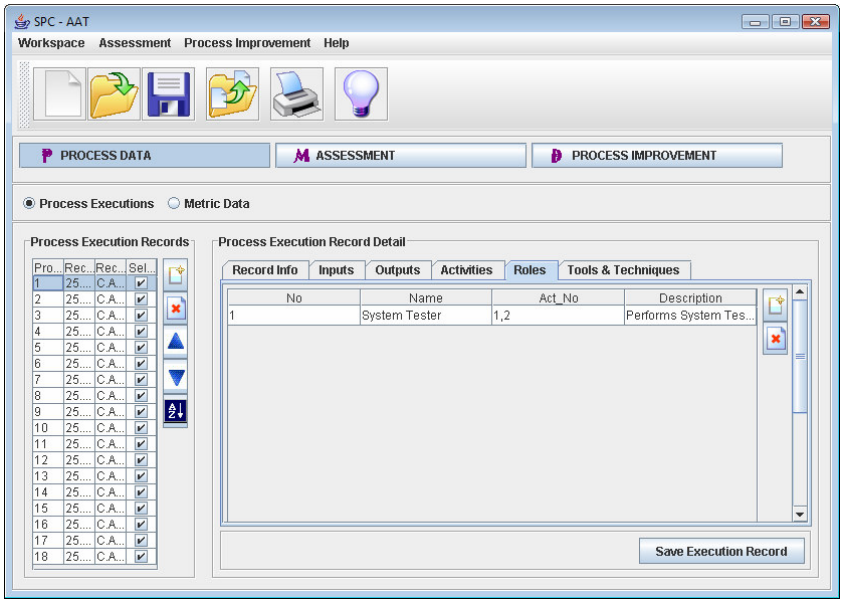


Figure 117 Process Execution Record of Process Execution #1

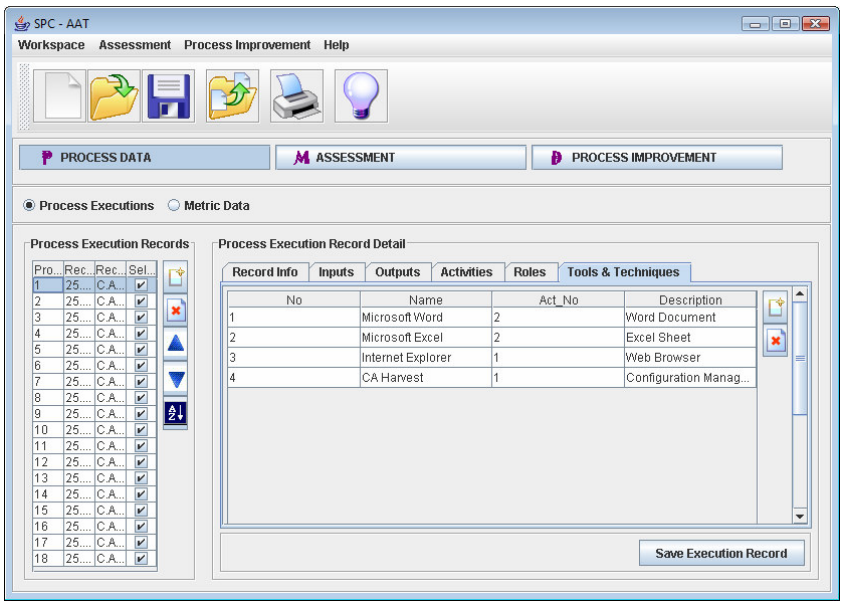


Figure 118 Process Execution Record of Process Execution #1

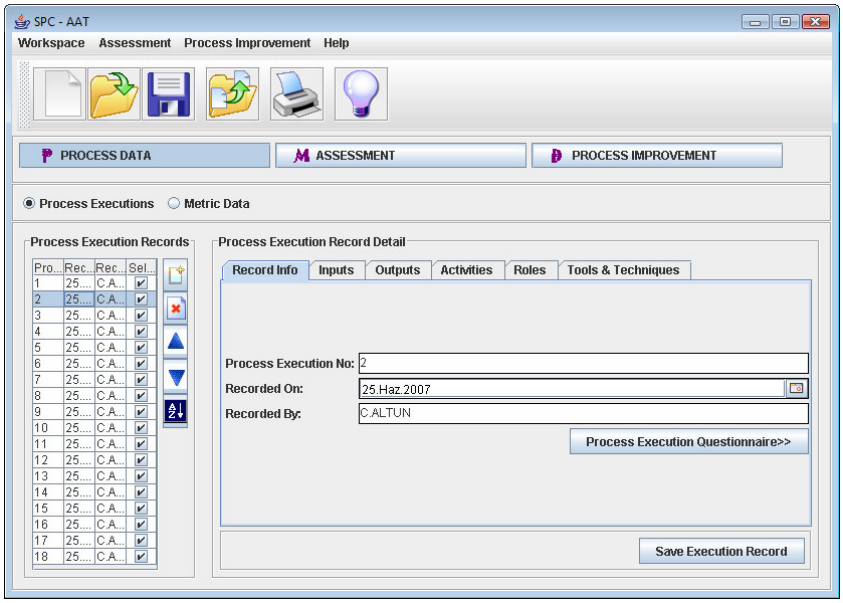


Figure 119 Process Execution Record of Process Execution #2

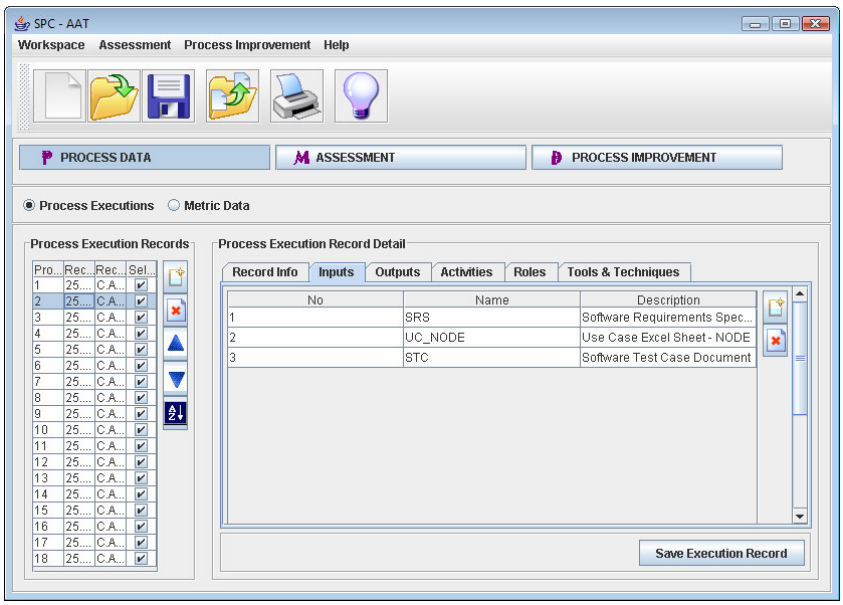


Figure 120 Process Execution Record of Process Execution #2

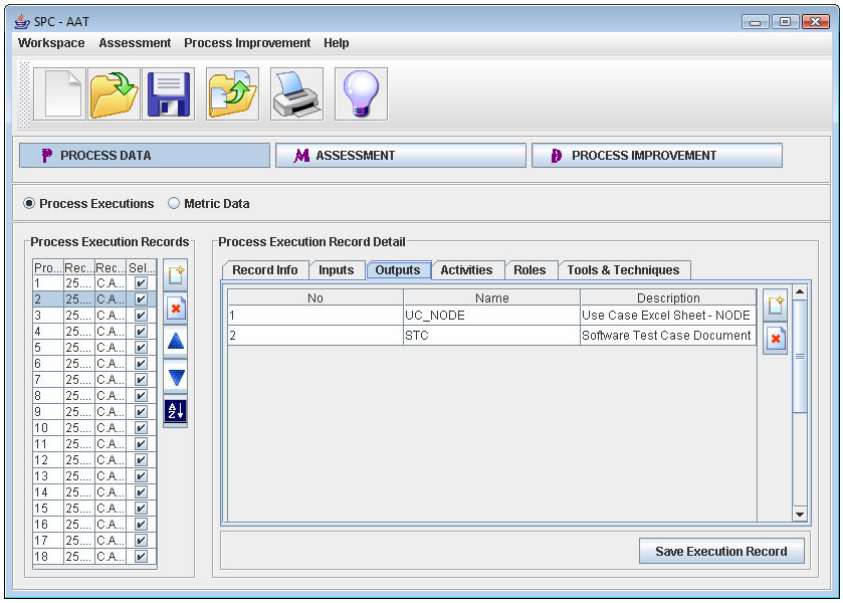


Figure 121 Process Execution Record of Process Execution #2

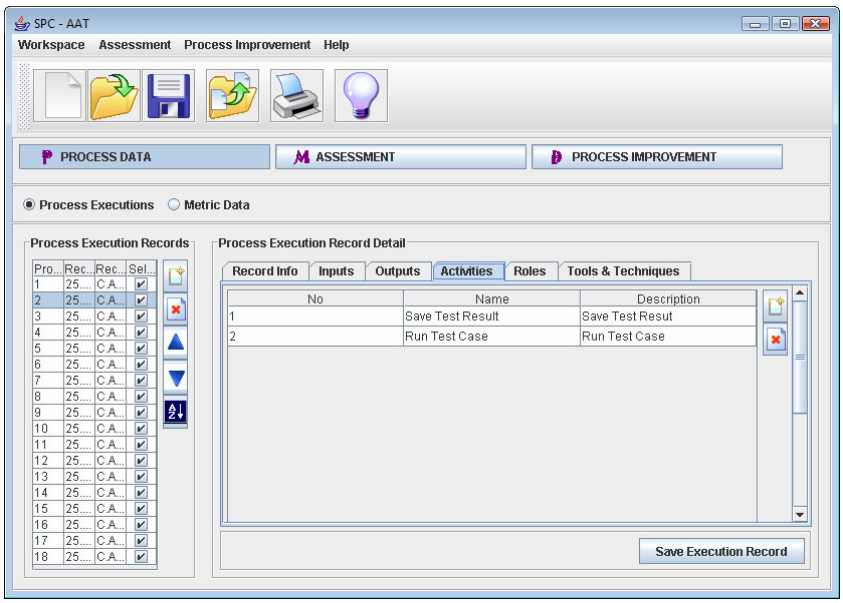


Figure 122 Process Execution Record of Process Execution #2

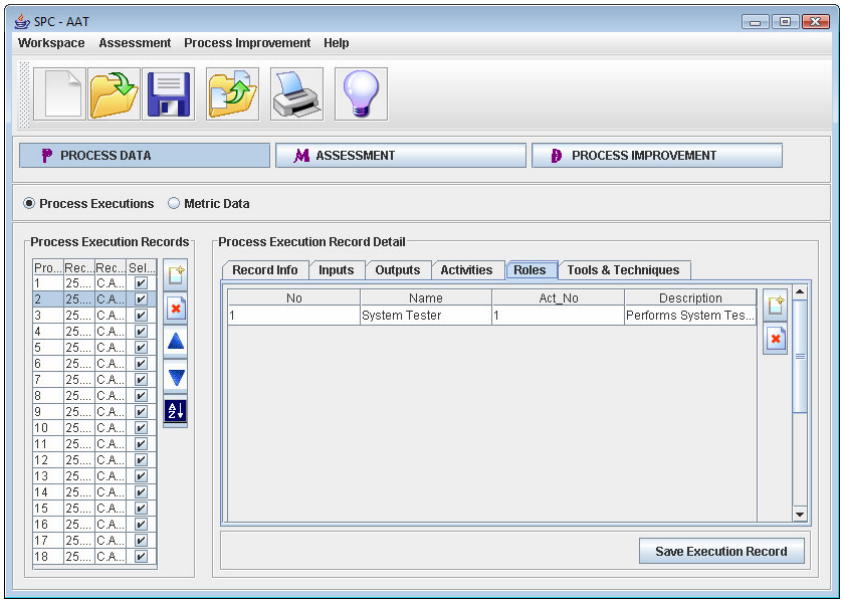


Figure 123 Process Execution Record of Process Execution #2

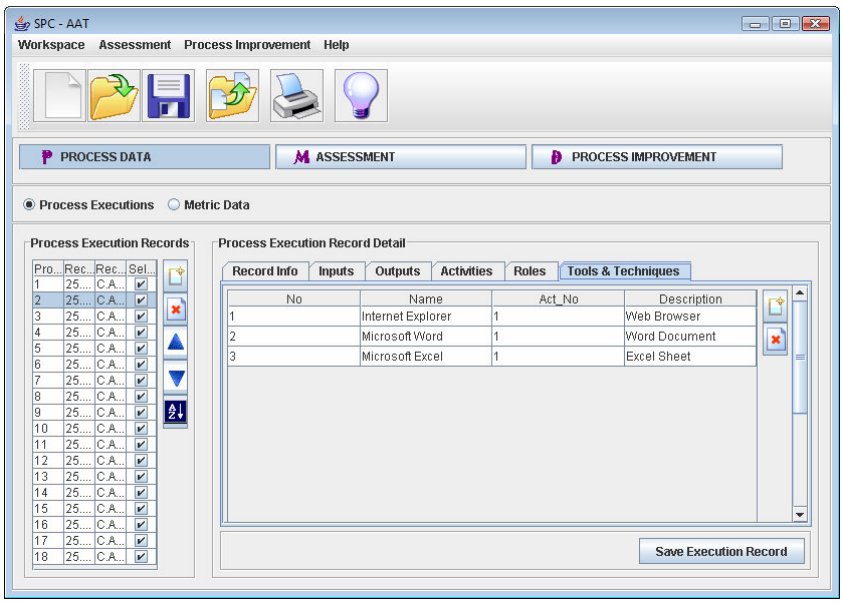


Figure 124 Process Execution Record of Process Execution #2

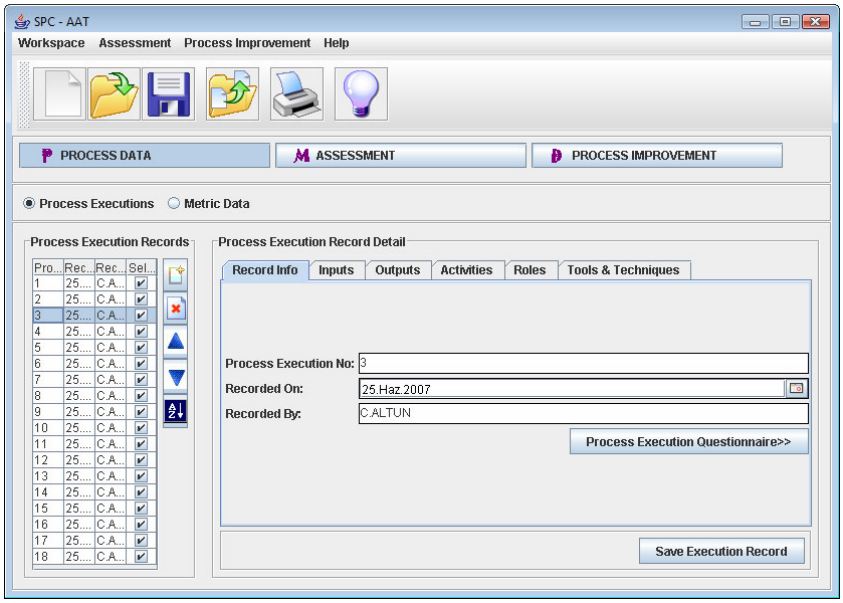


Figure 125 Process Execution Record of Process Execution #3

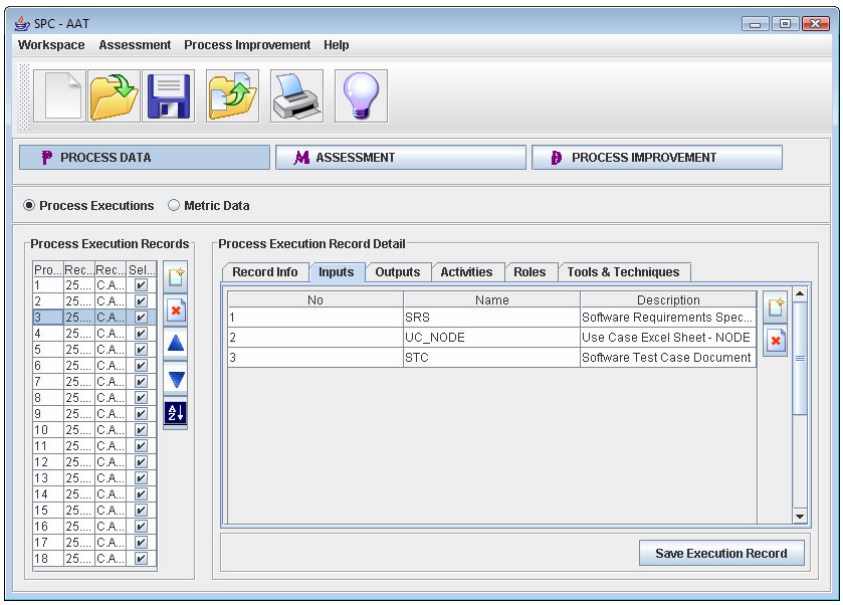


Figure 126 Process Execution Record of Process Execution #3

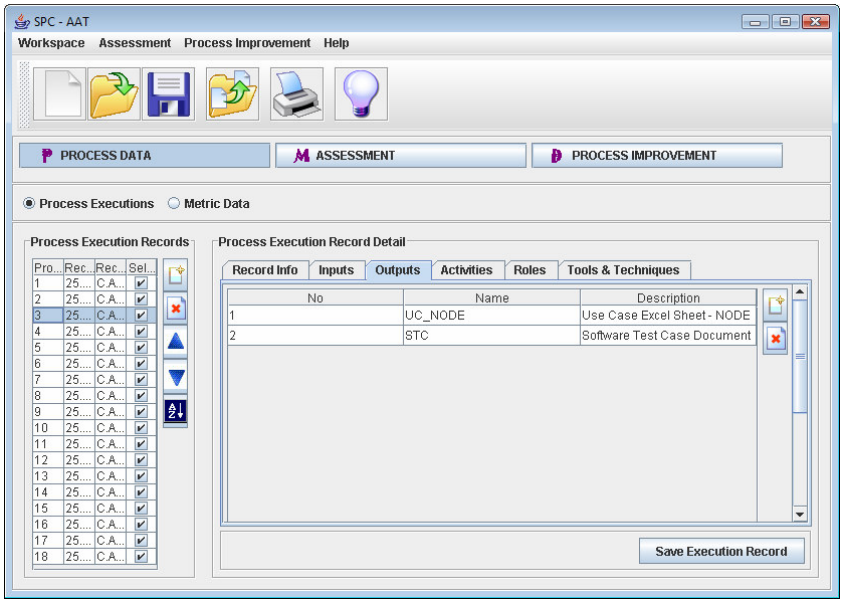


Figure 127 Process Execution Record of Process Execution #3

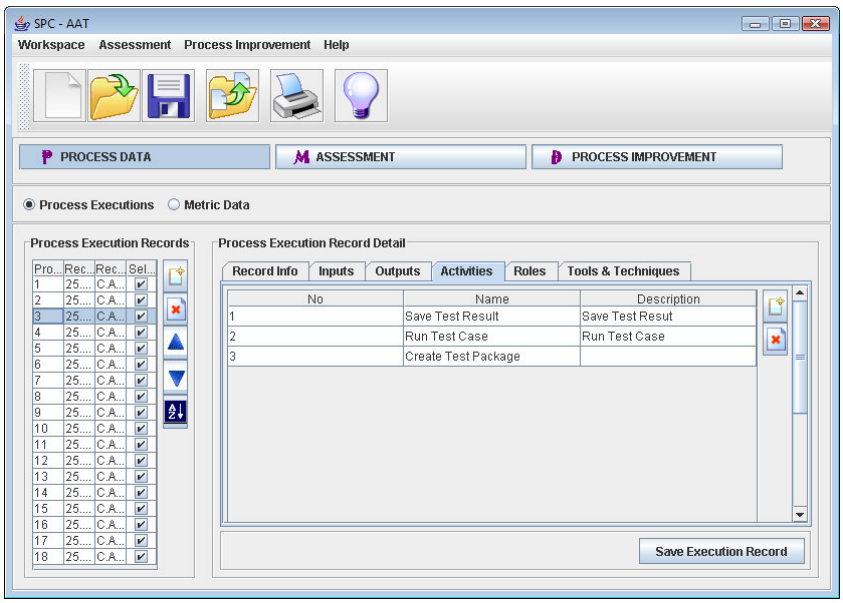


Figure 128 Process Execution Record of Process Execution #3

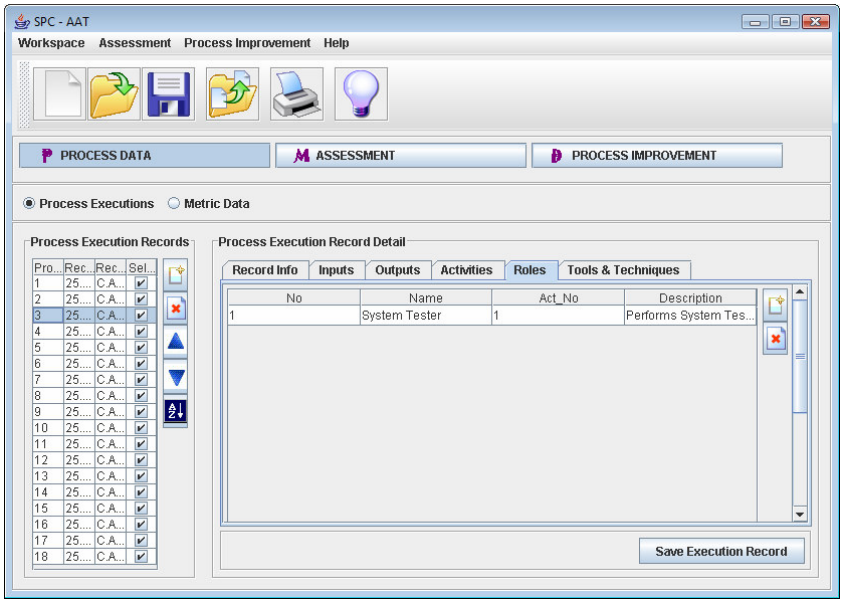


Figure 129 Process Execution Record of Process Execution #3

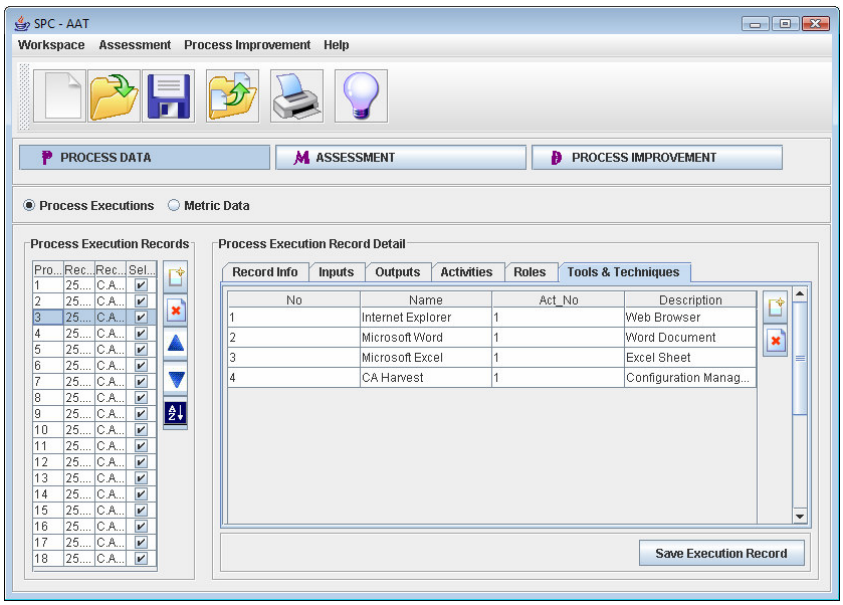


Figure 130 Process Execution Record of Process Execution #3

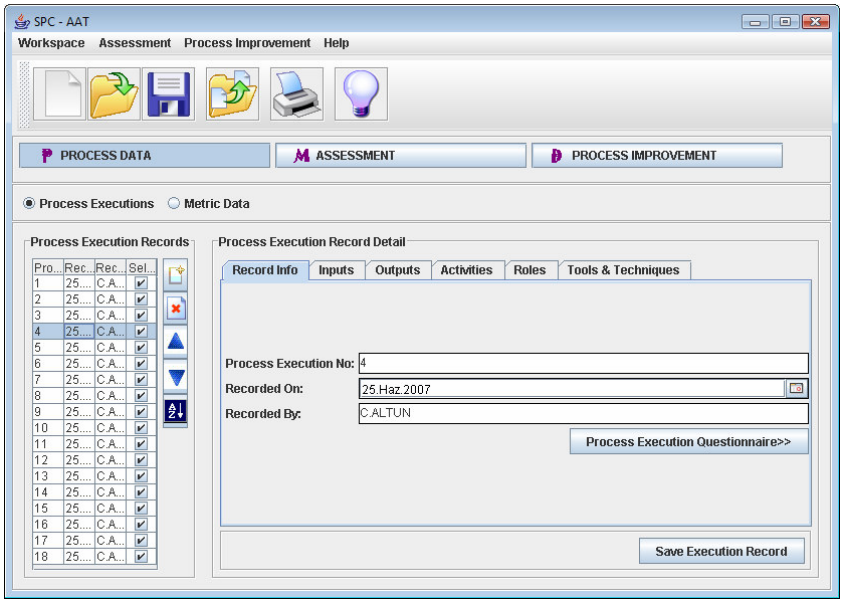


Figure 131 Process Execution Record of Process Execution #4

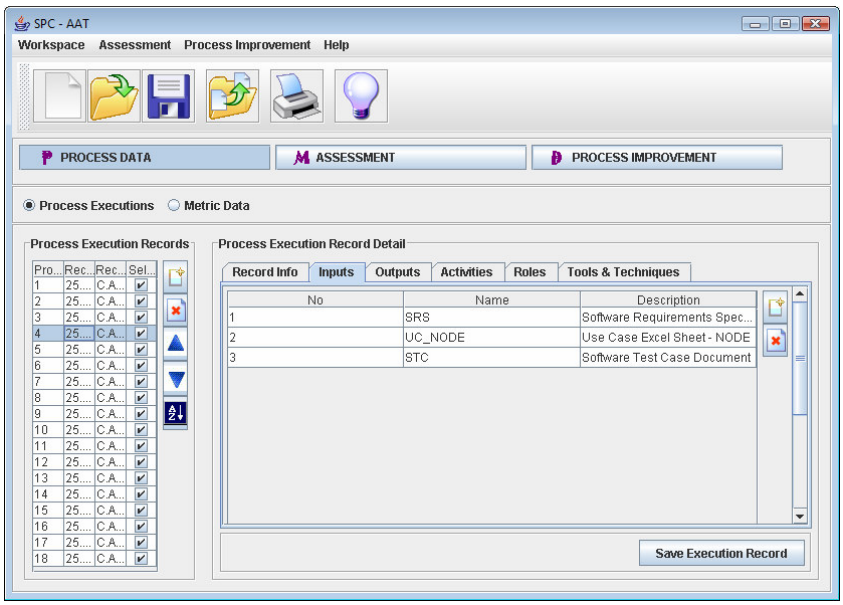


Figure 132 Process Execution Record of Process Execution #4

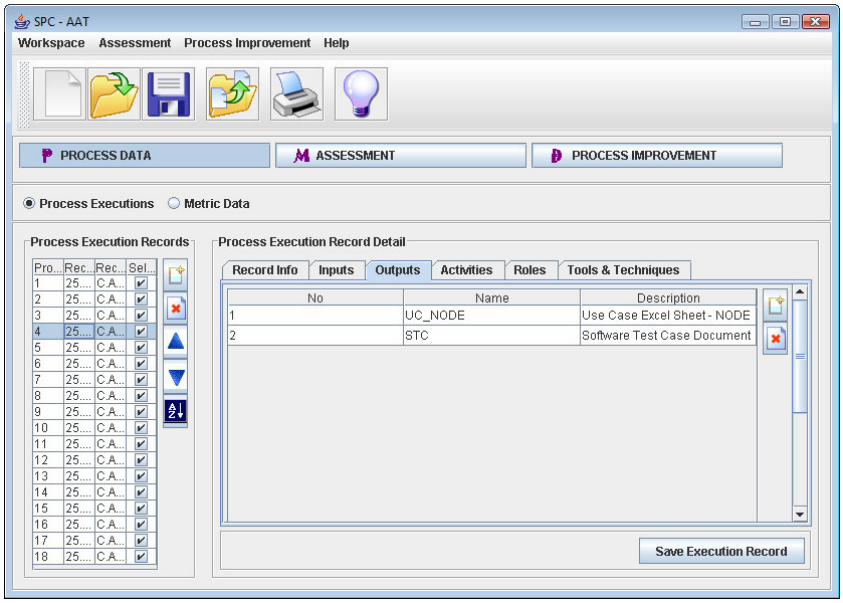


Figure 133 Process Execution Record of Process Execution #4

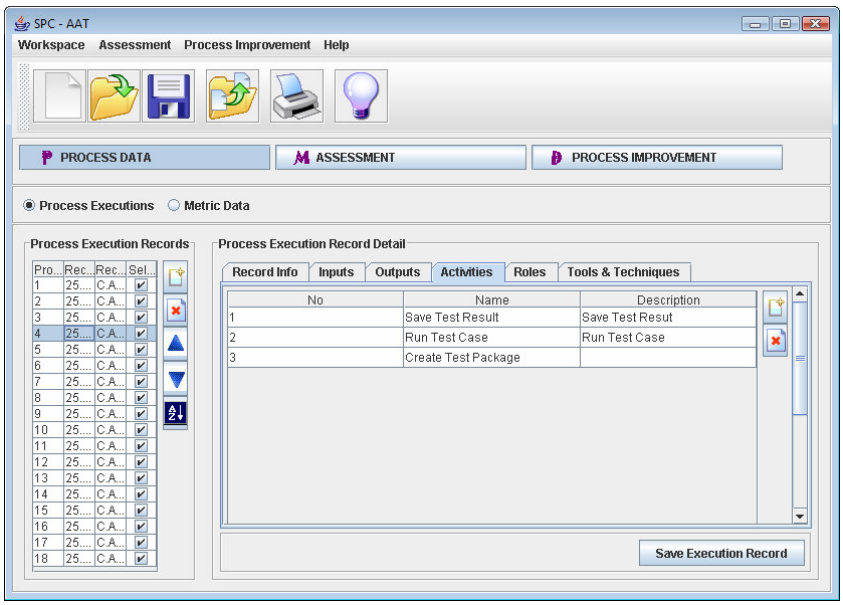


Figure 134 Process Execution Record of Process Execution #4

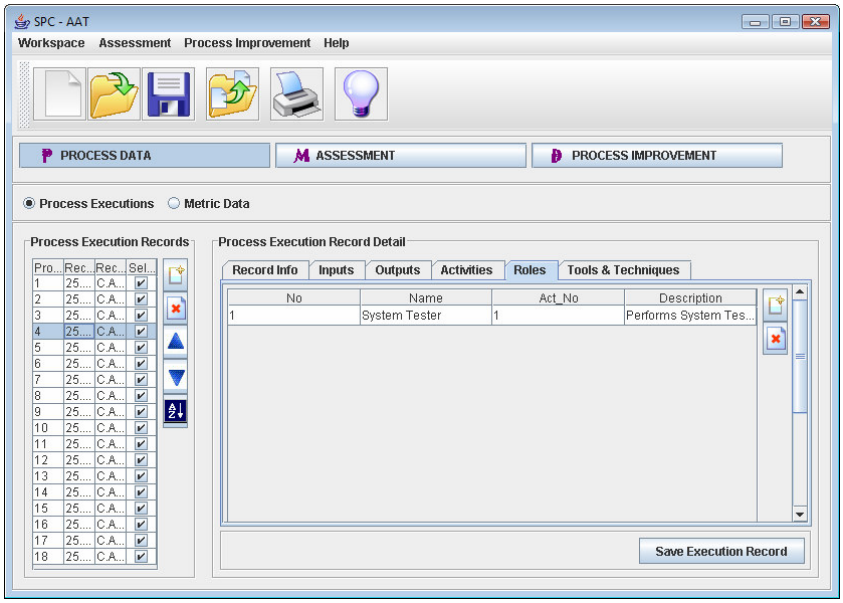


Figure 135 Process Execution Record of Process Execution #4

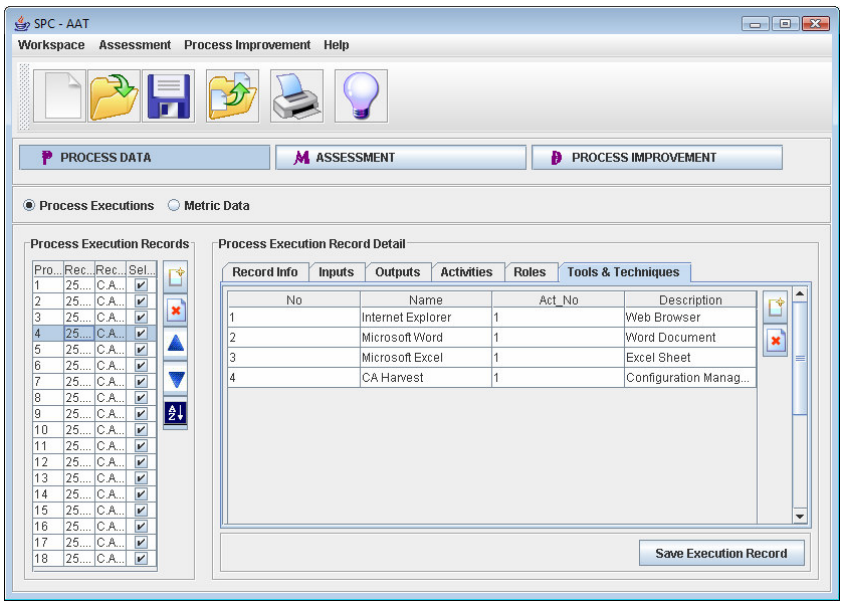


Figure 136 Process Execution Record of Process Execution #4

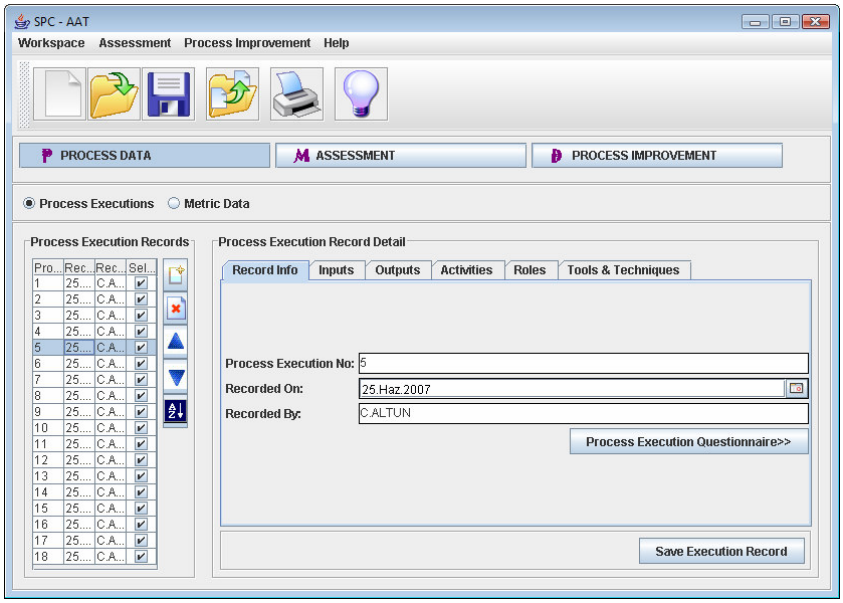


Figure 137 Process Execution Record of Process Execution #5

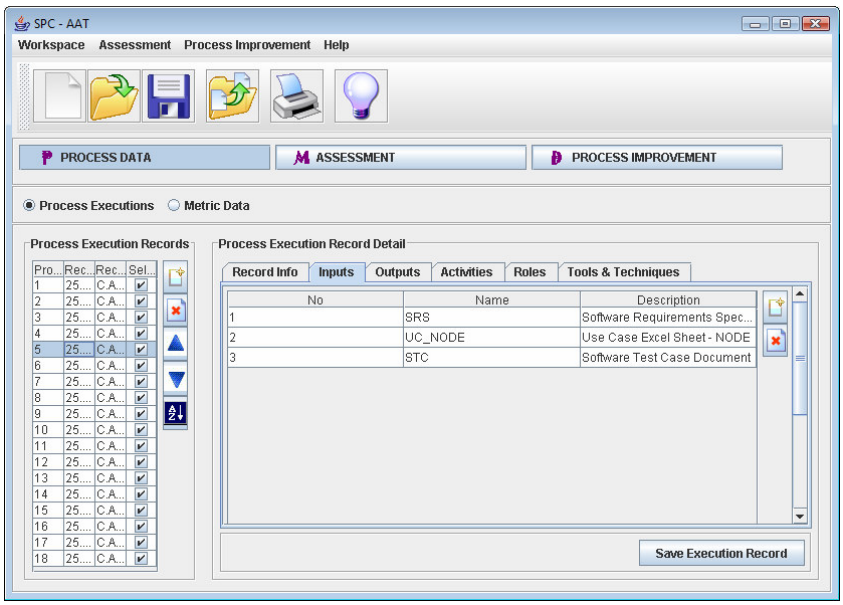


Figure 138 Process Execution Record of Process Execution #5

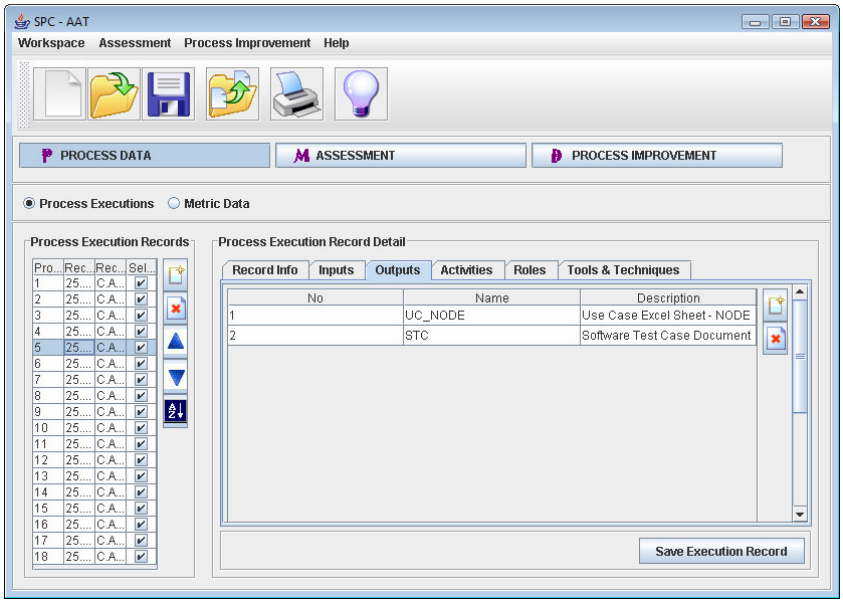


Figure 139 Process Execution Record of Process Execution #5

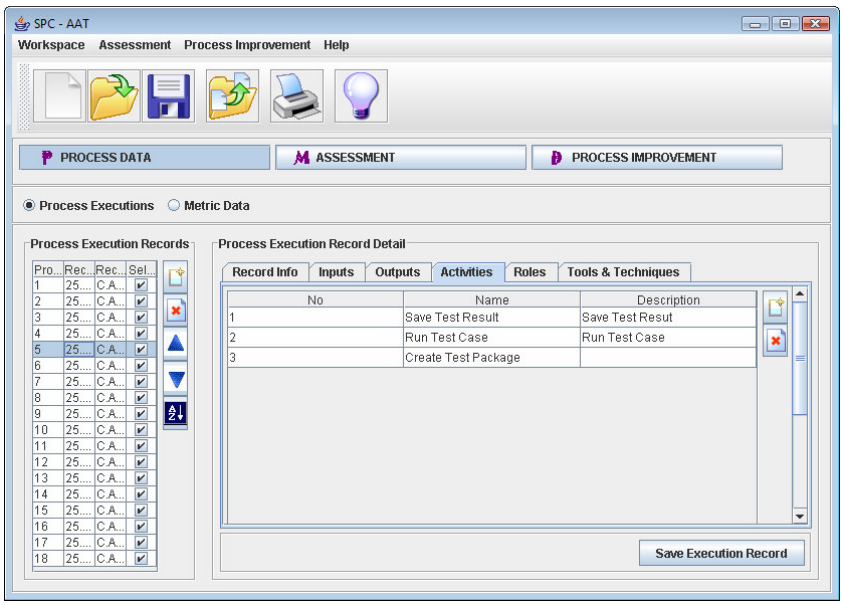


Figure 140 Process Execution Record of Process Execution #5

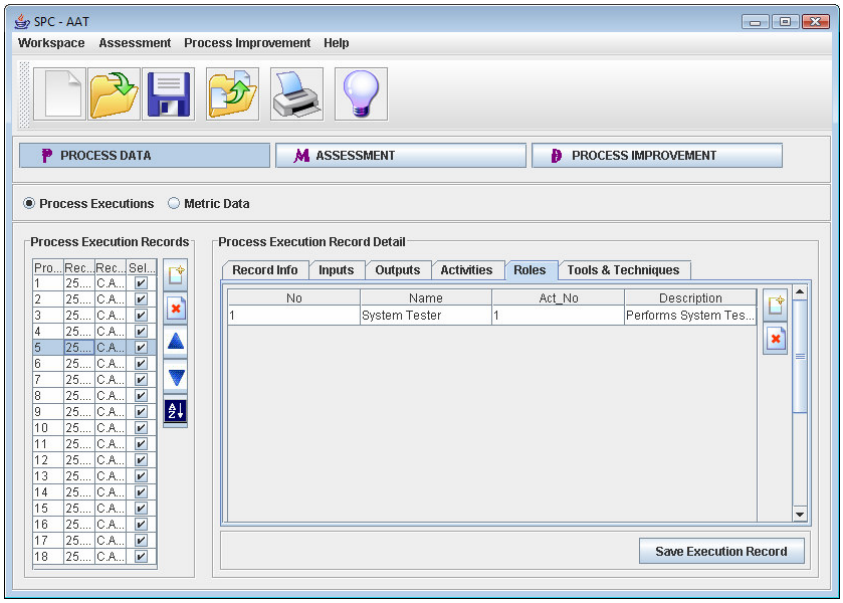


Figure 141 Process Execution Record of Process Execution #5

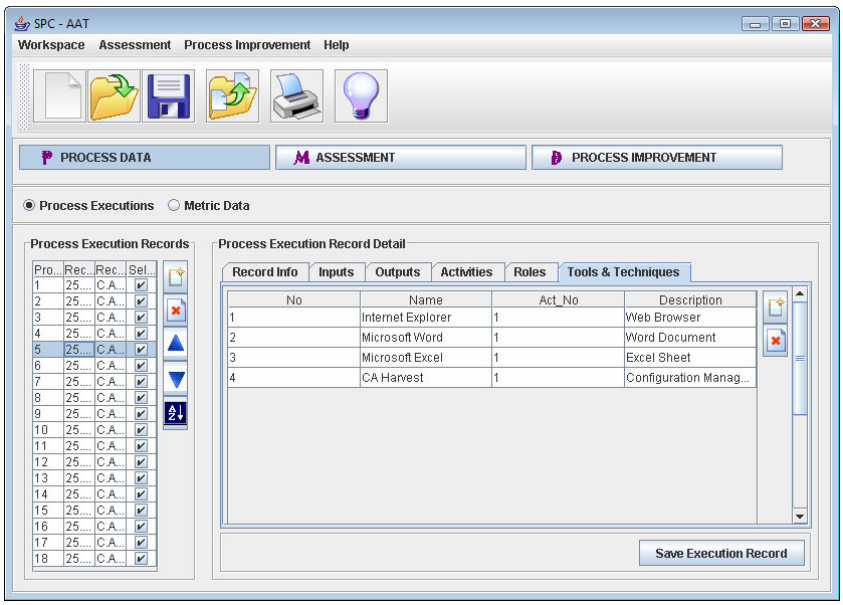


Figure 142 Process Execution Record of Process Execution #5

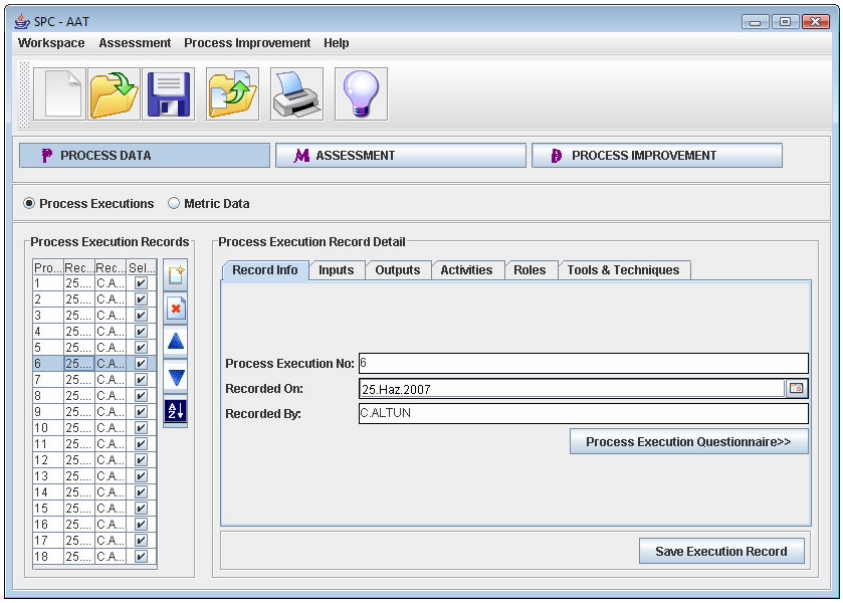


Figure 143 Process Execution Record of Process Execution #6

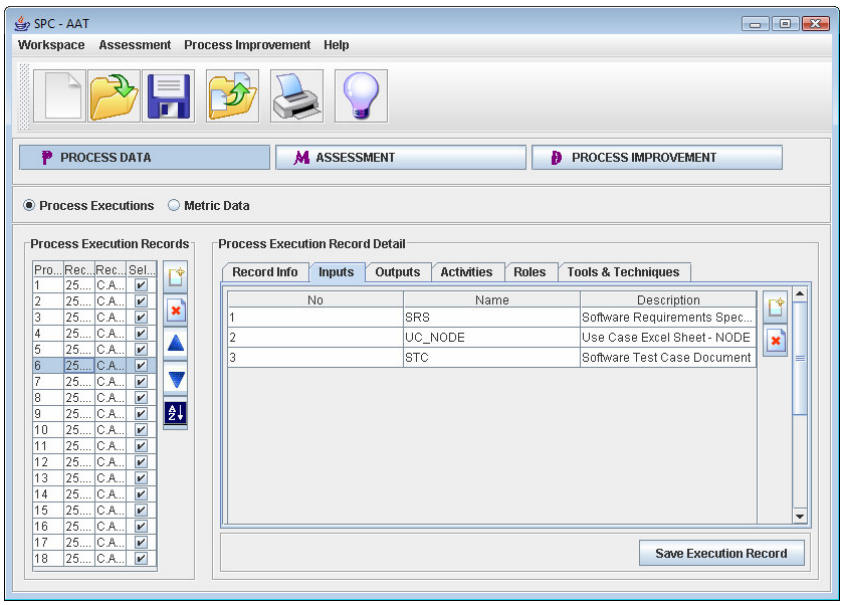


Figure 144 Process Execution Record of Process Execution #6

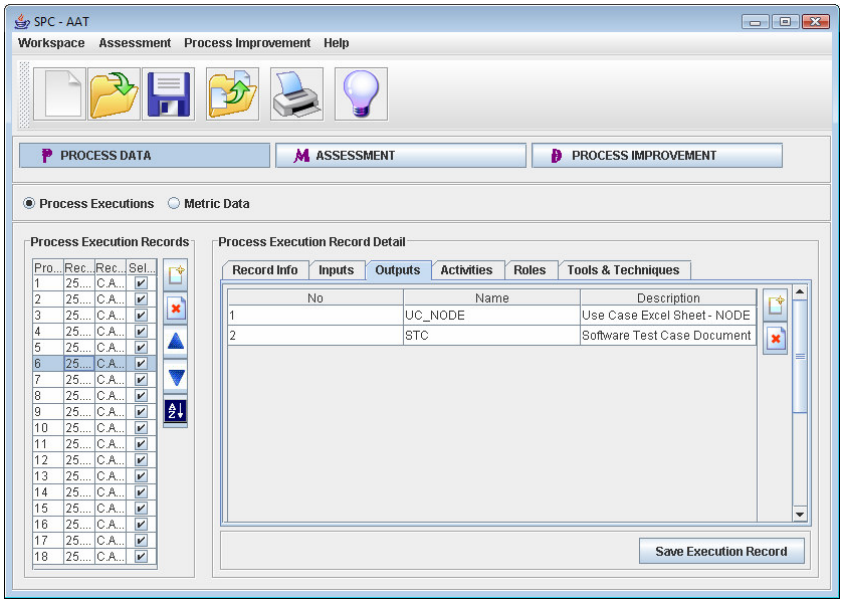


Figure 145 Process Execution Record of Process Execution #6

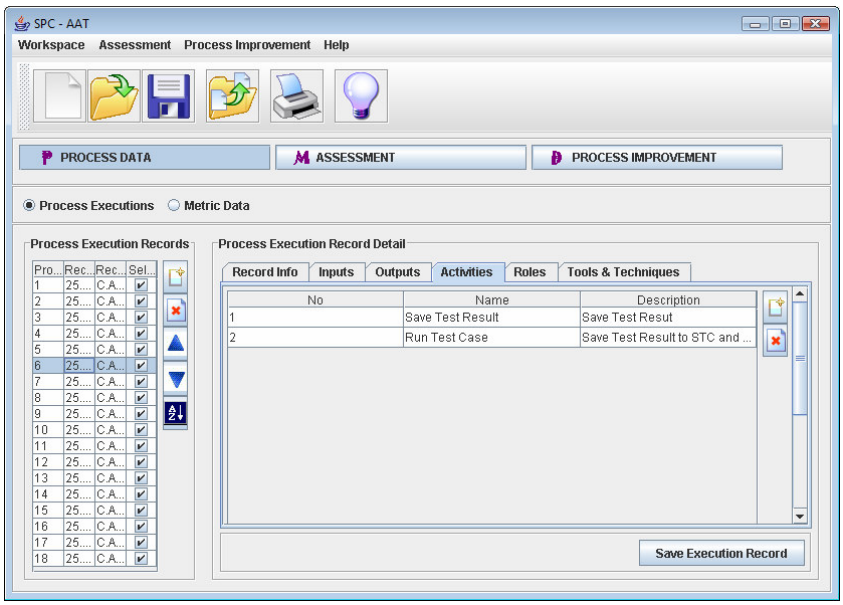


Figure 146 Process Execution Record of Process Execution #6

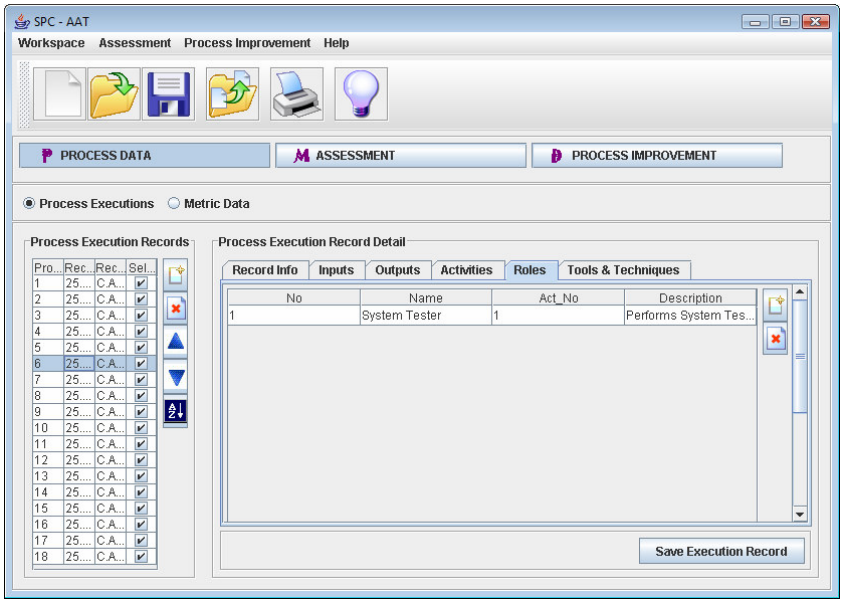


Figure 147 Process Execution Record of Process Execution #6

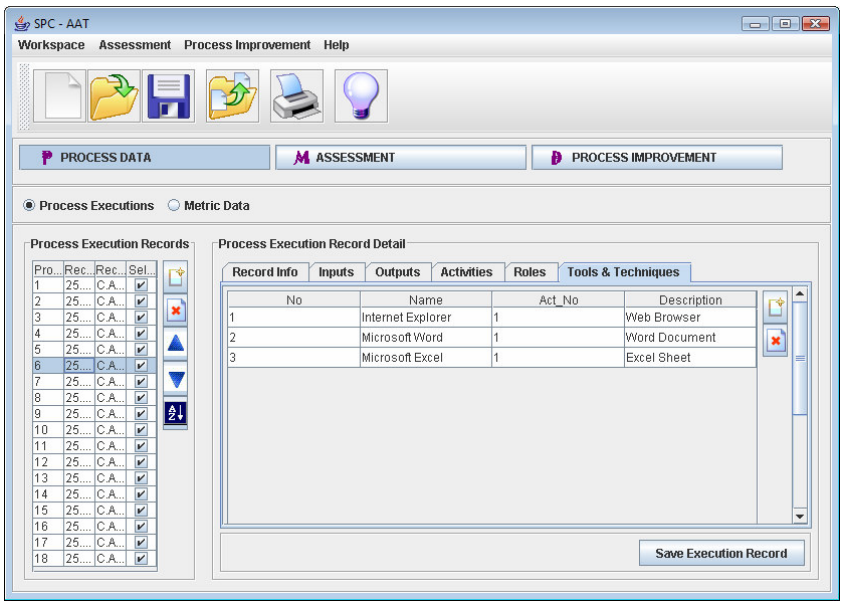


Figure 148 Process Execution Record of Process Execution #6

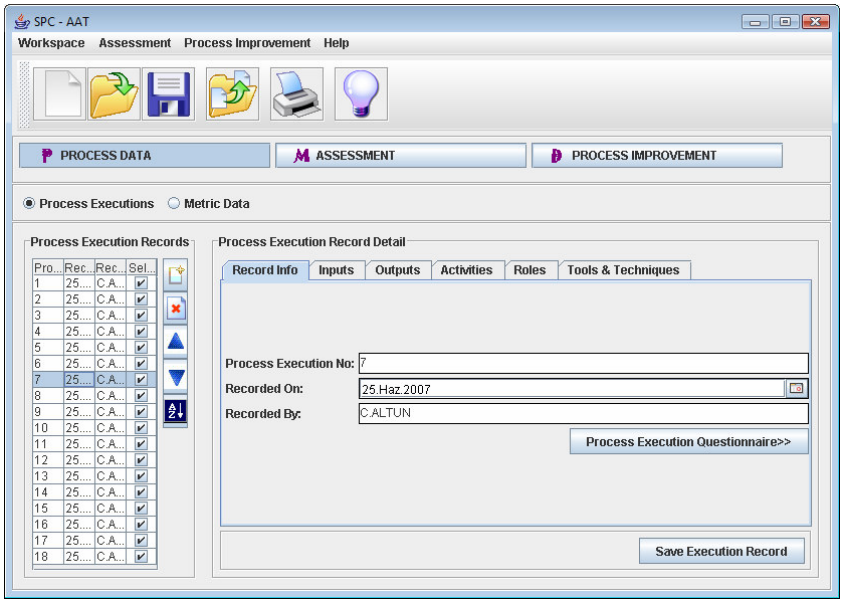


Figure 149 Process Execution Record of Process Execution #7

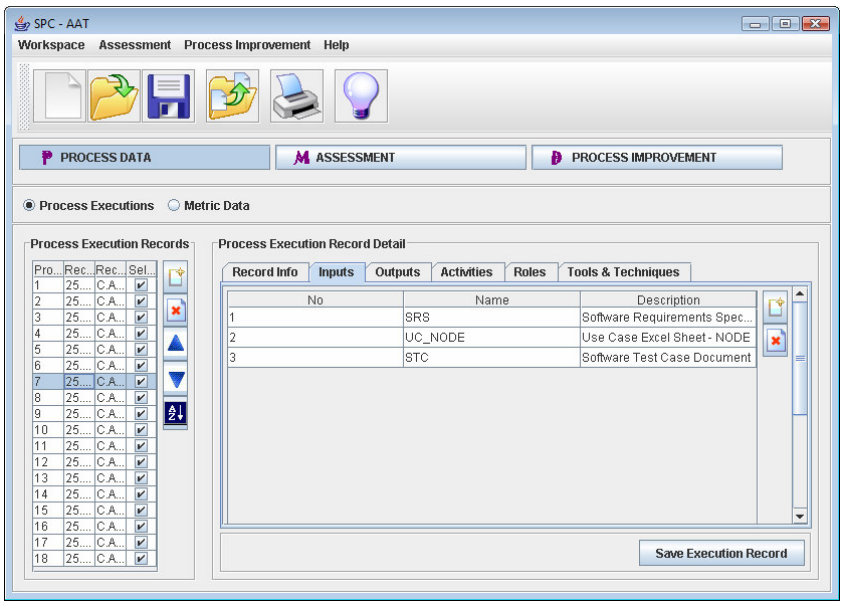


Figure 150 Process Execution Record of Process Execution #7

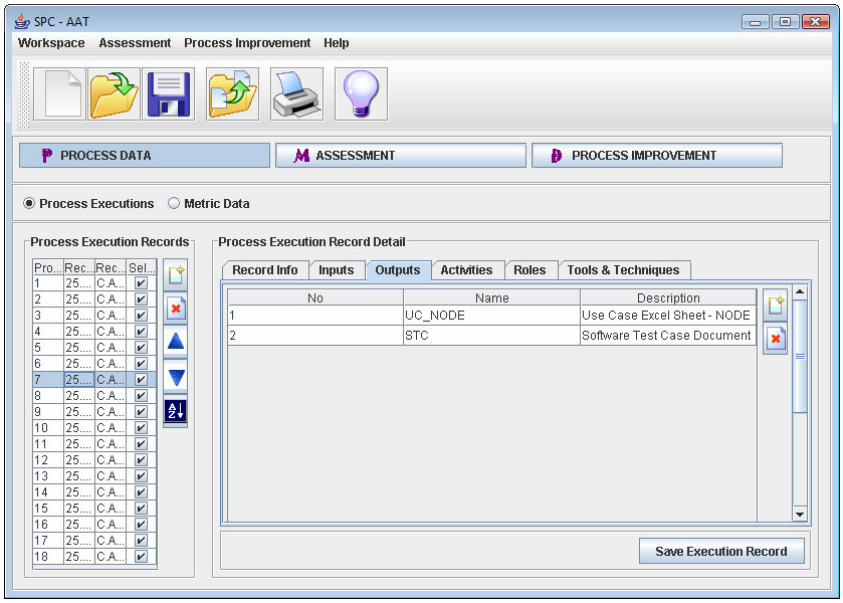


Figure 151 Process Execution Record of Process Execution #7

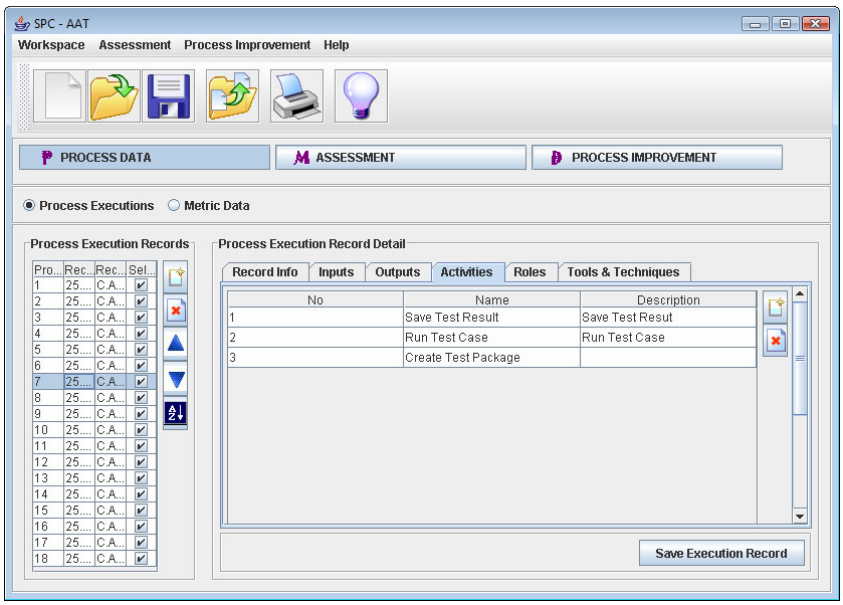


Figure 152 Process Execution Record of Process Execution #7

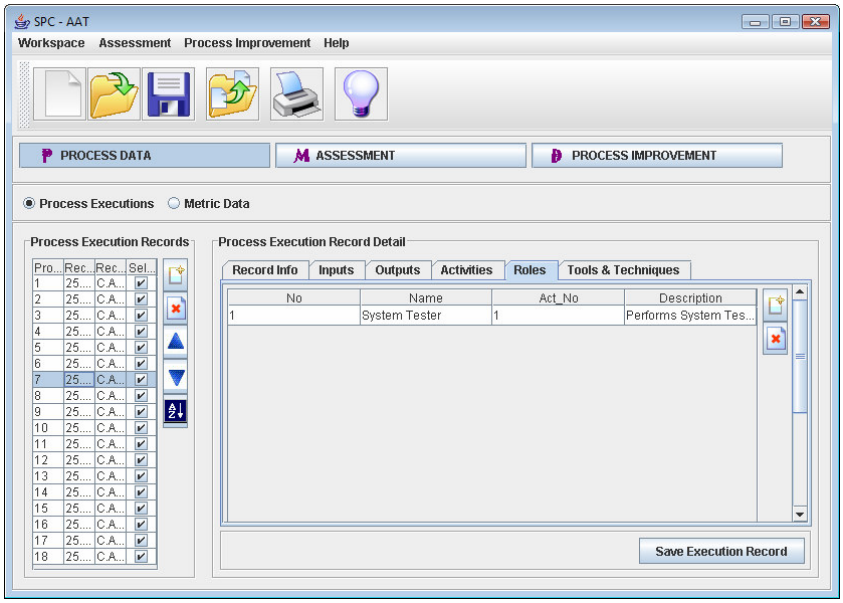


Figure 153 Process Execution Record of Process Execution #7

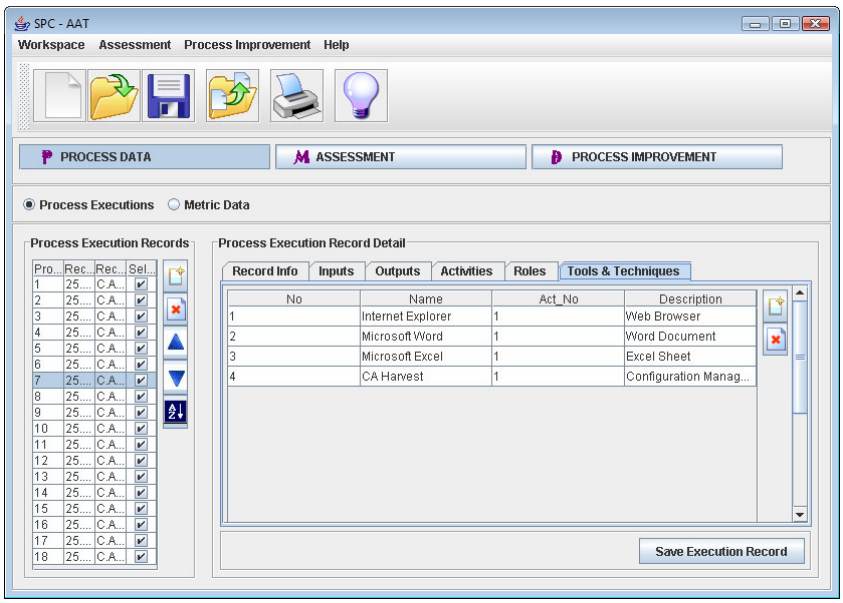


Figure 154 Process Execution Record of Process Execution #7

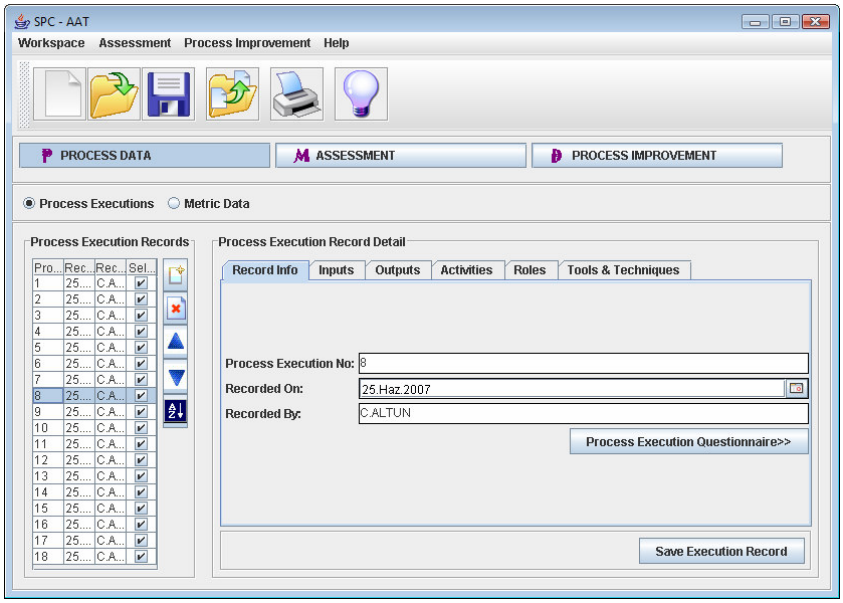


Figure 155 Process Execution Record of Process Execution #8

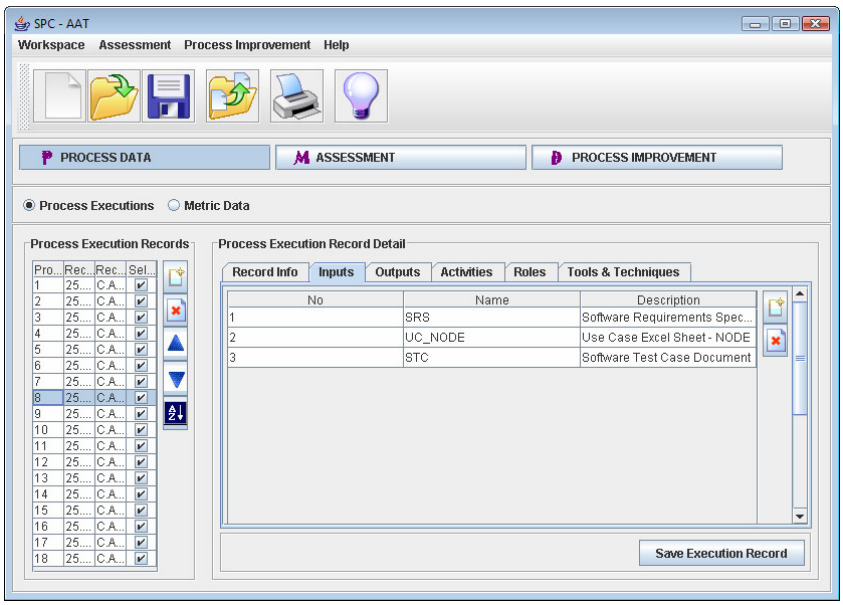


Figure 156 Process Execution Record of Process Execution #8

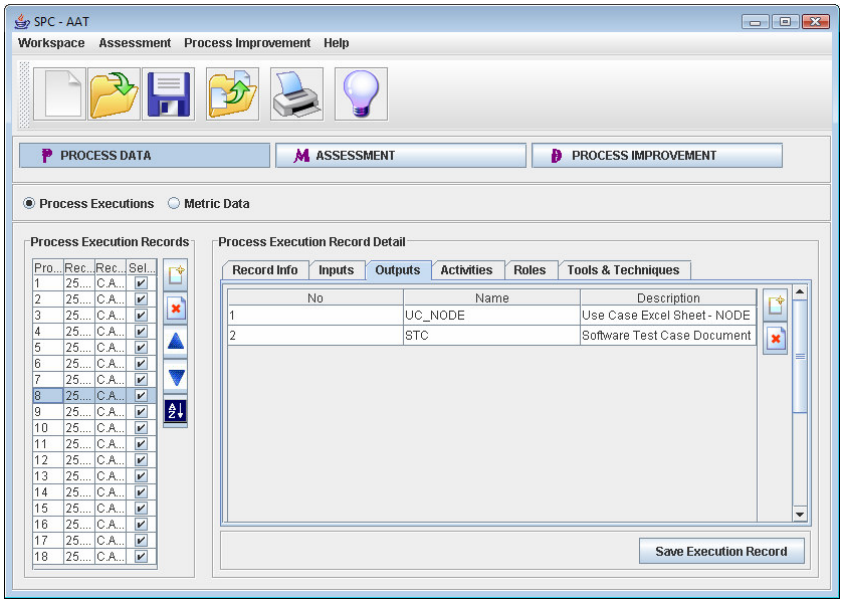


Figure 157 Process Execution Record of Process Execution #8

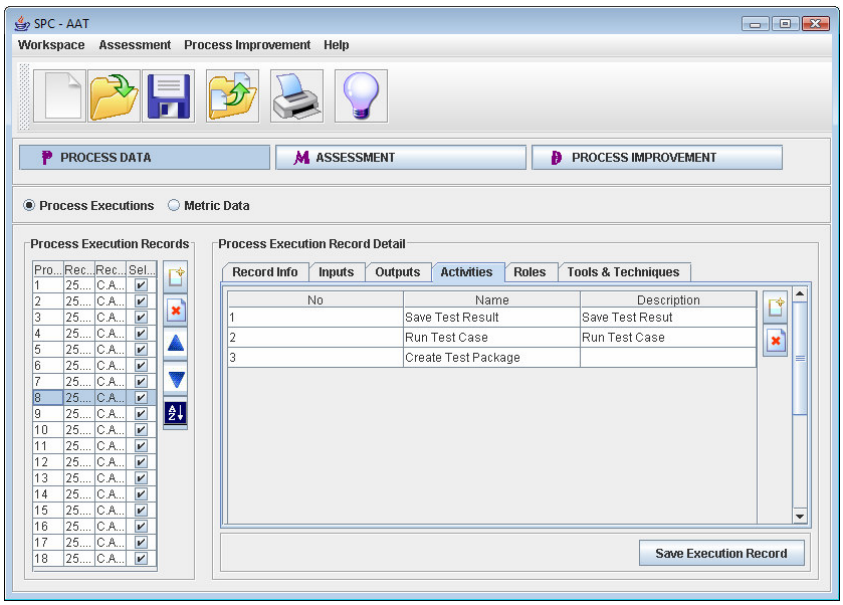


Figure 158 Process Execution Record of Process Execution #8

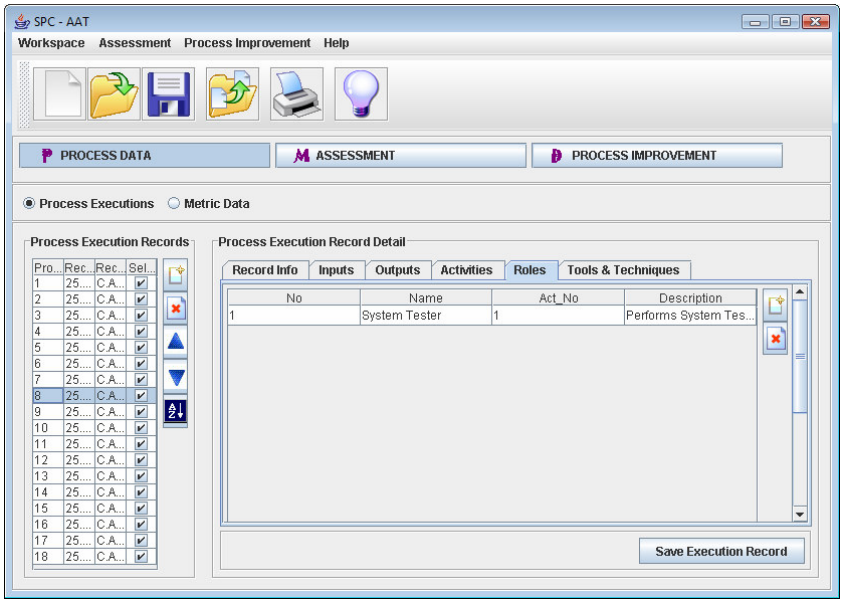


Figure 159 Process Execution Record of Process Execution #8

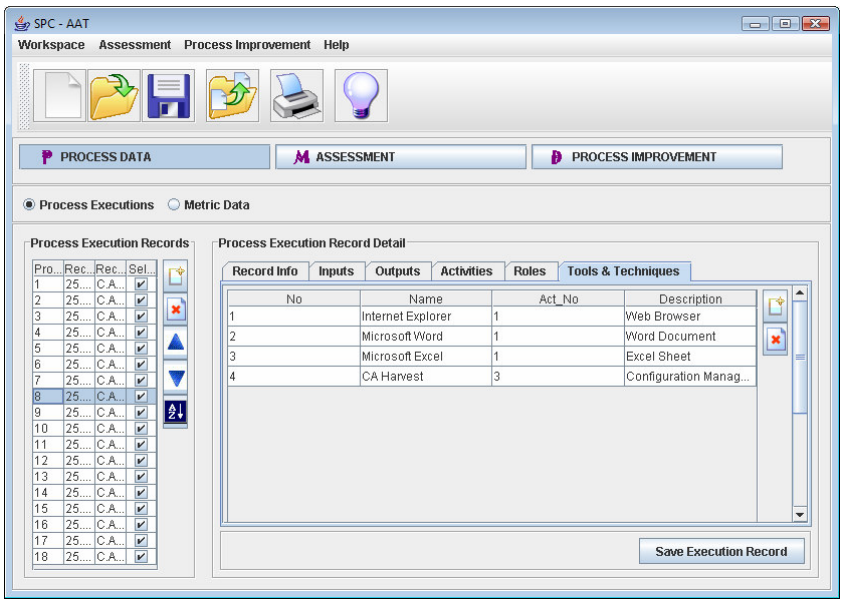


Figure 160 Process Execution Record of Process Execution #8

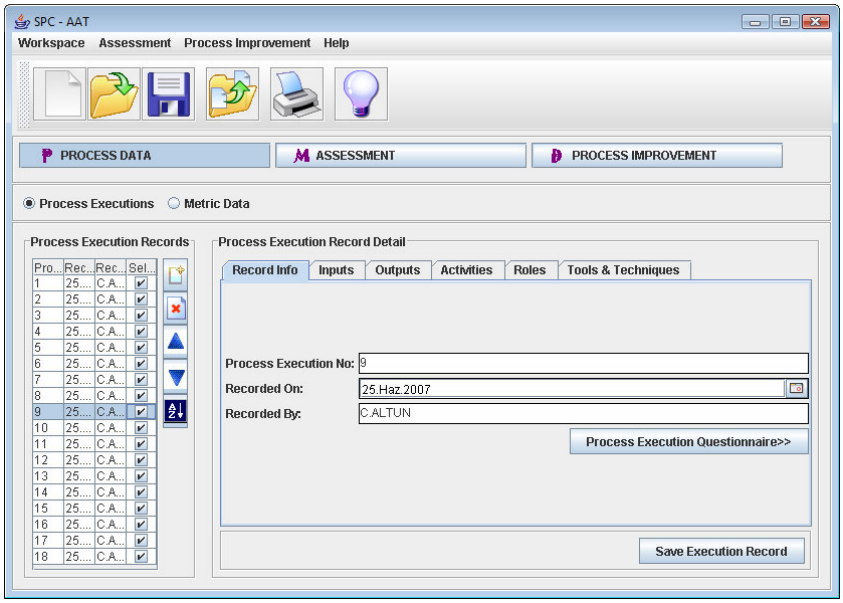


Figure 161 Process Execution Record of Process Execution #9

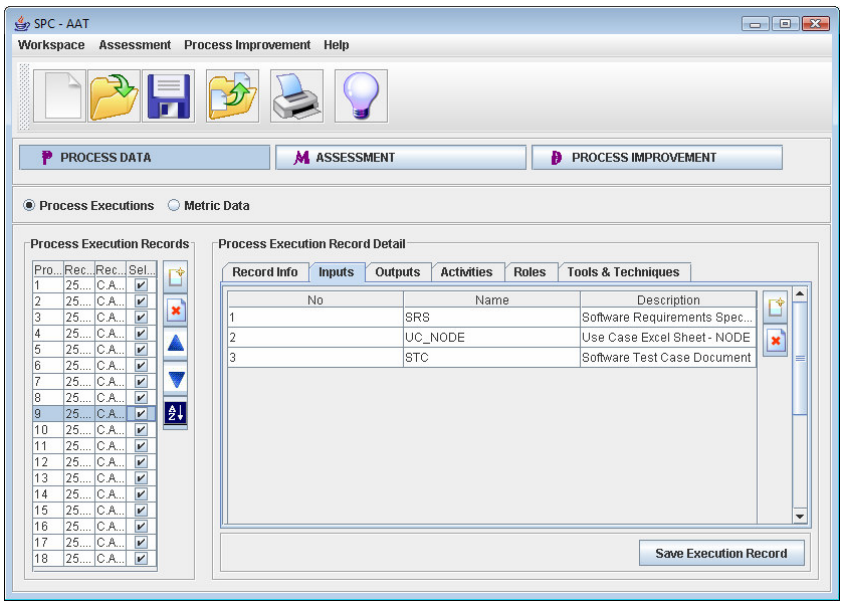


Figure 162 Process Execution Record of Process Execution #9

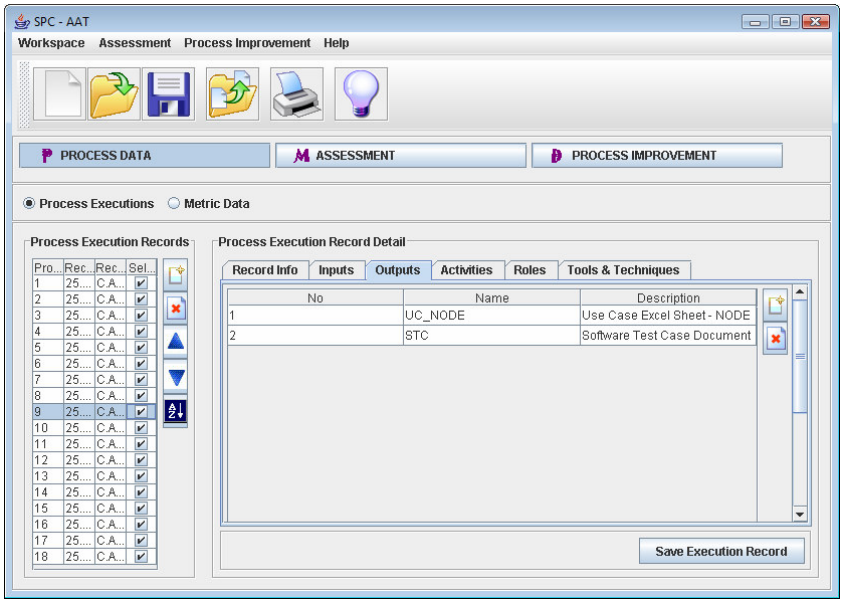


Figure 163 Process Execution Record of Process Execution #9

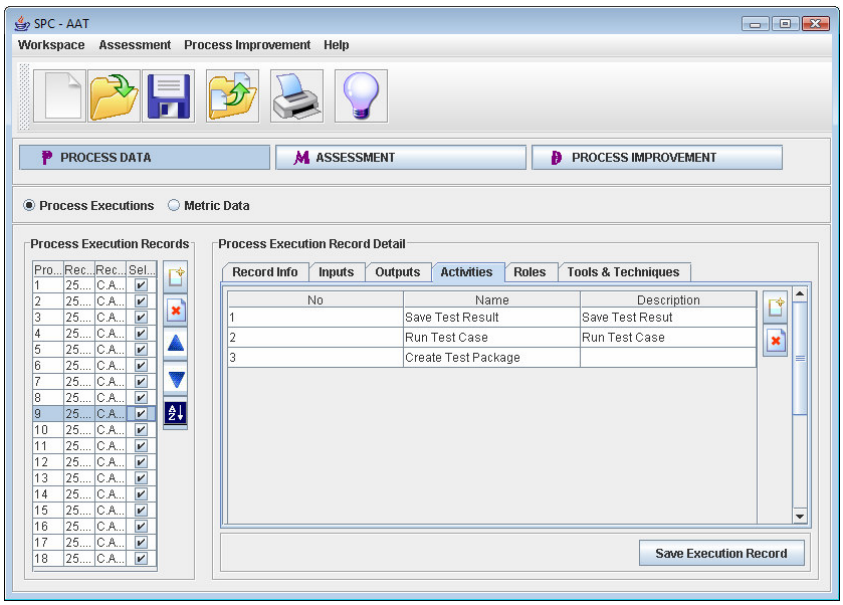


Figure 164 Process Execution Record of Process Execution #9

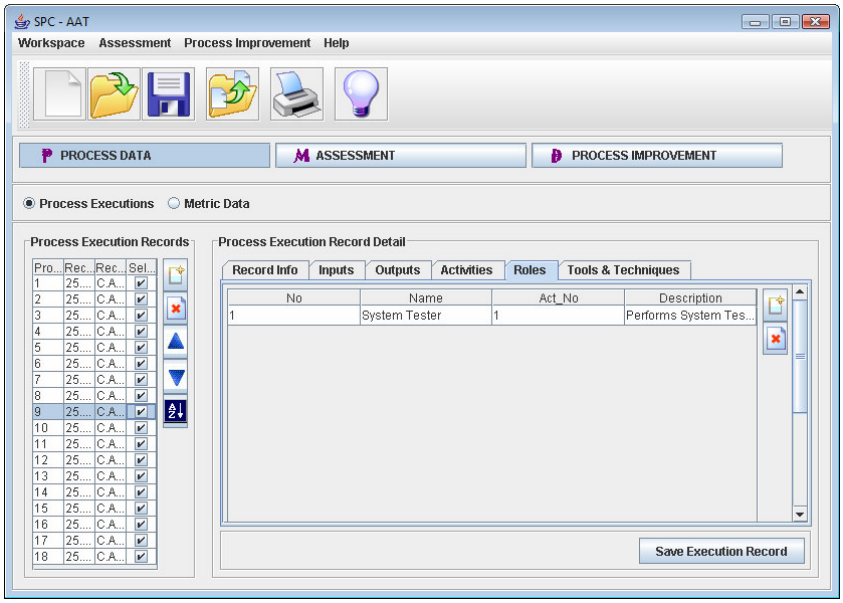


Figure 165 Process Execution Record of Process Execution #9

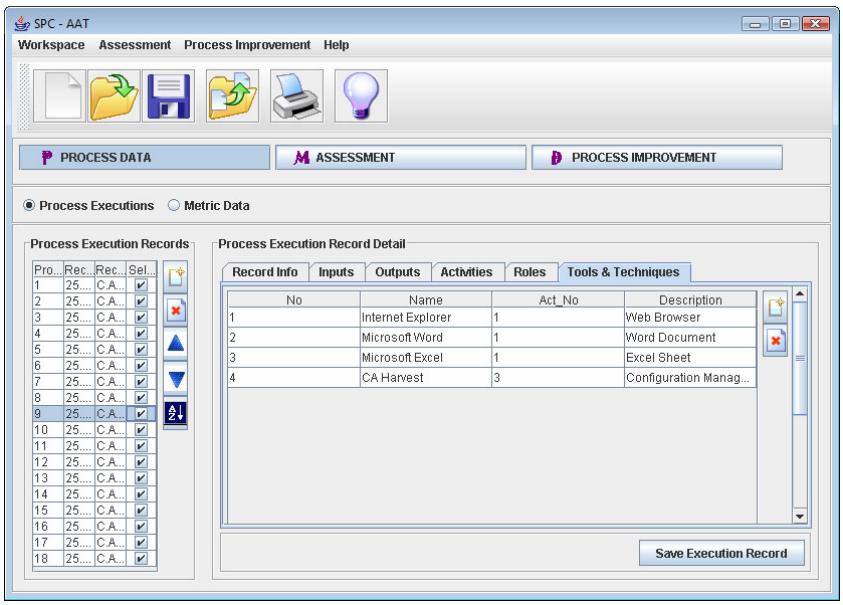


Figure 166 Process Execution Record of Process Execution #9

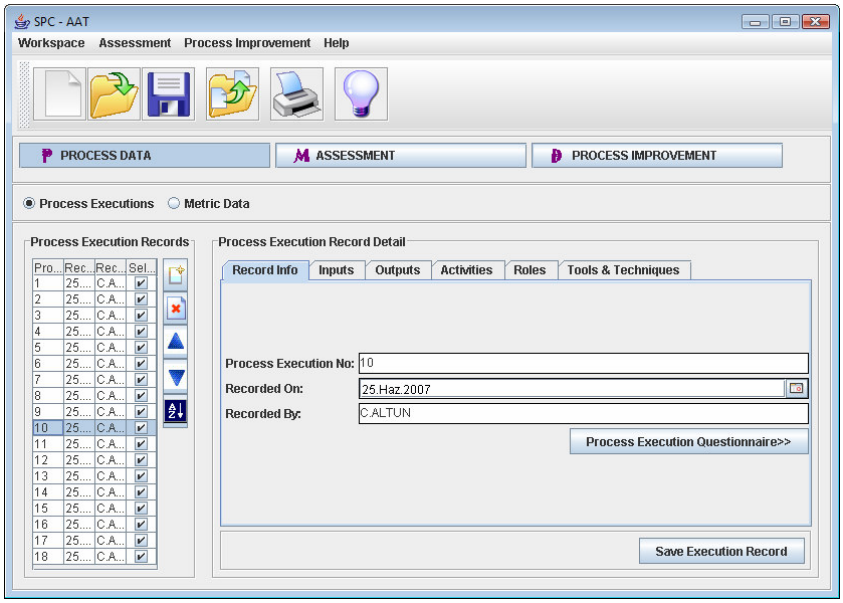


Figure 167 Process Execution Record of Process Execution #10

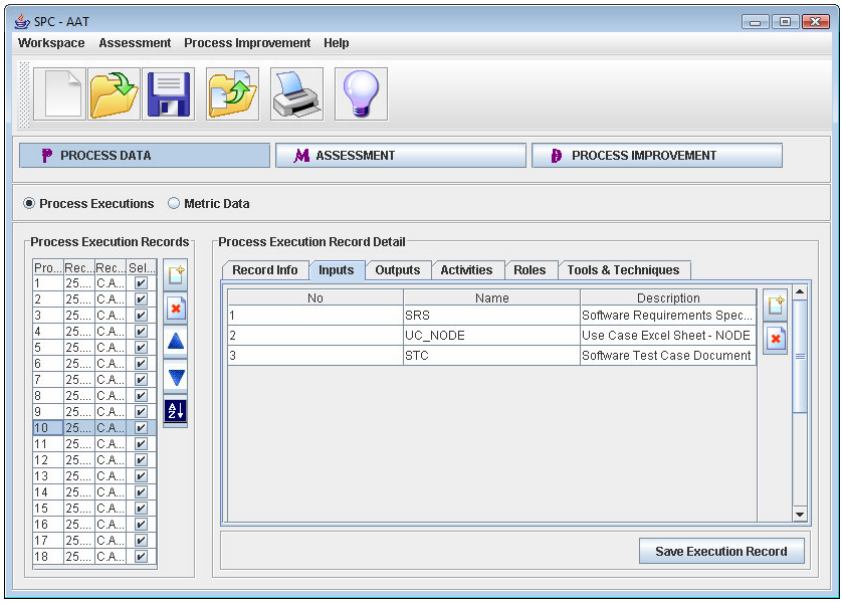


Figure 168 Process Execution Record of Process Execution #10

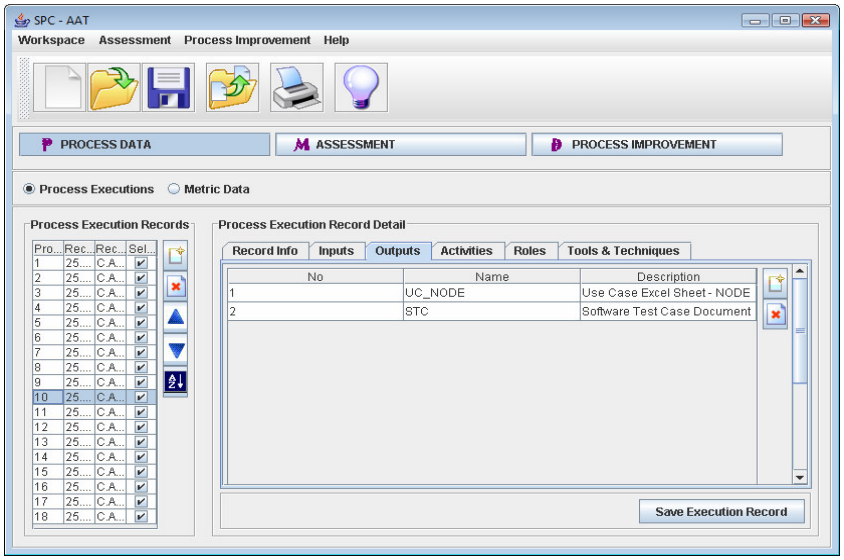


Figure 169 Process Execution Record of Process Execution #10

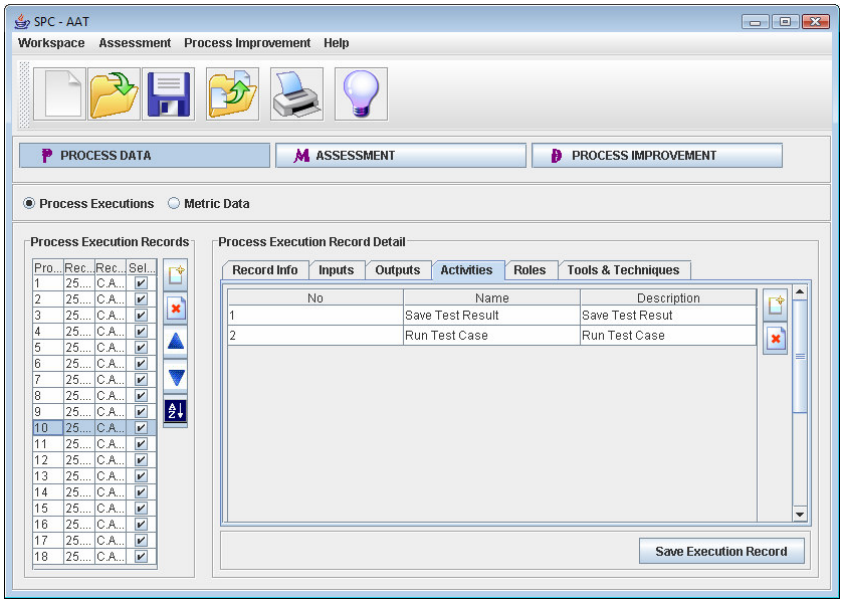


Figure 170 Process Execution Record of Process Execution #10

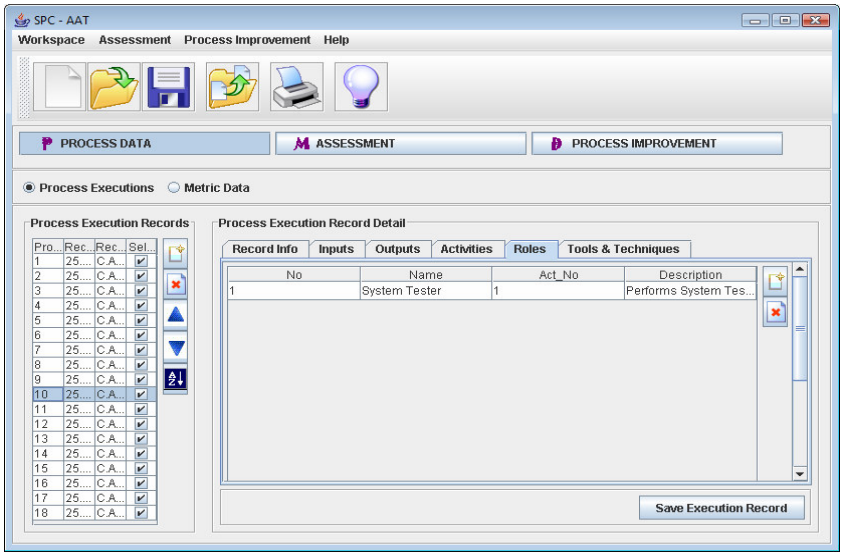


Figure 171 Process Execution Record of Process Execution #10

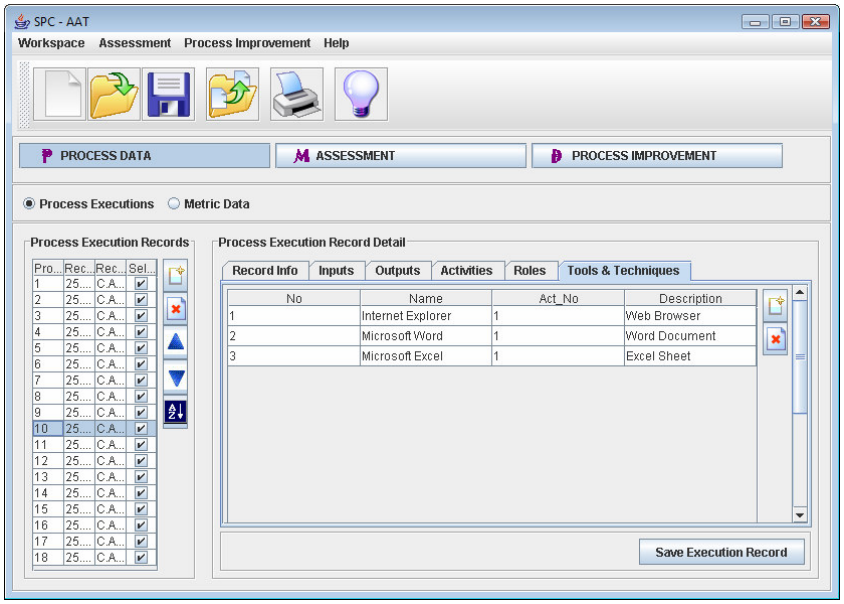


Figure 172 Process Execution Record of Process Execution #10

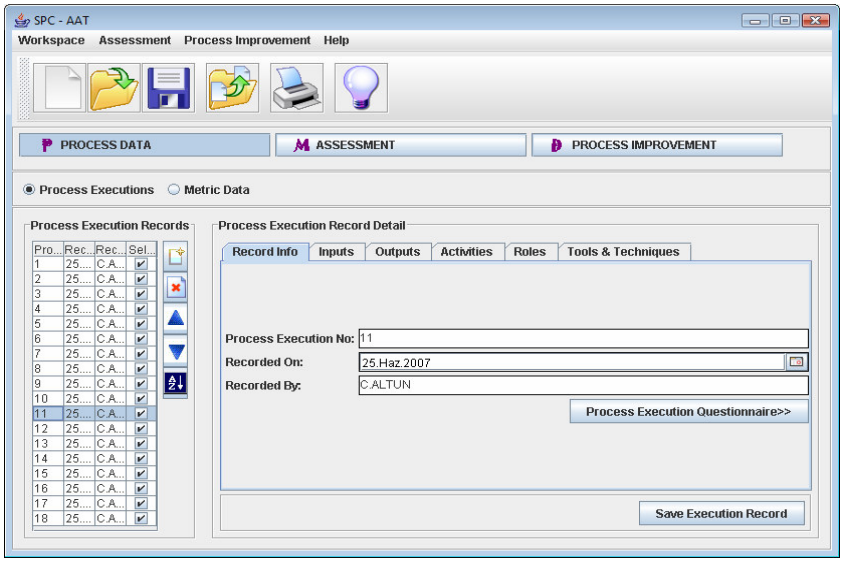


Figure 173 Process Execution Record of Process Execution #11

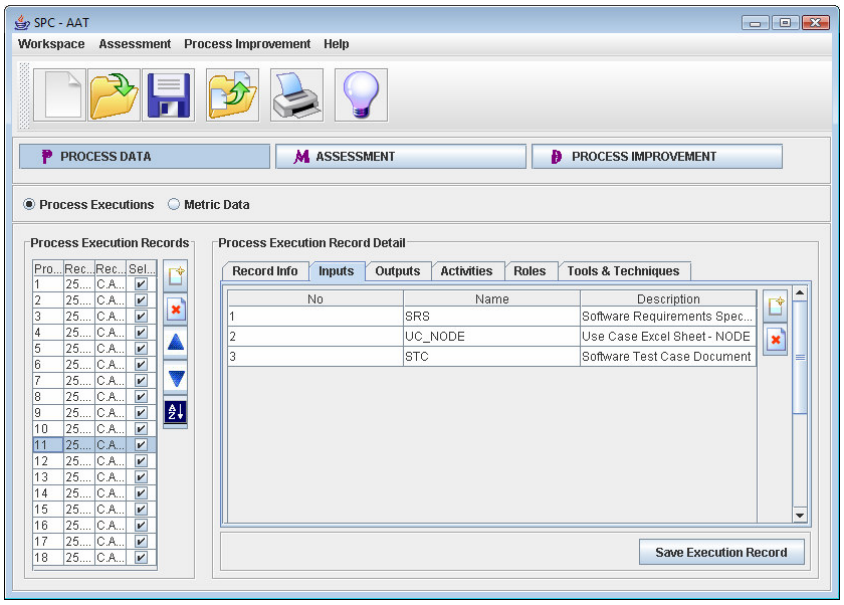


Figure 174 Process Execution Record of Process Execution #11

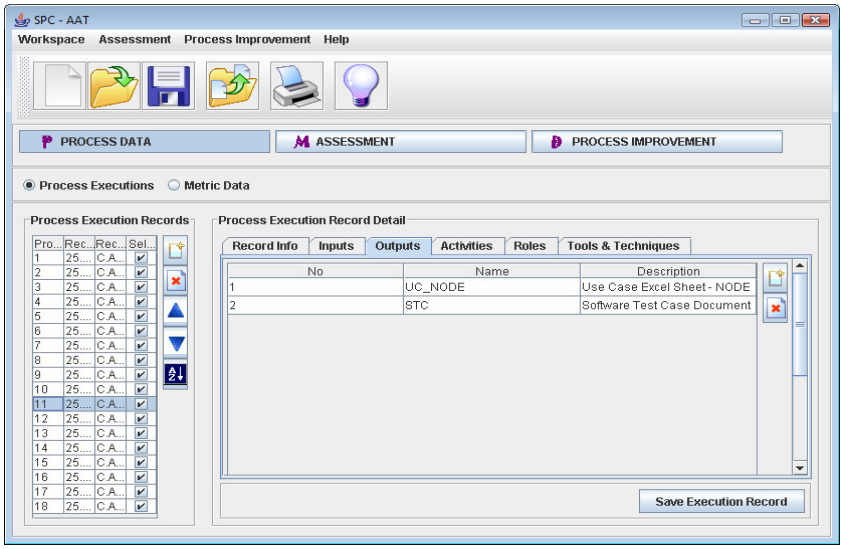


Figure 175 Process Execution Record of Process Execution #11

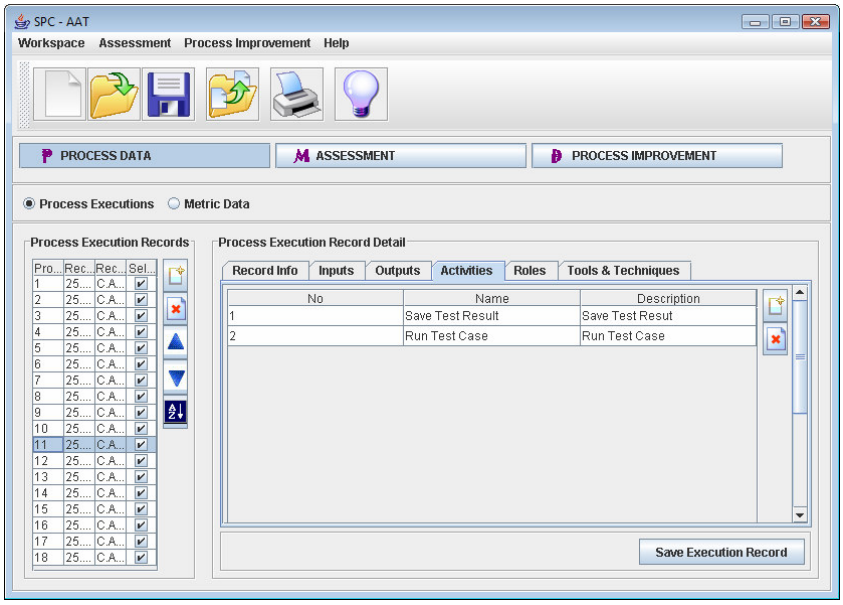


Figure 176 Process Execution Record of Process Execution #11

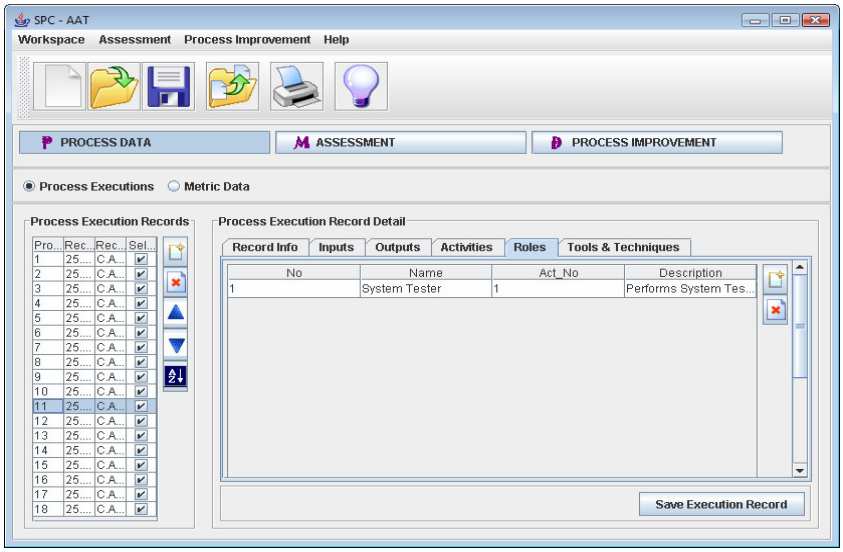


Figure 177 Process Execution Record of Process Execution #11

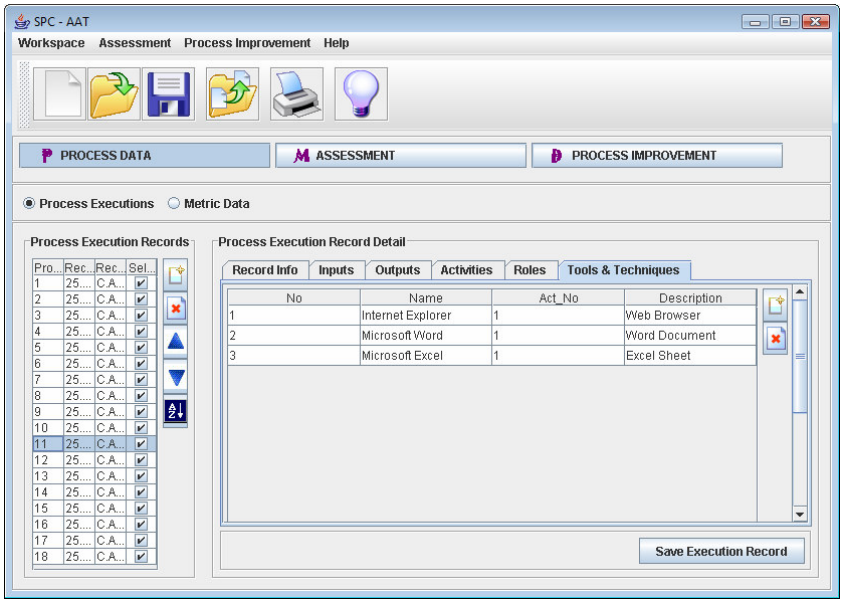


Figure 178 Process Execution Record of Process Execution #11

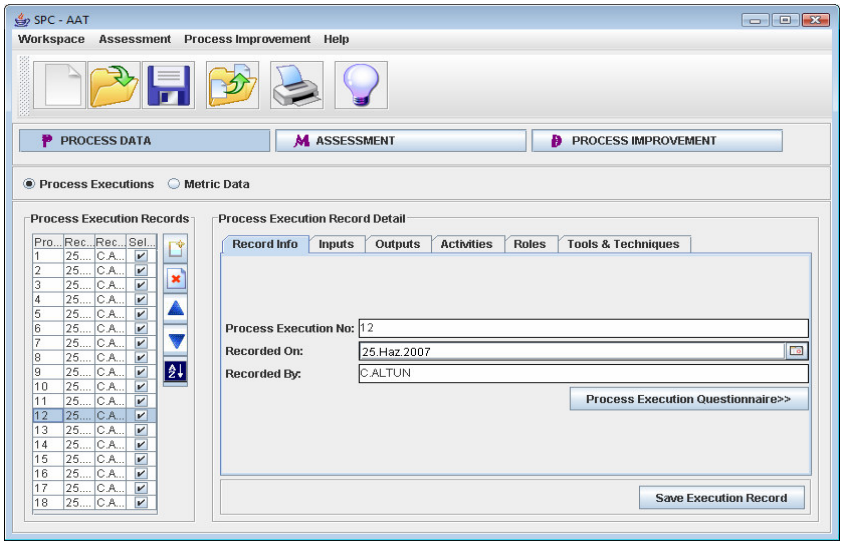


Figure 179 Process Execution Record of Process Execution #12

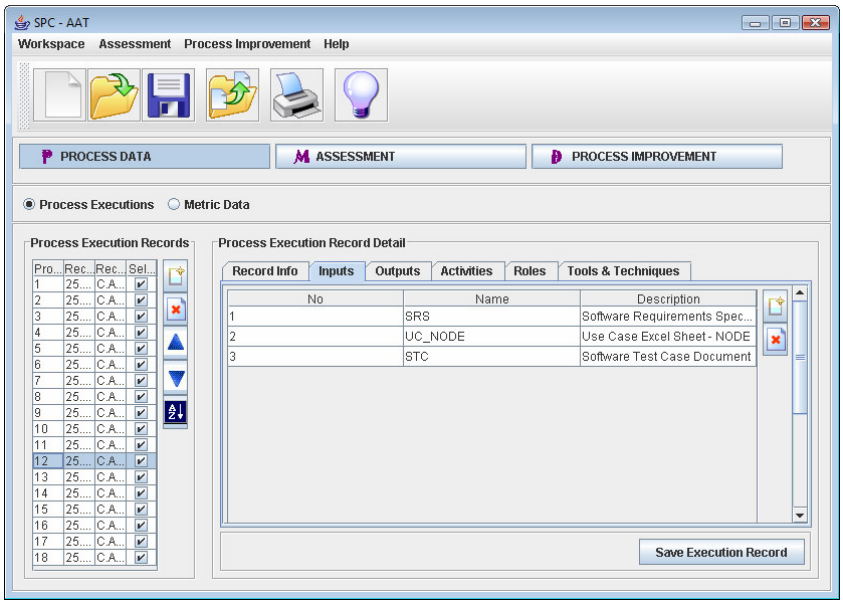


Figure 180 Process Execution Record of Process Execution #12

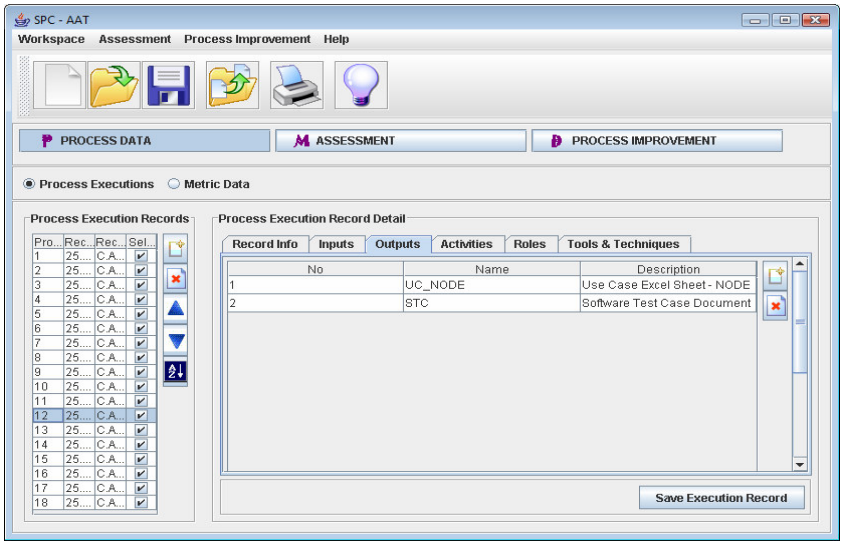


Figure 181 Process Execution Record of Process Execution #12

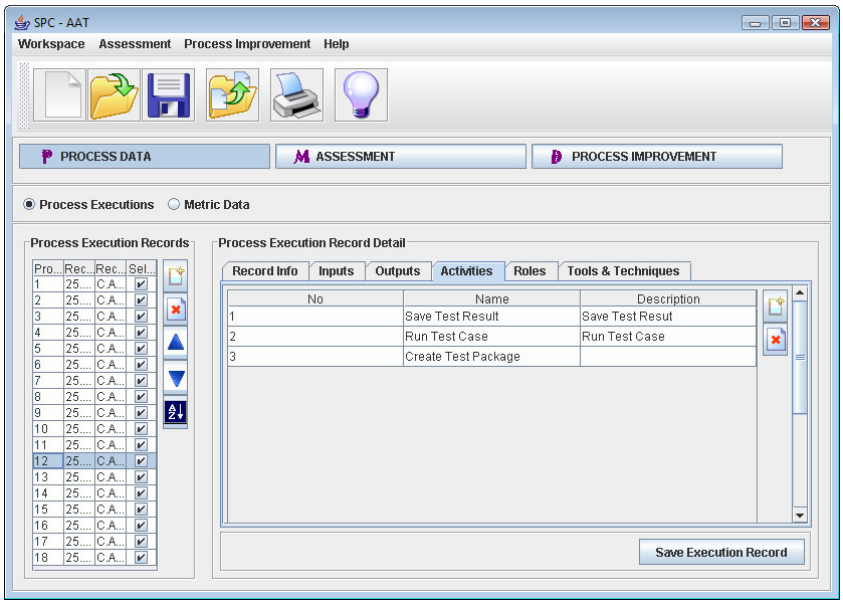


Figure 182 Process Execution Record of Process Execution #12

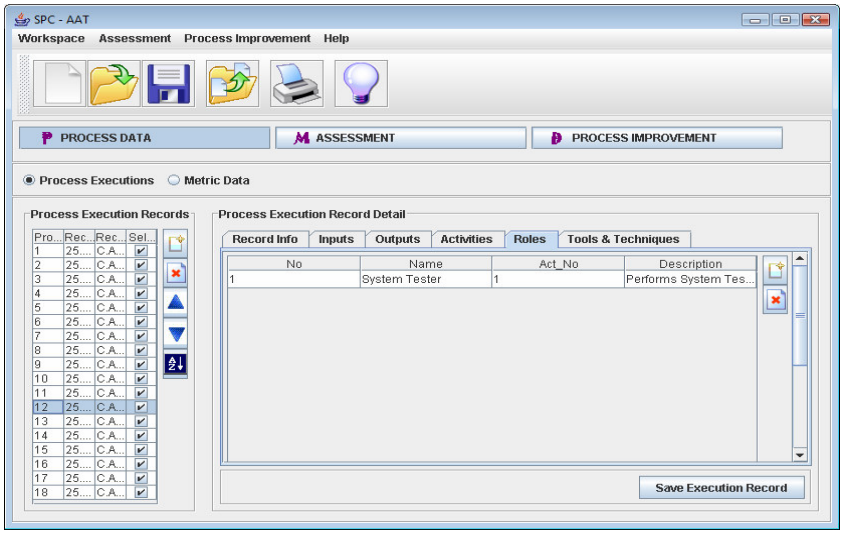


Figure 183 Process Execution Record of Process Execution #12

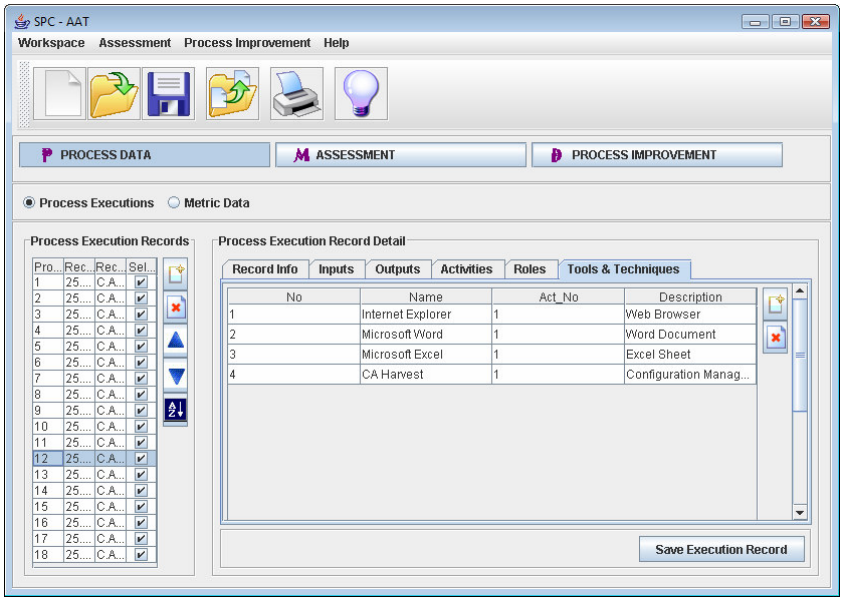


Figure 184 Process Execution Record of Process Execution #12

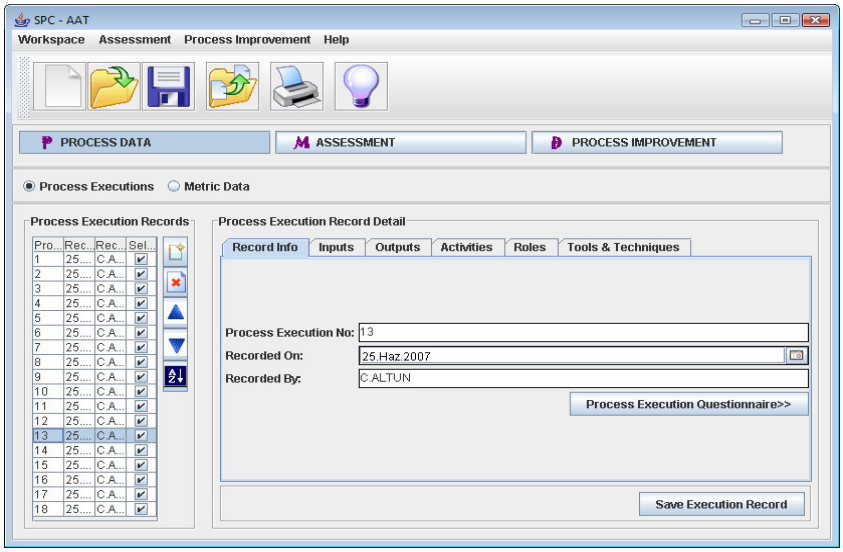


Figure 185 Process Execution Record of Process Execution #12

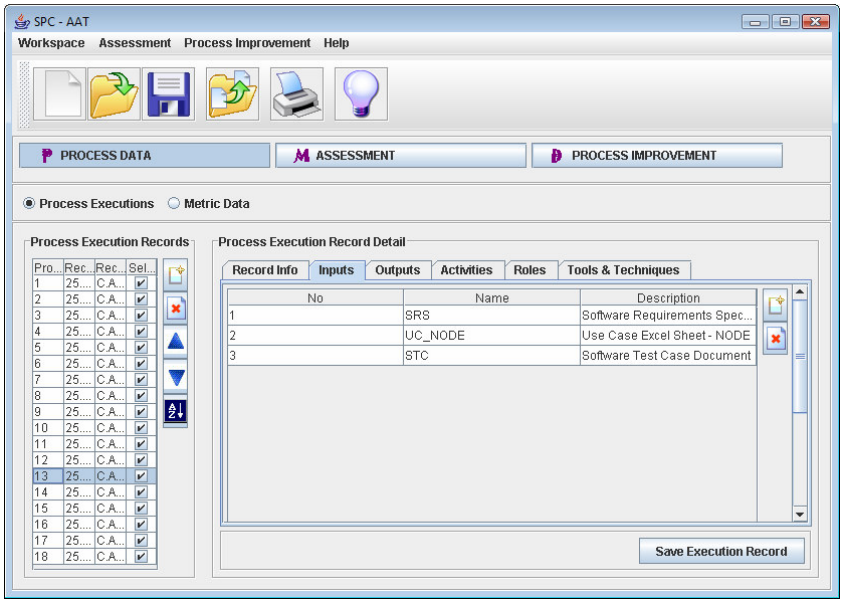


Figure 186 Process Execution Record of Process Execution #13

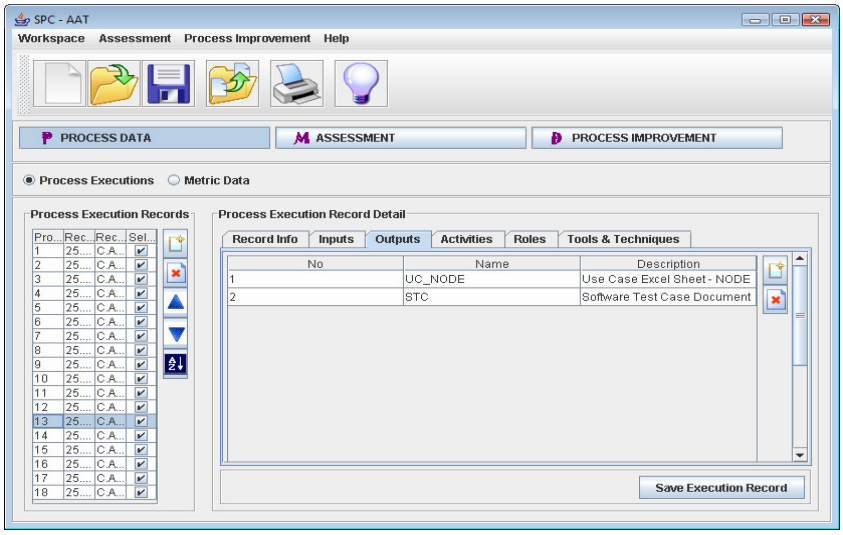


Figure 187 Process Execution Record of Process Execution #13

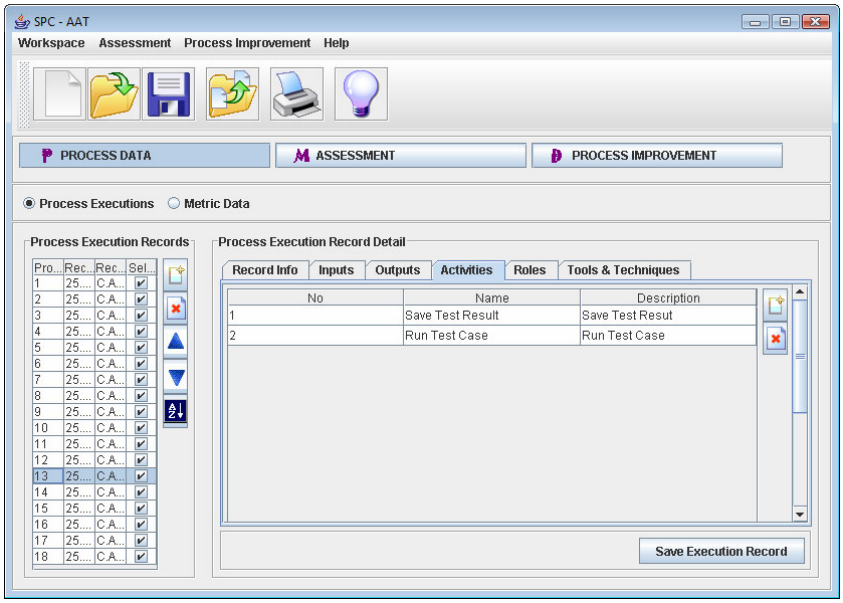


Figure 188 Process Execution Record of Process Execution #13

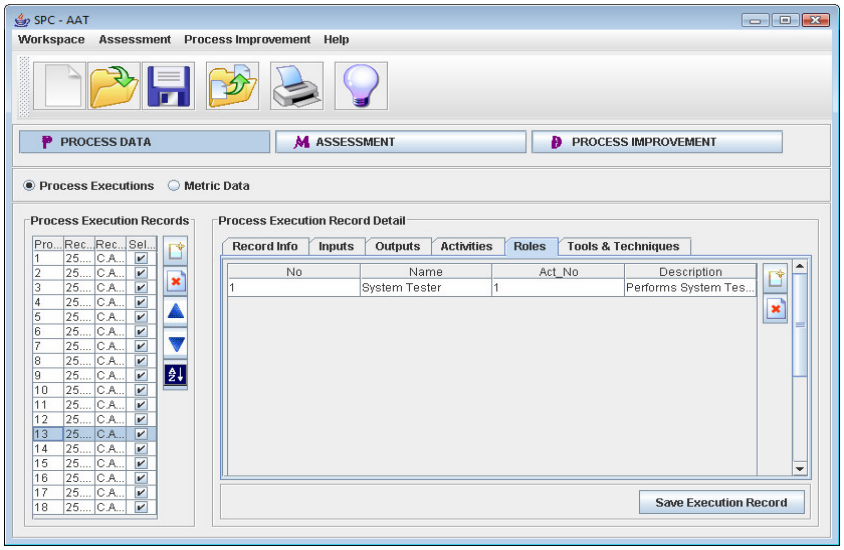


Figure 189 Process Execution Record of Process Execution #13

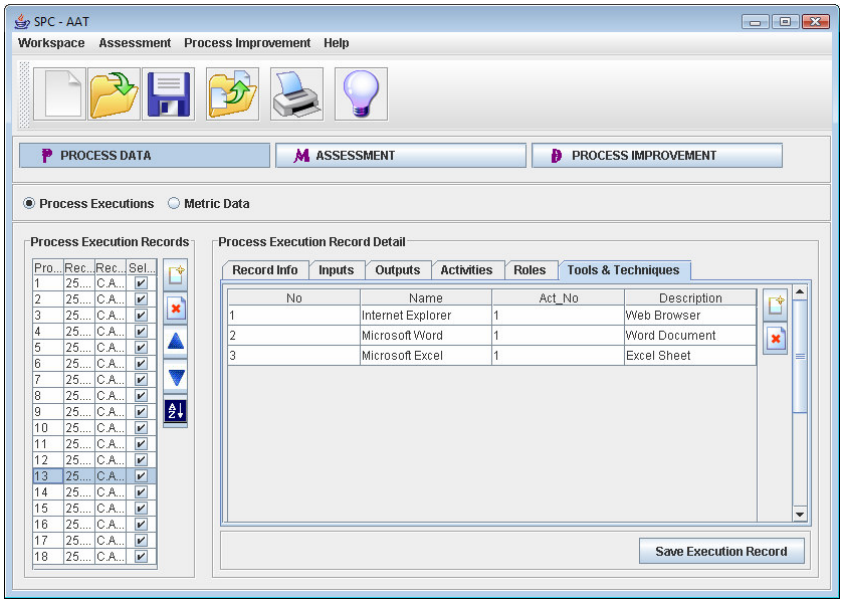


Figure 190 Process Execution Record of Process Execution #13

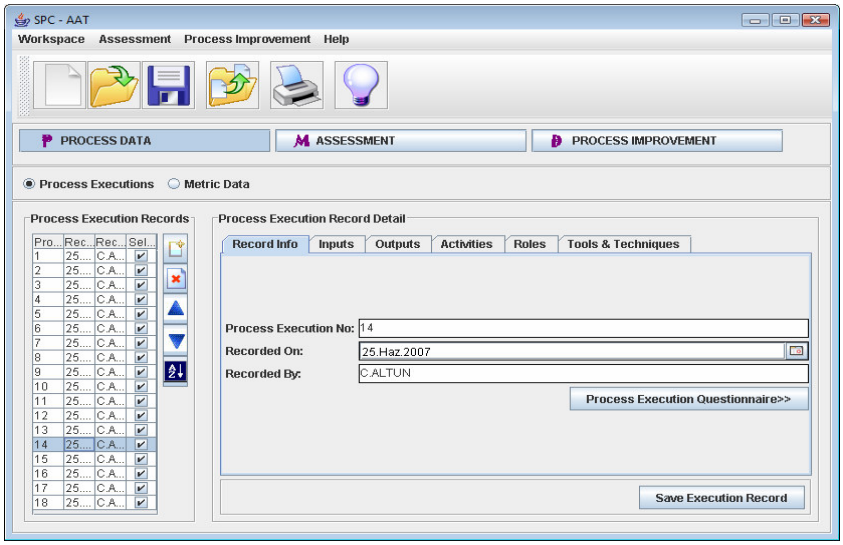


Figure 191 Process Execution Record of Process Execution #14

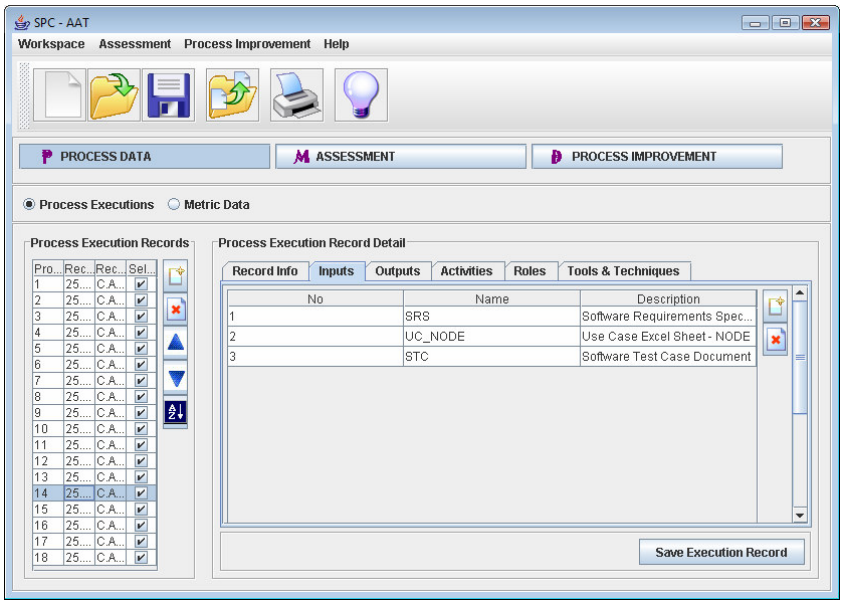


Figure 192 Process Execution Record of Process Execution #14

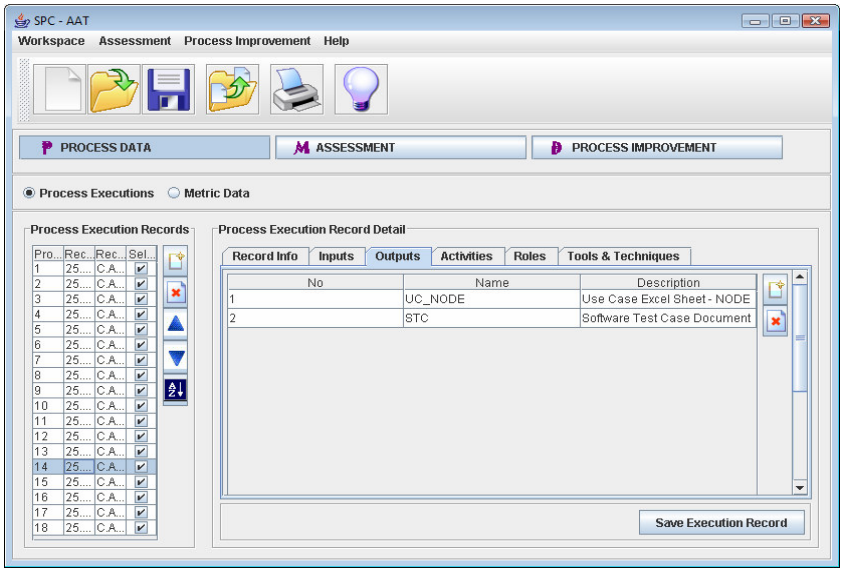


Figure 193 Process Execution Record of Process Execution #14

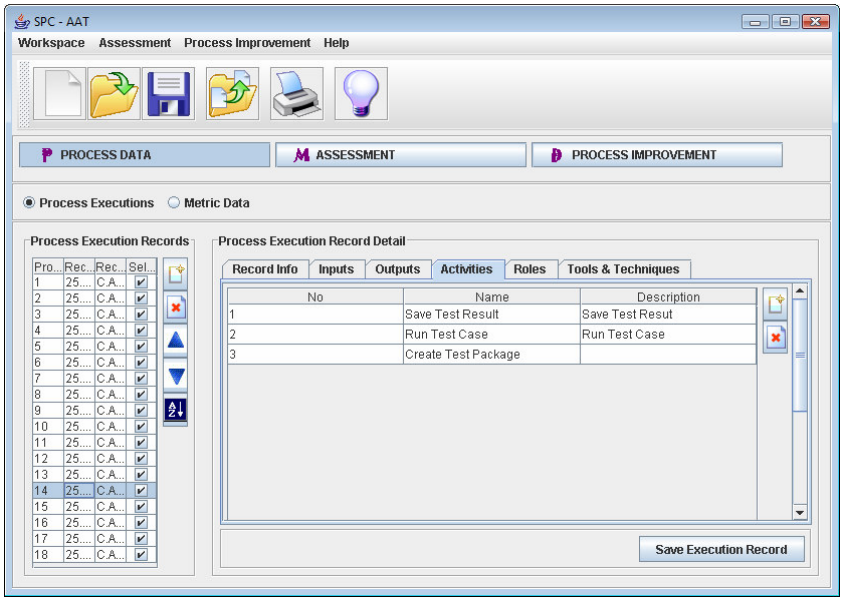


Figure 194 Process Execution Record of Process Execution #14

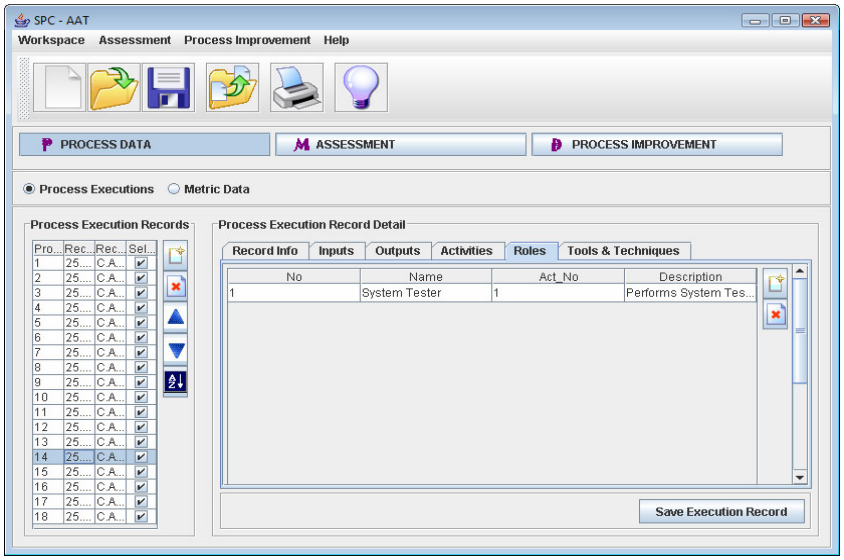


Figure 195 Process Execution Record of Process Execution #14

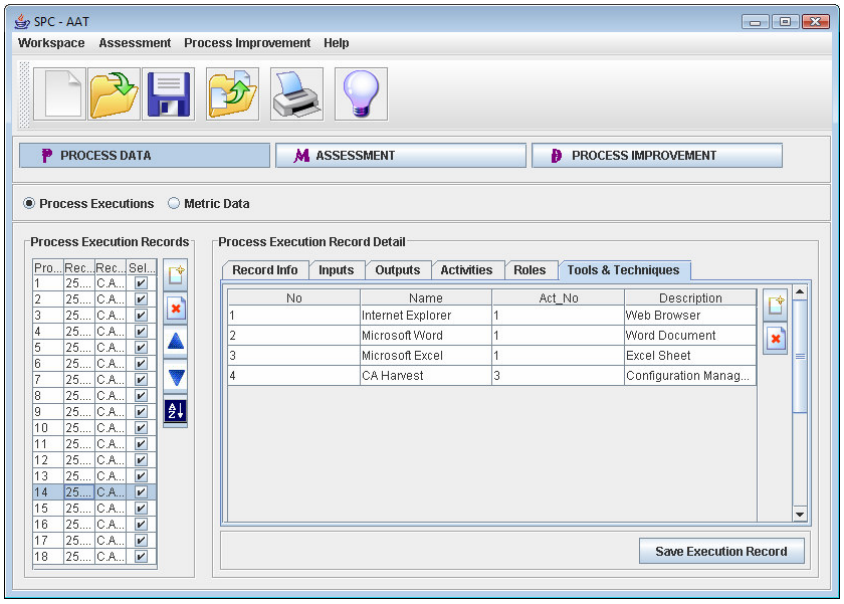


Figure 196 Process Execution Record of Process Execution #14

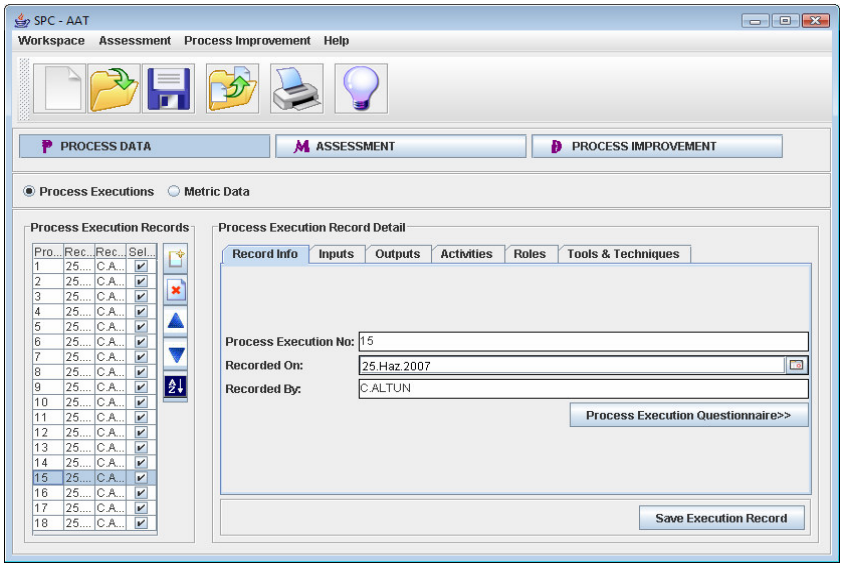


Figure 197 Process Execution Record of Process Execution #15

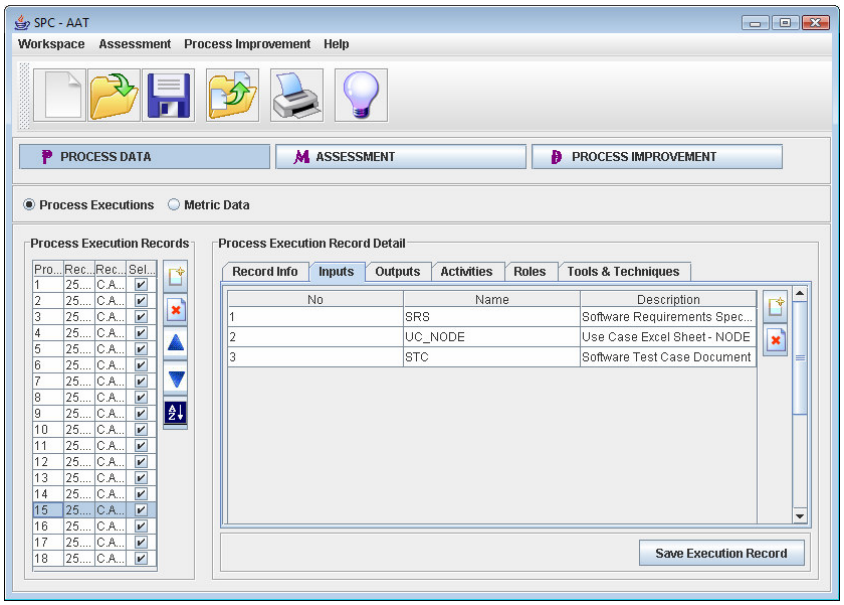


Figure 198 Process Execution Record of Process Execution #15

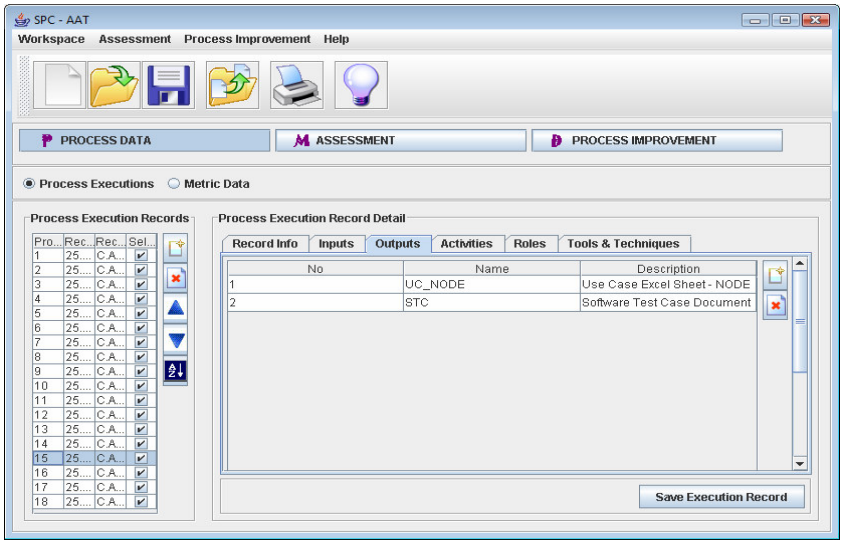


Figure 199 Process Execution Record of Process Execution #15

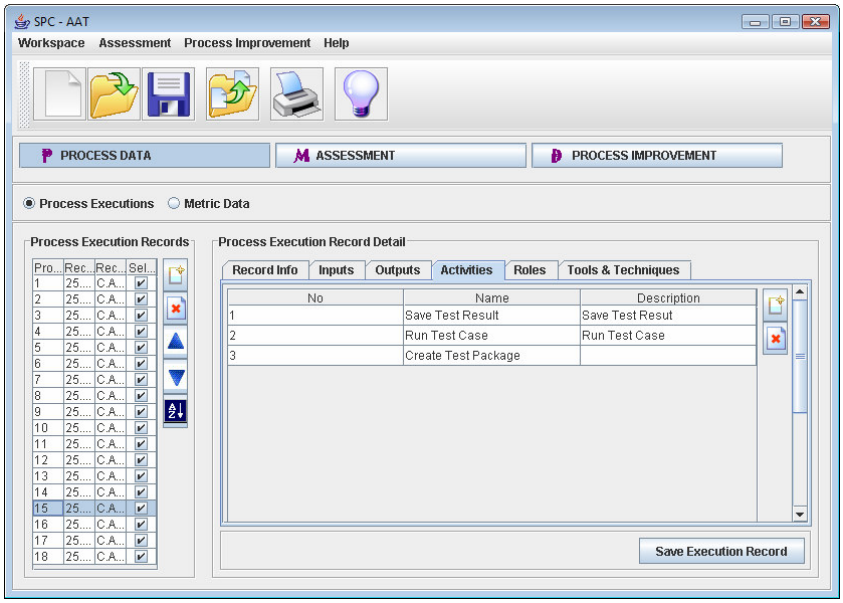


Figure 200 Process Execution Record of Process Execution #15

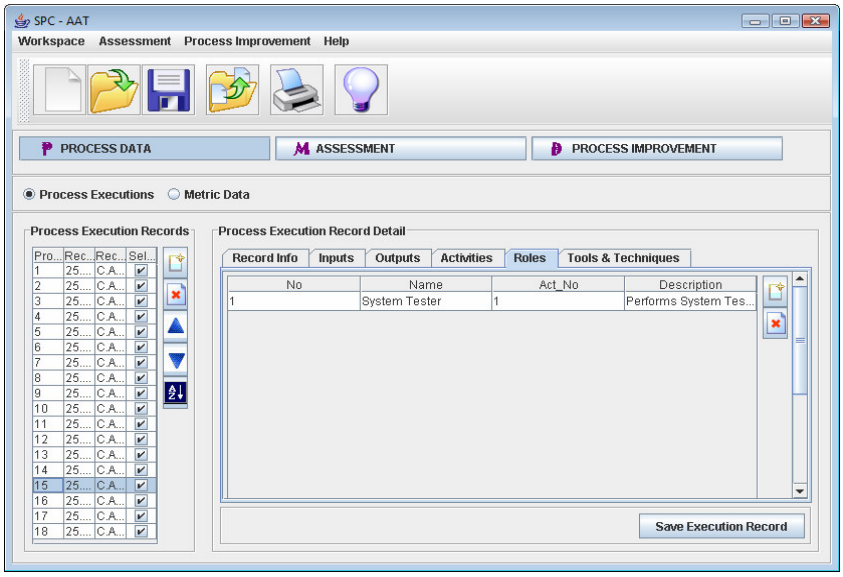


Figure 201 Process Execution Record of Process Execution #15

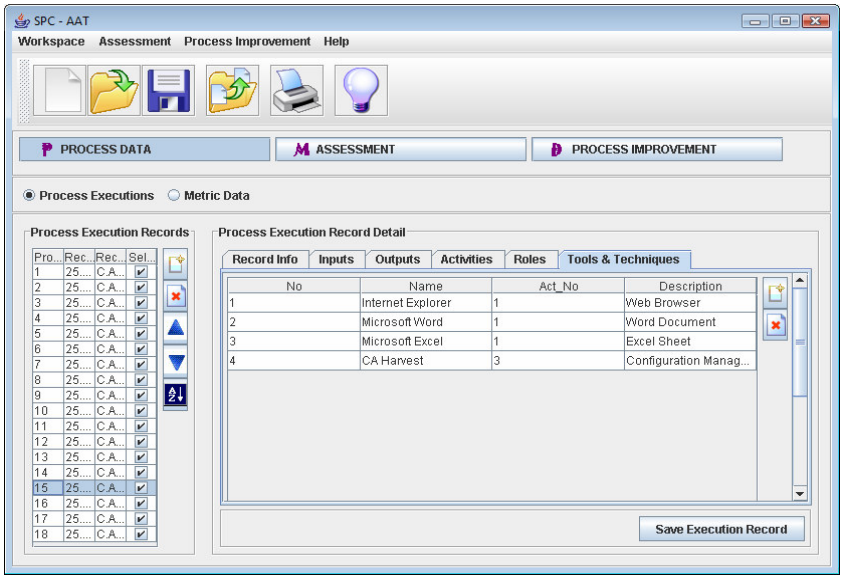


Figure 202 Process Execution Record of Process Execution #15

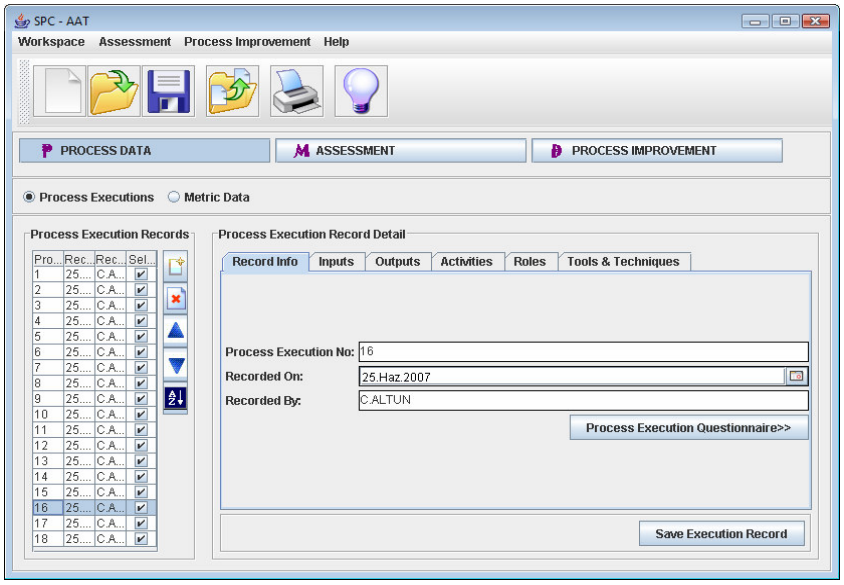


Figure 203 Process Execution Record of Process Execution #16

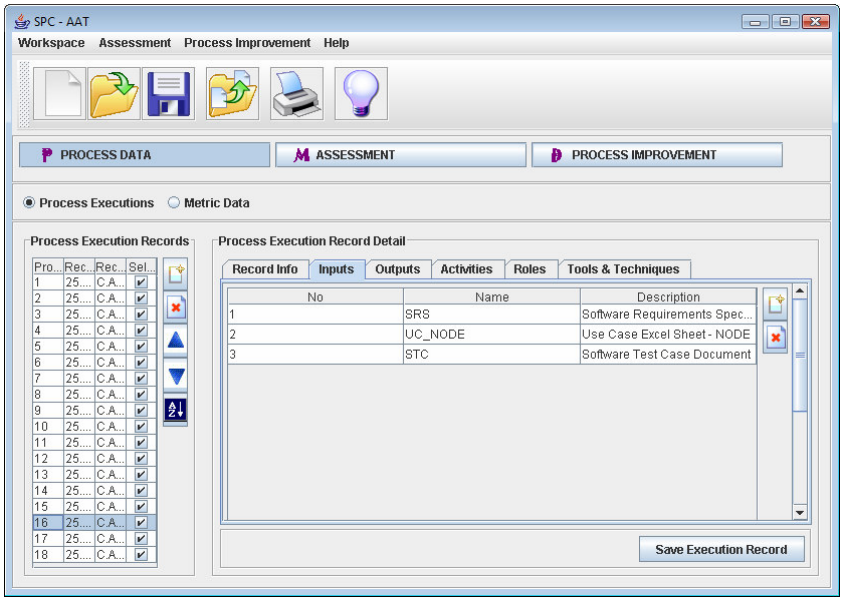


Figure 204 Process Execution Record of Process Execution #16

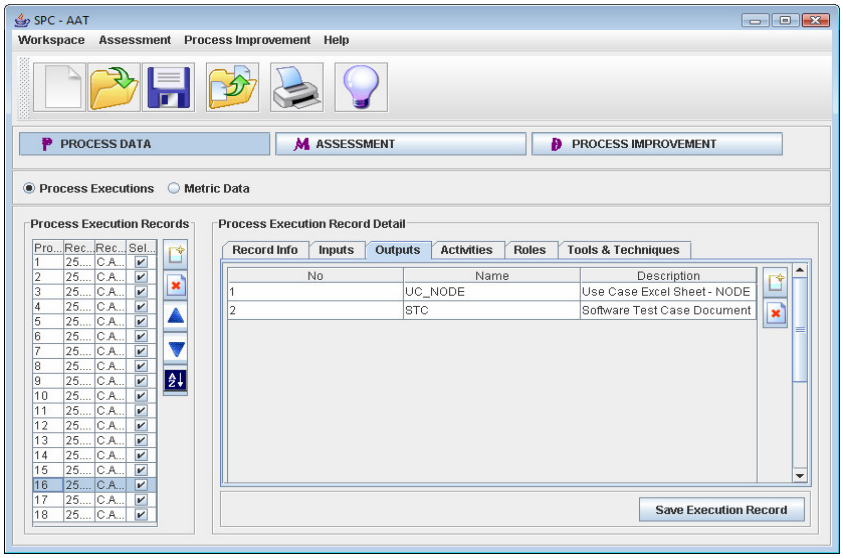


Figure 205 Process Execution Record of Process Execution #16

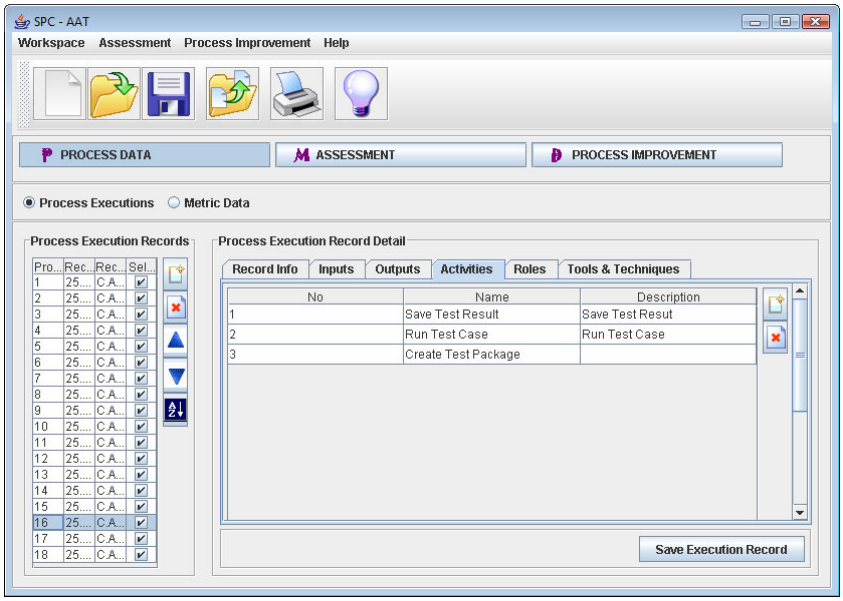


Figure 206 Process Execution Record of Process Execution #16

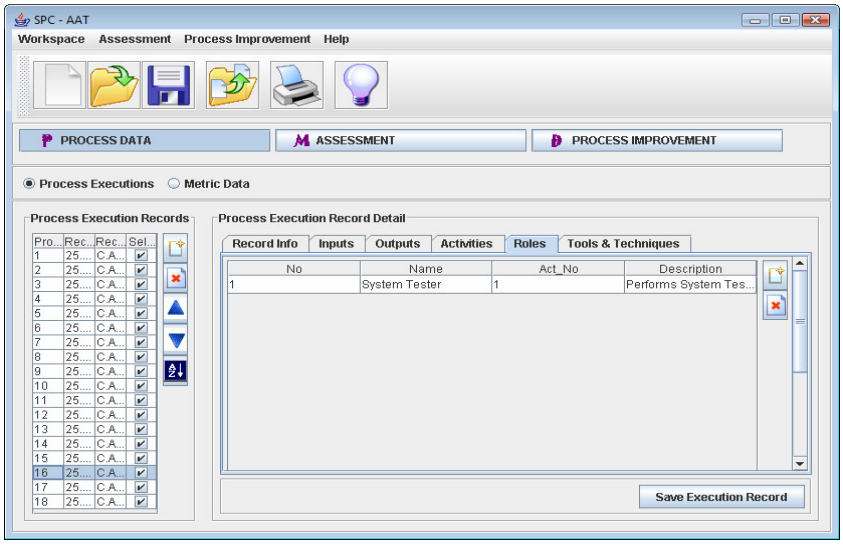


Figure 207 Process Execution Record of Process Execution #16

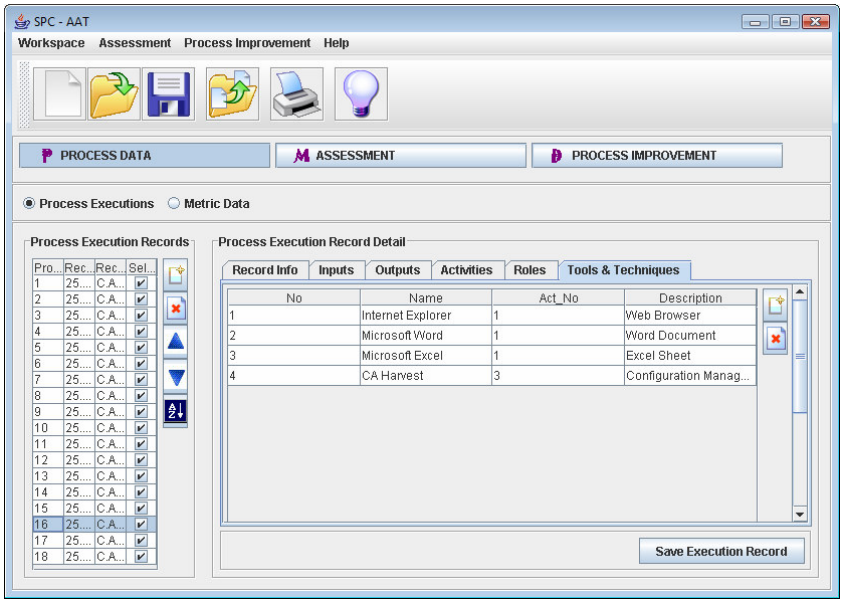


Figure 208 Process Execution Record of Process Execution #16

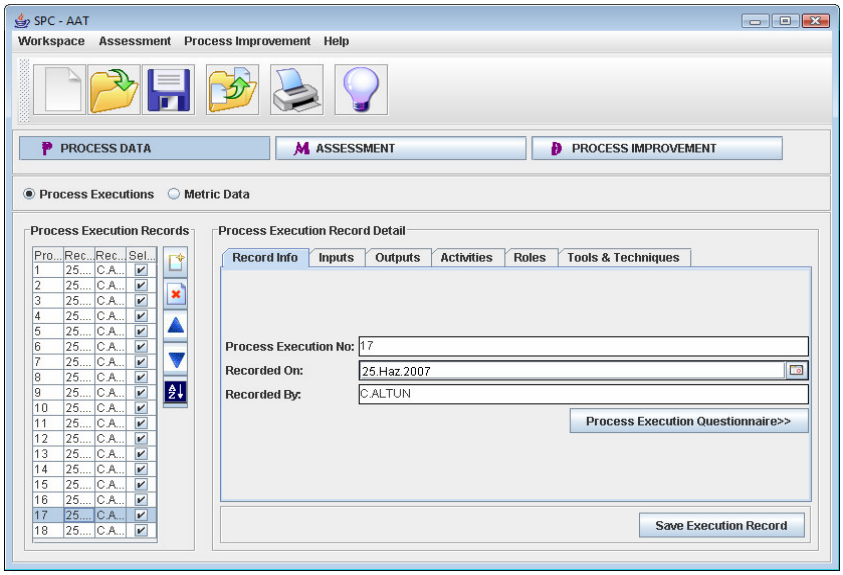


Figure 209 Process Execution Record of Process Execution #17

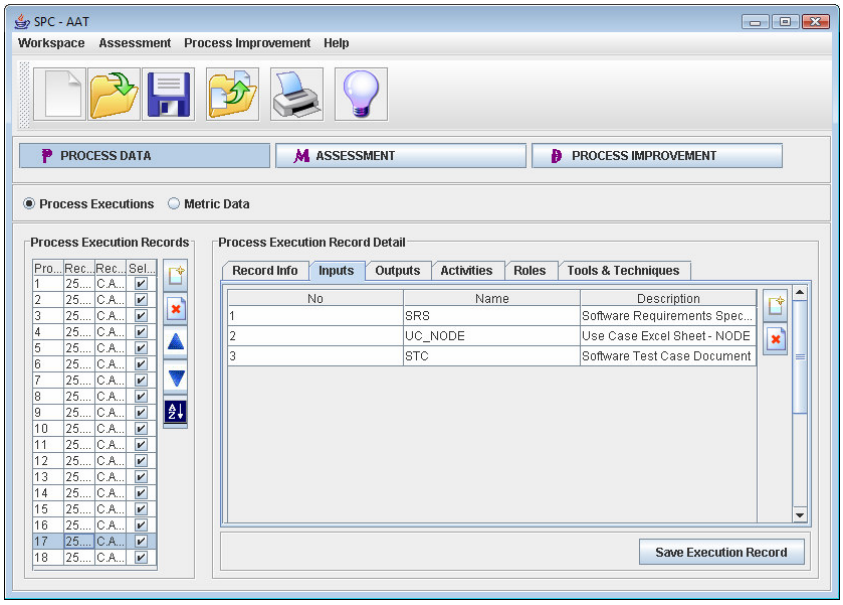


Figure 210 Process Execution Record of Process Execution #17

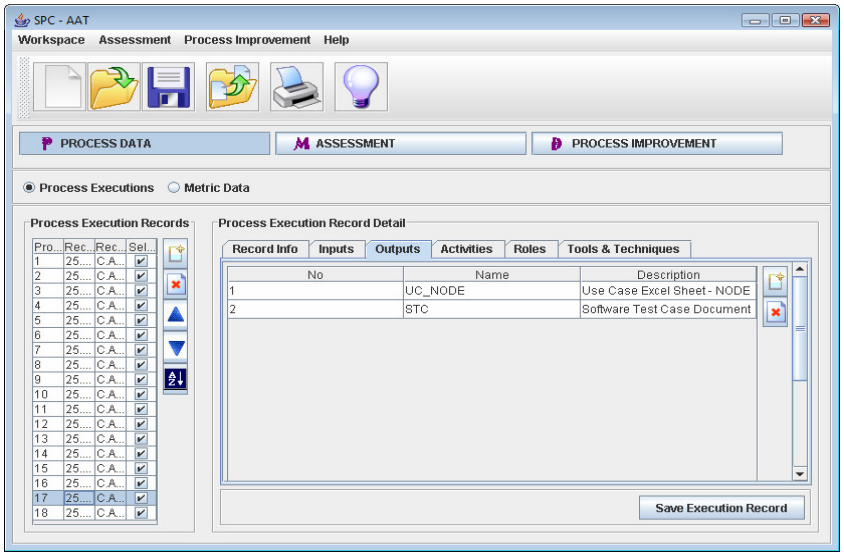


Figure 211 Process Execution Record of Process Execution #17

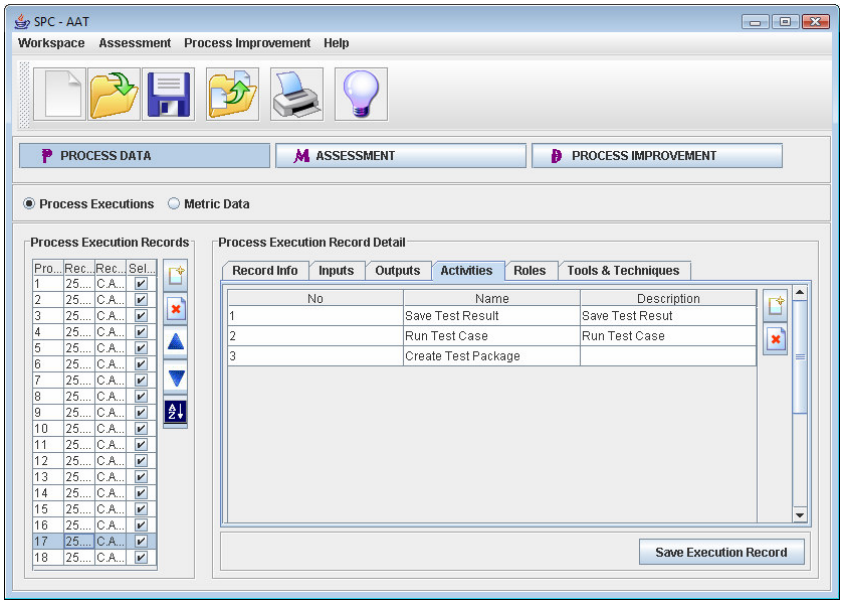


Figure 212 Process Execution Record of Process Execution #17

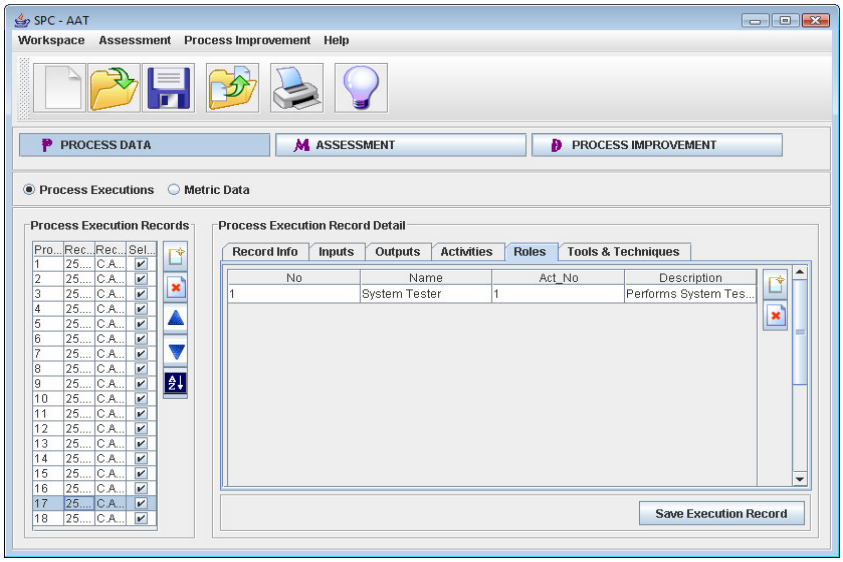


Figure 213 Process Execution Record of Process Execution #17

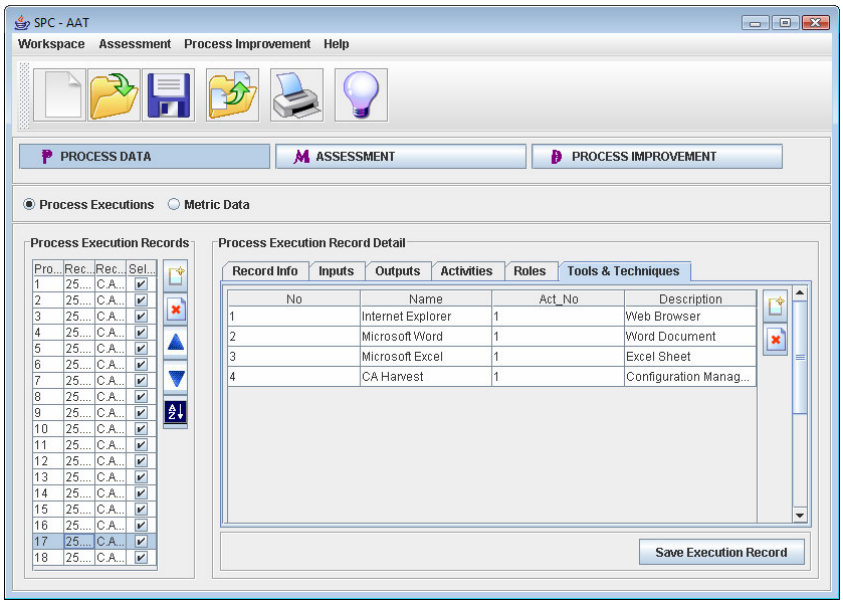


Figure 214 Process Execution Record of Process Execution #17

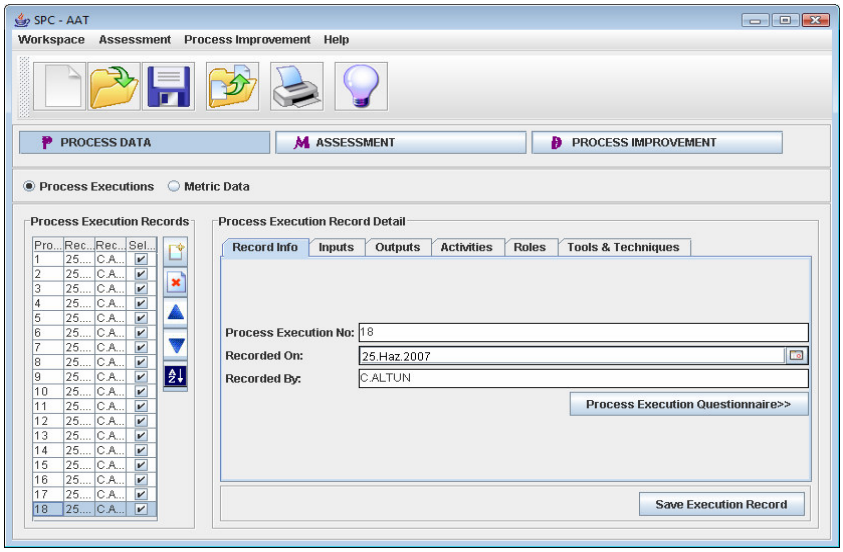


Figure 215 Process Execution Record of Process Execution #17

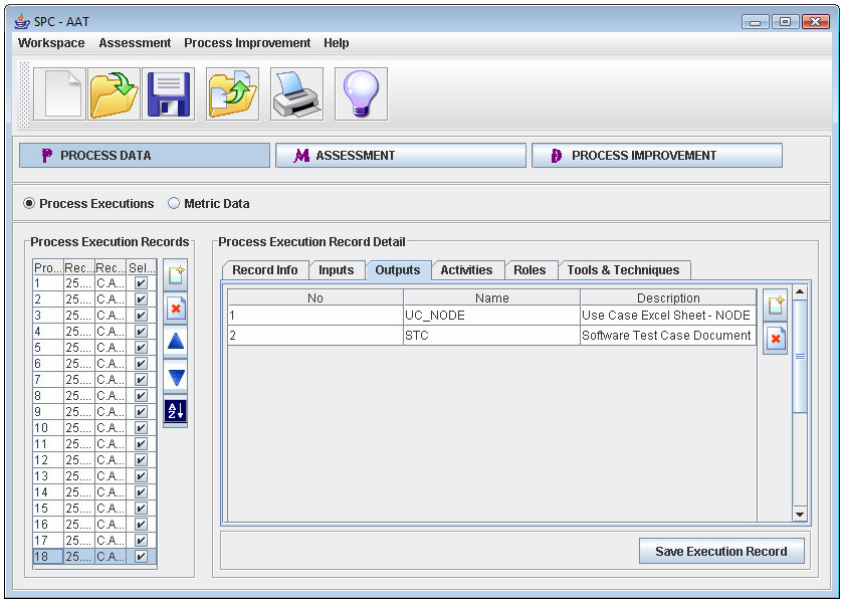


Figure 216 Process Execution Record of Process Execution #18

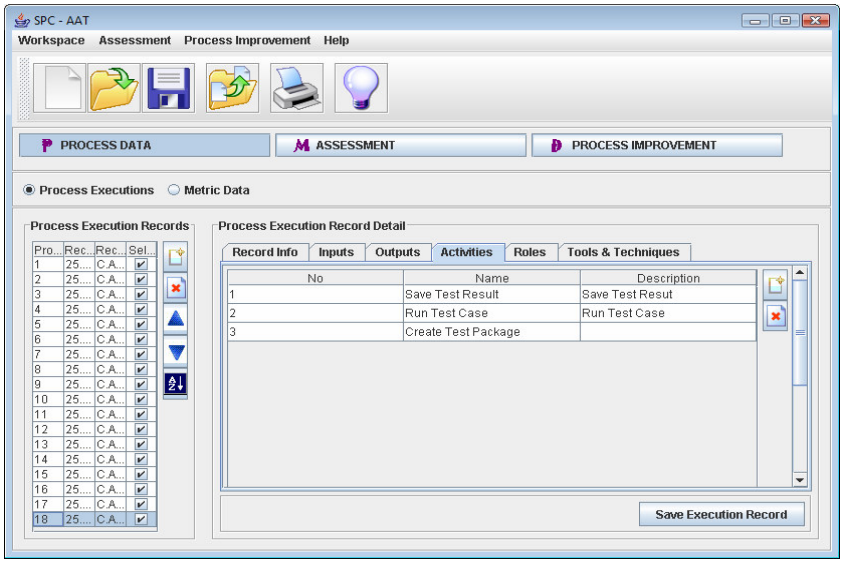


Figure 217 Process Execution Record of Process Execution #18

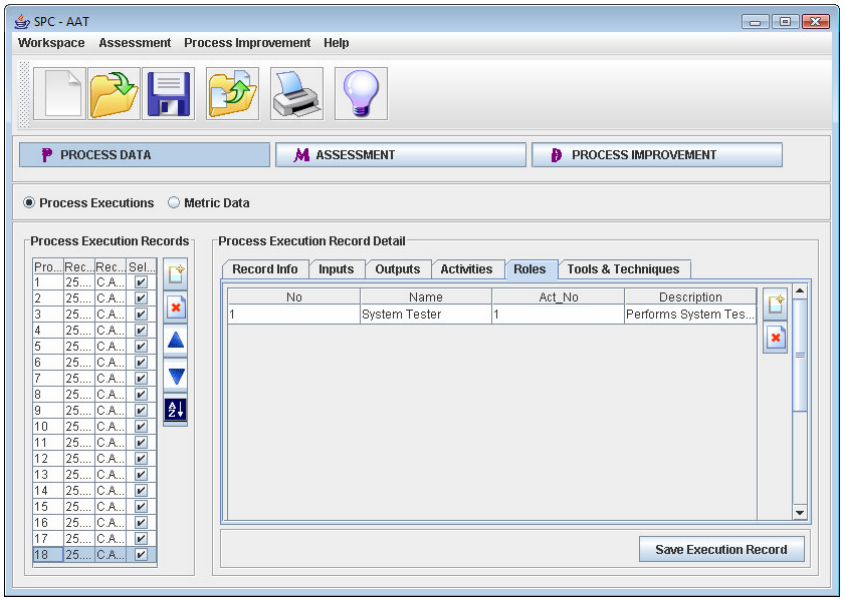


Figure 218 Process Execution Record of Process Execution #18

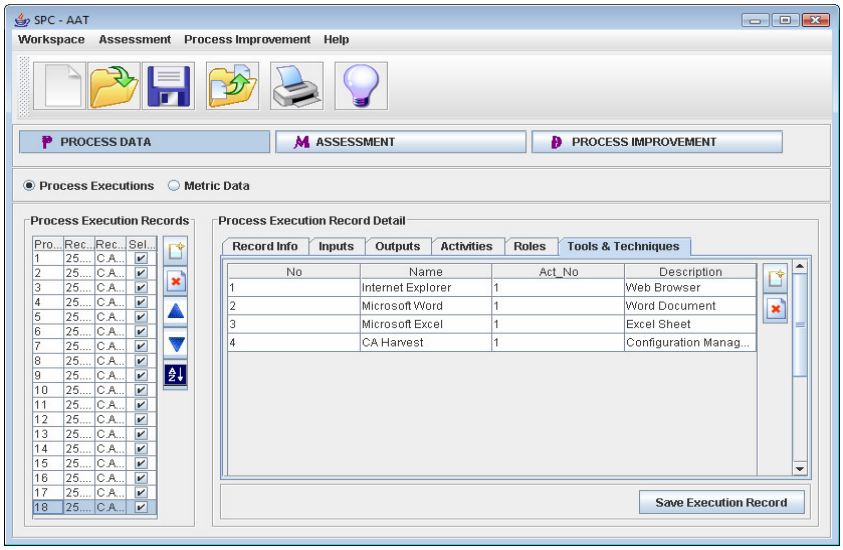


Figure 219 Process Execution Record of Process Execution #18

C. METRIC USABILITY QUESTIONNAIRES OF CASE STUDIES A, B

SPC-AM Assets

Metric Usability Questionnaires of Case Study A

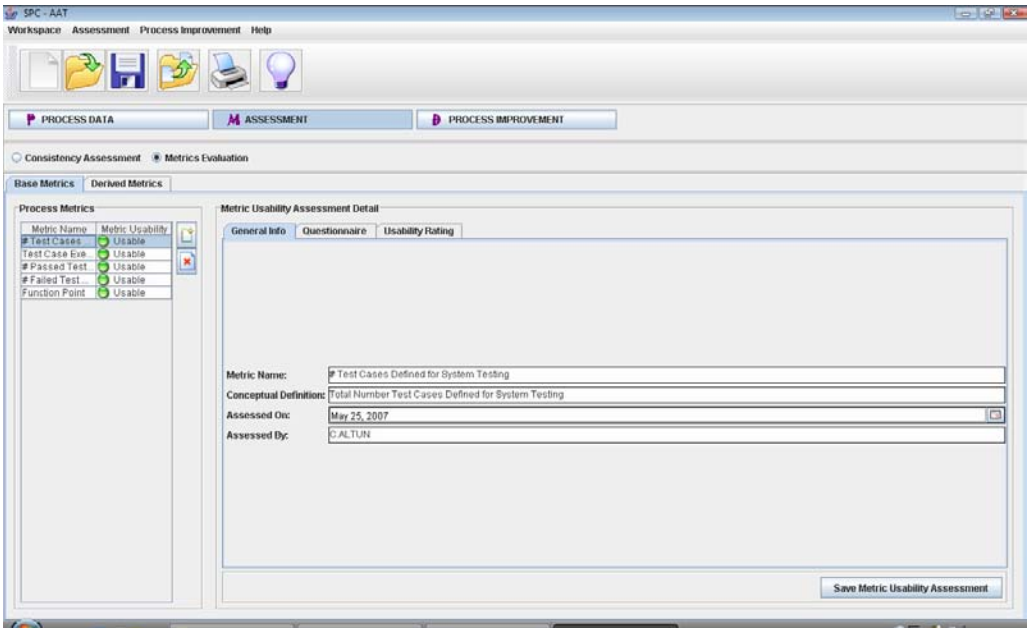


Figure 220 Metric Usability Questionnaire of “# Test Cases” Base Metric

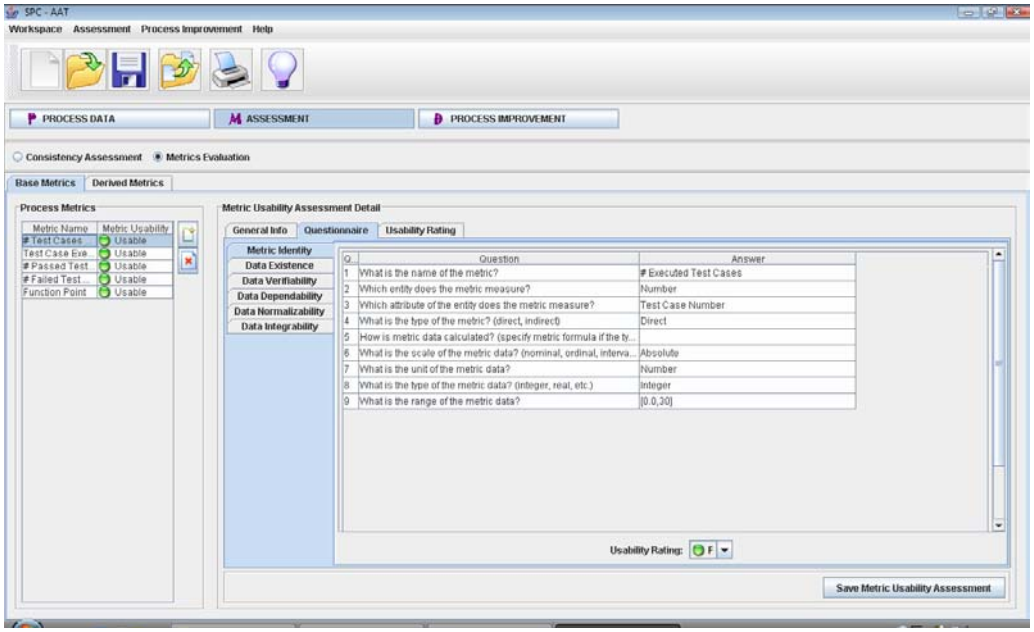


Figure 221 Metric Usability Questionnaire of “# Test Cases” Base Metric

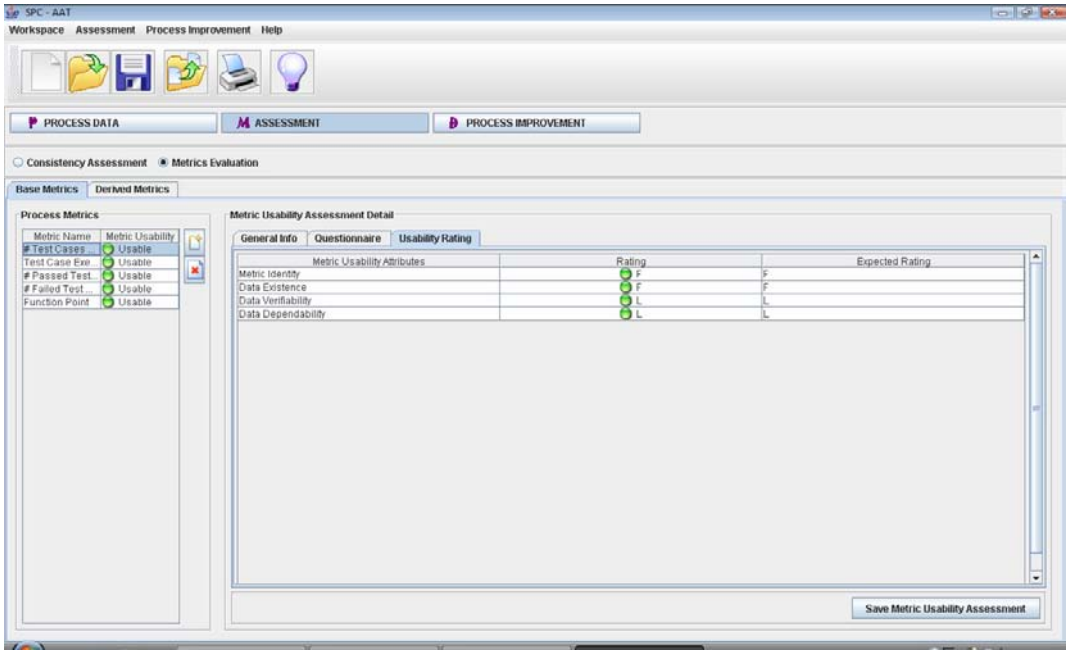


Figure 222 Metric Usability Questionnaire of “# Test Cases” Base Metric

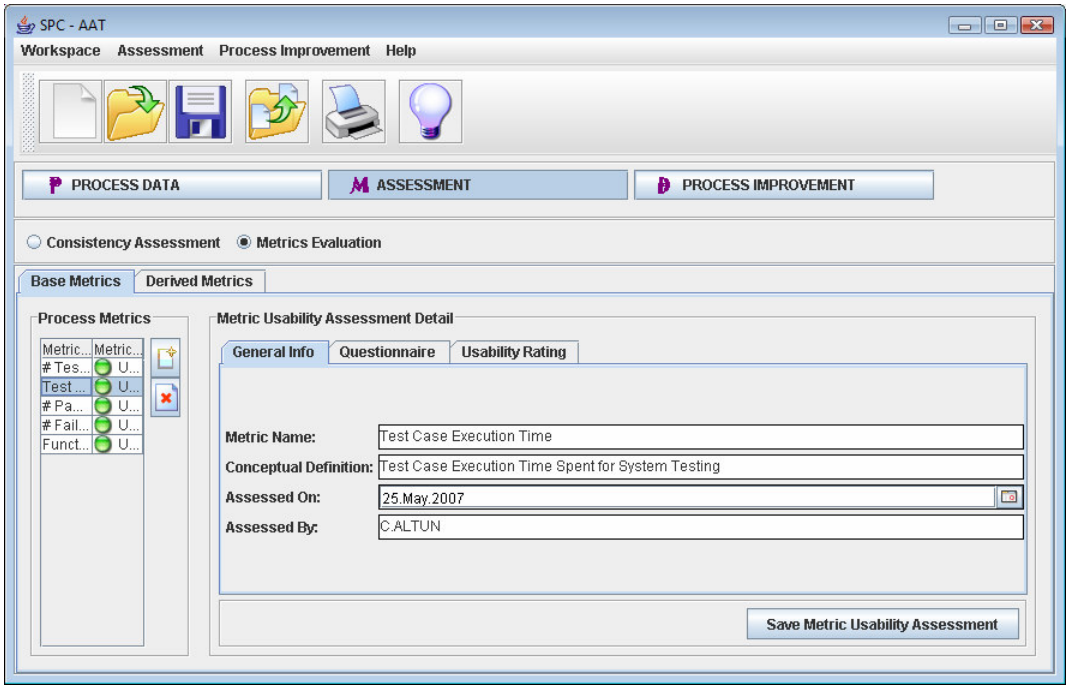


Figure 223 Metric Usability Questionnaire of “Test Case Execution Time” Base Metric

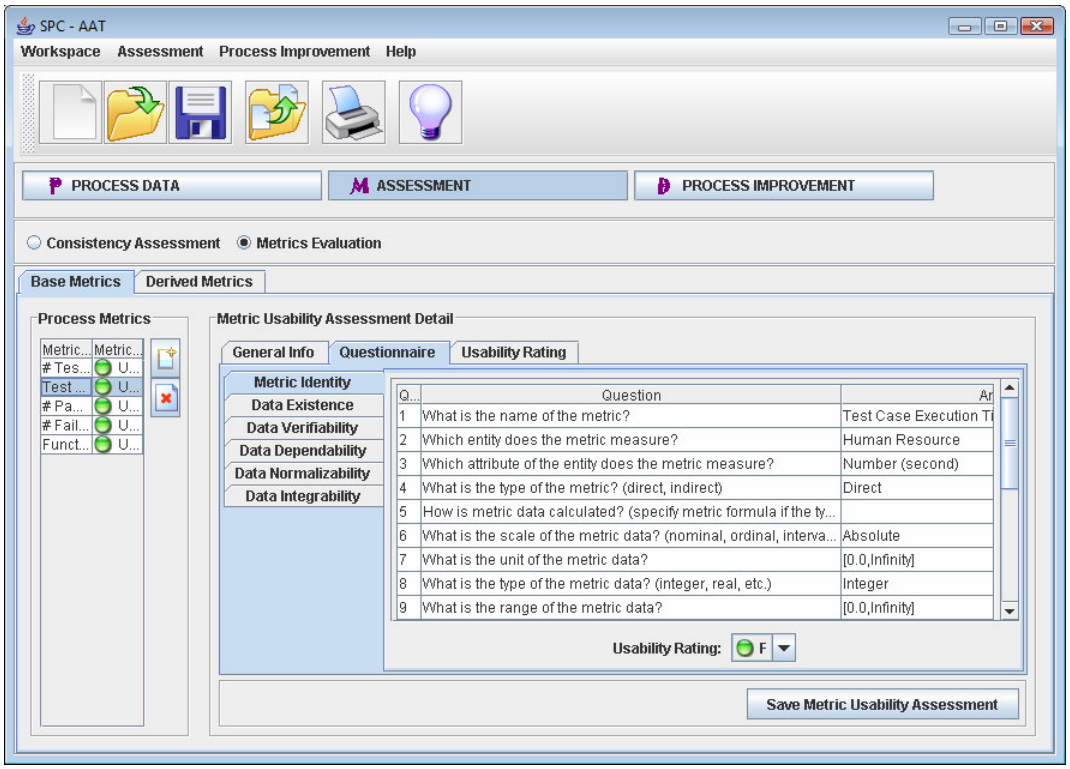


Figure 224 Metric Usability Questionnaire of “Test Case Execution Time” Base Metric

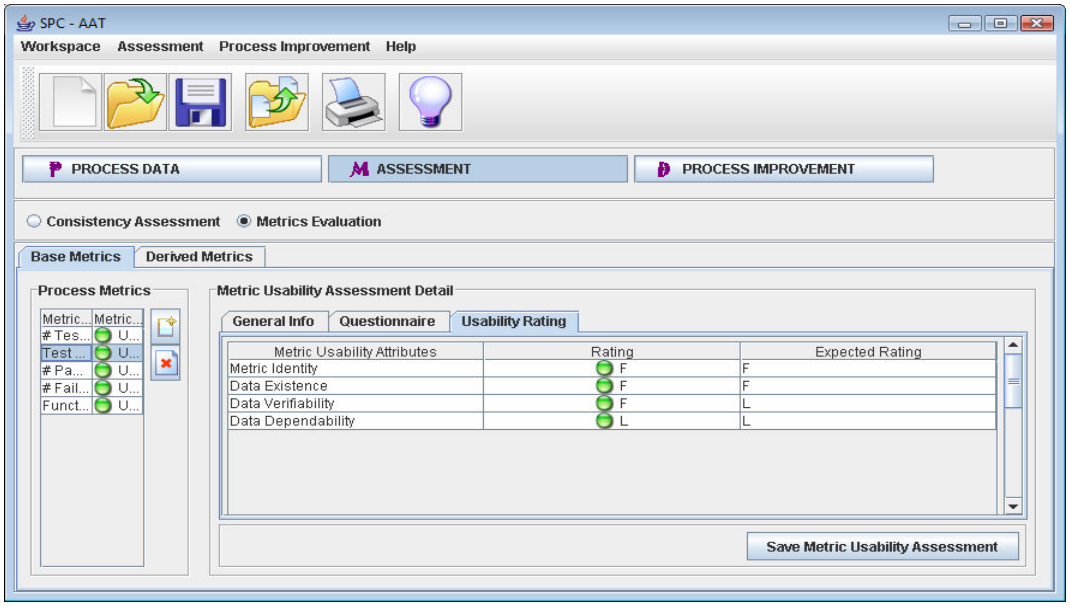


Figure 225 Metric Usability Questionnaire of “Test Case Execution Time” Base Metric

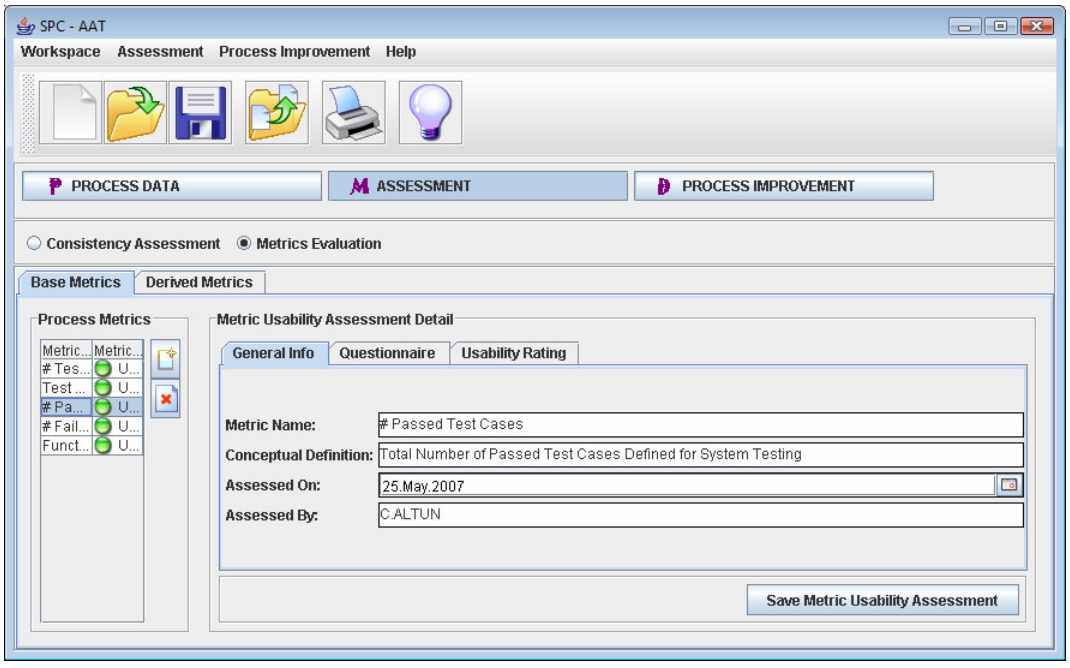


Figure 226 Metric Usability Questionnaire of “# Passed Test Cases” Base Metric

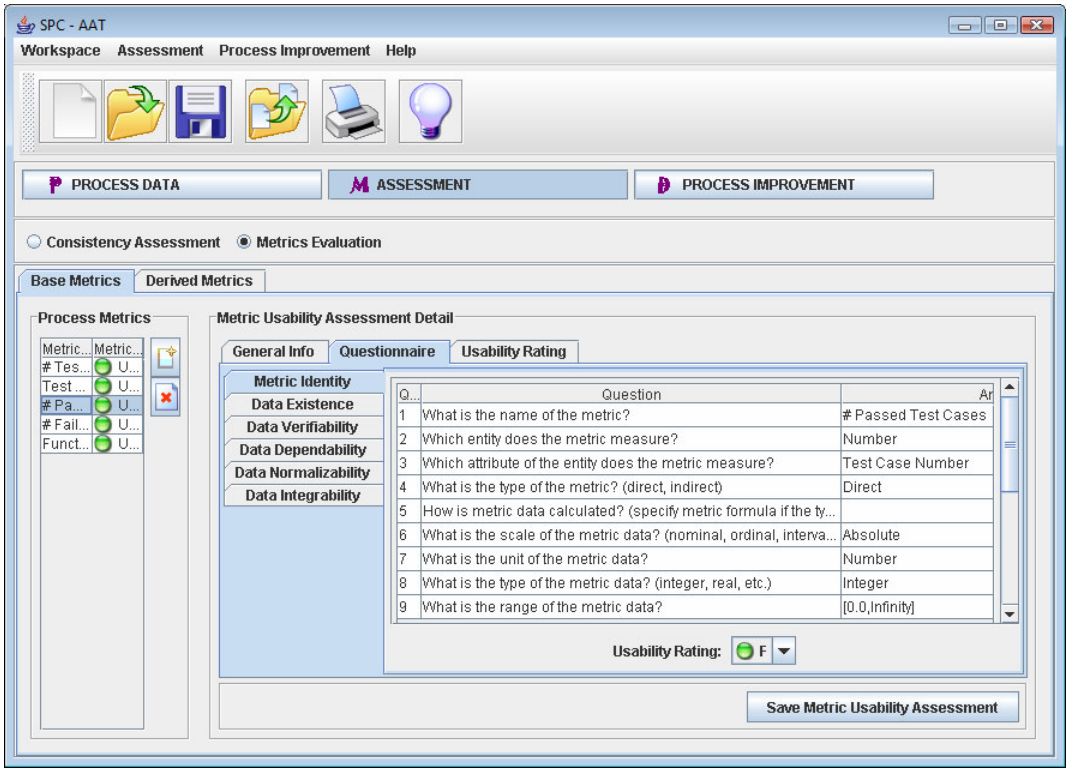


Figure 227 Metric Usability Questionnaire of “# Passed Test Cases” Base Metric

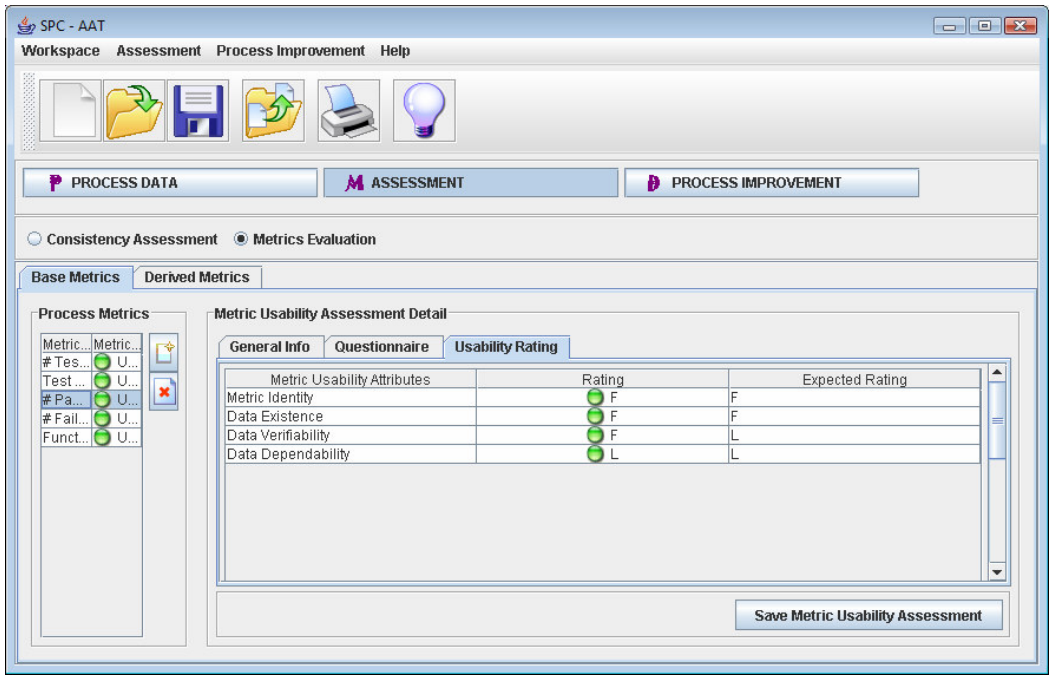


Figure 228 Metric Usability Questionnaire of “# Passed Test Cases” Base Metric

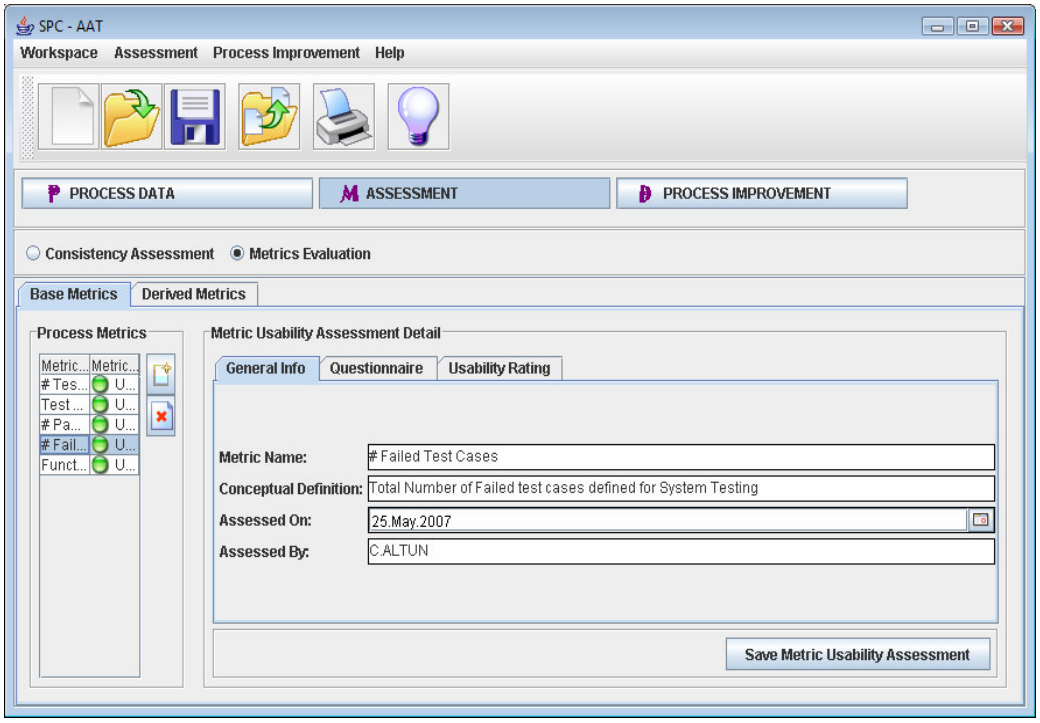


Figure 229 Metric Usability Questionnaire of “# Failed Test Cases” Base Metric

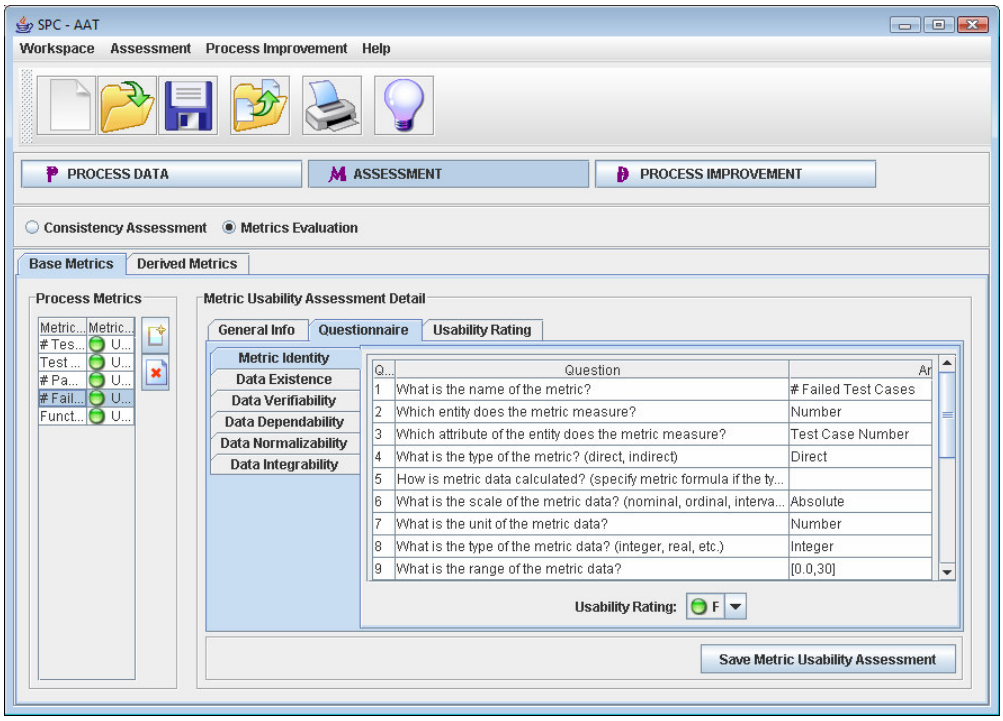


Figure 230 Metric Usability Questionnaire of “# Failed Test Cases” Base Metric

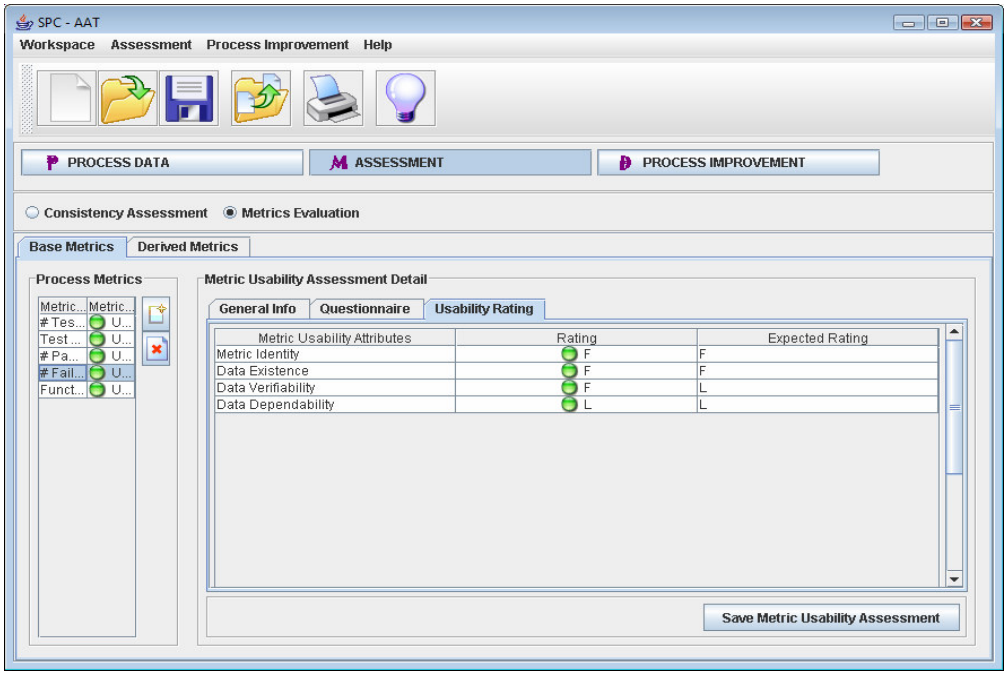


Figure 231 Metric Usability Questionnaire of “# Failed Test Cases” Base Metric

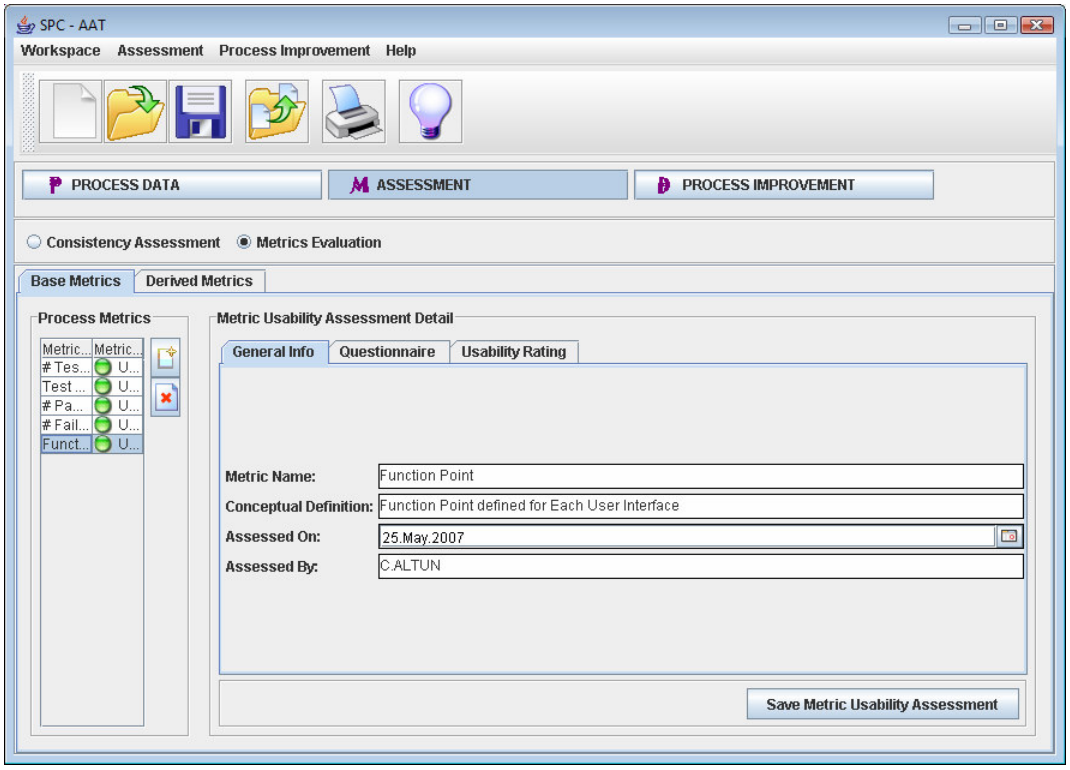


Figure 232 Metric Usability Questionnaire of “Function Point” Base Metric

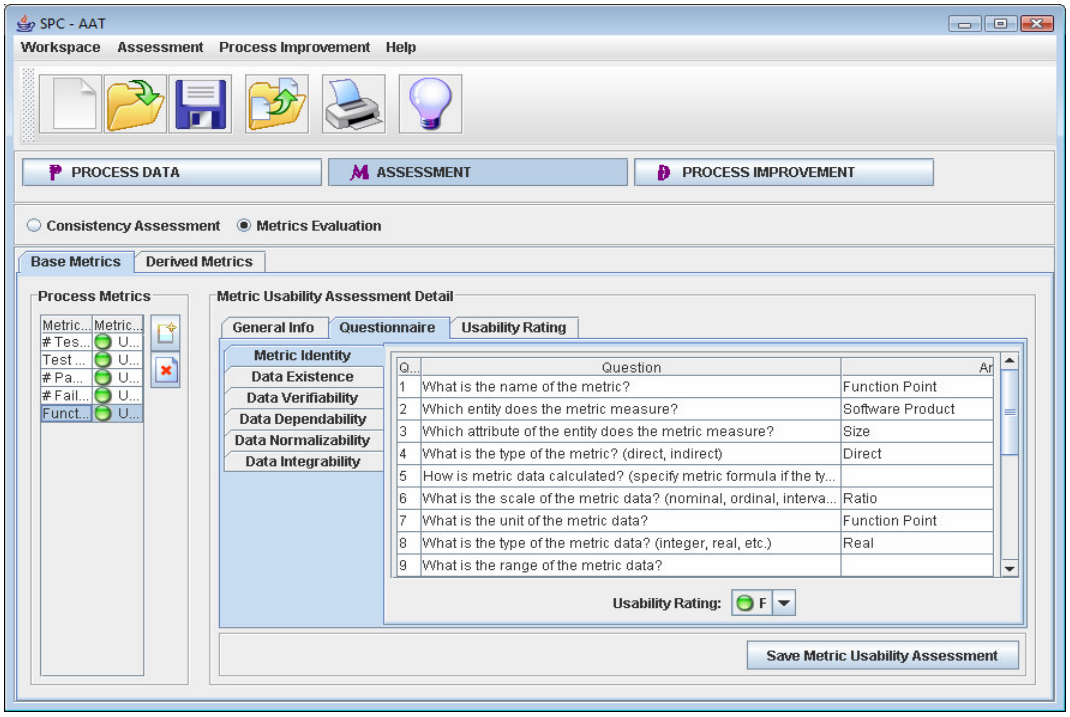


Figure 233 Metric Usability Questionnaire of “Function Point” Base Metric

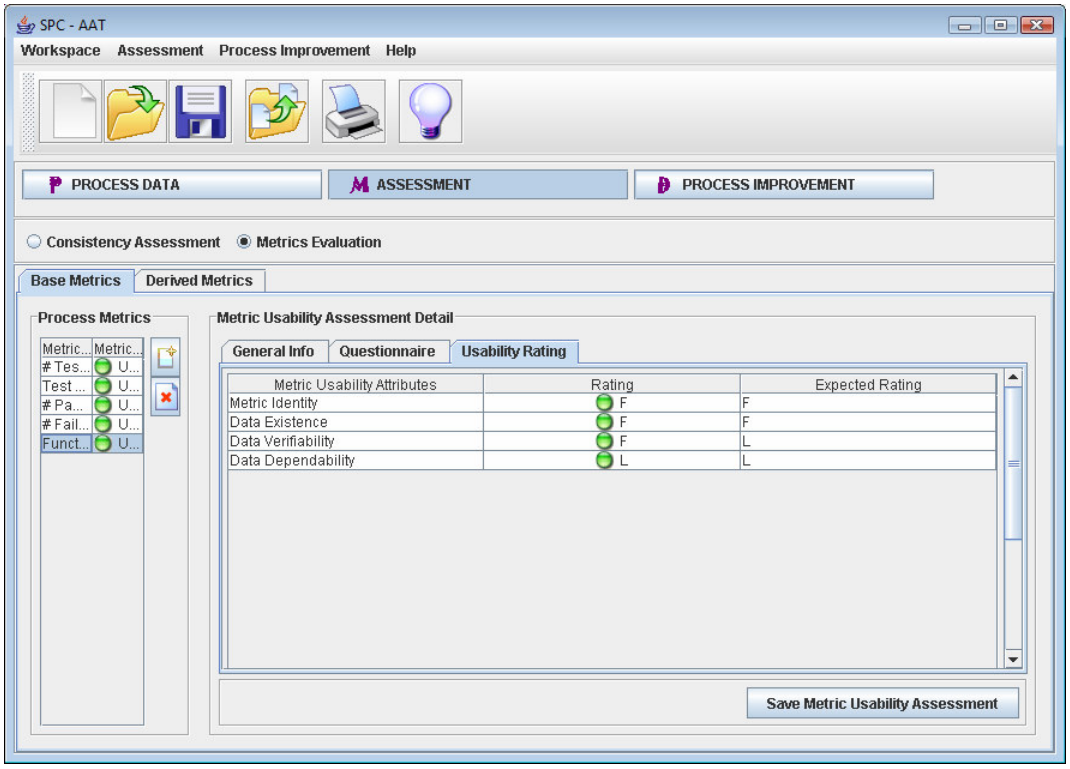


Figure 234 Metric Usability Questionnaire of “Function Point” Base Metric

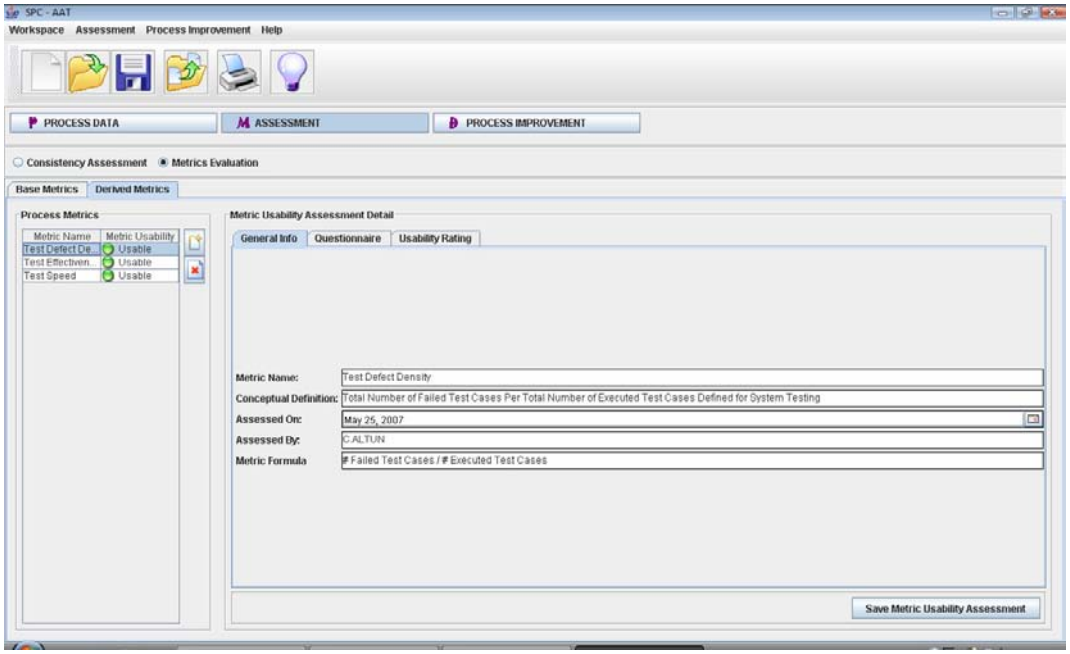


Figure 235 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

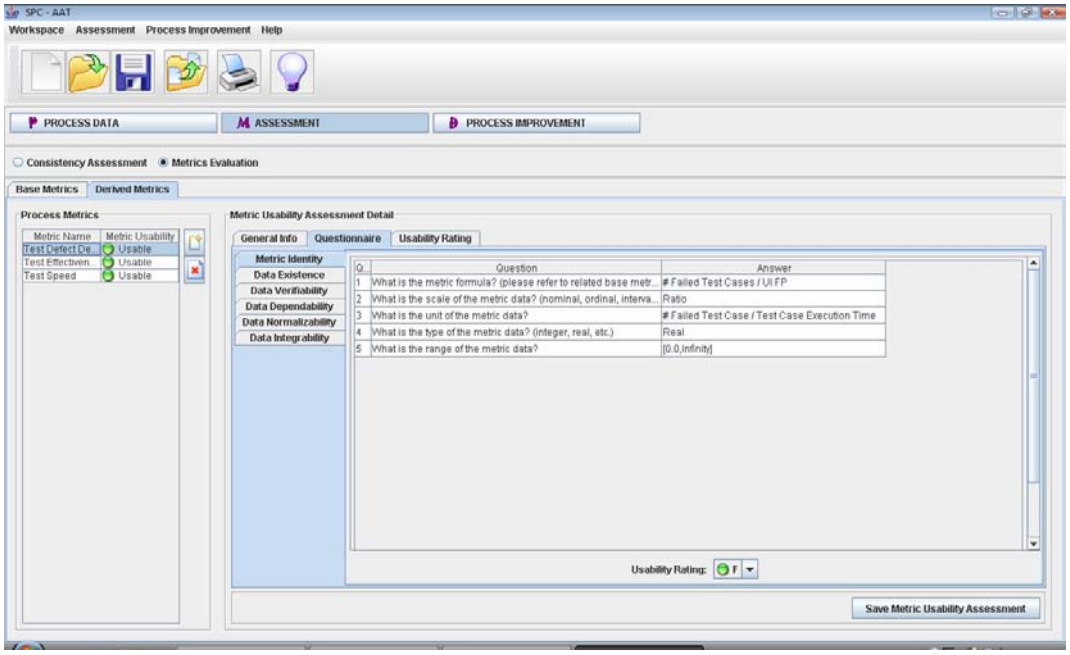


Figure 236 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

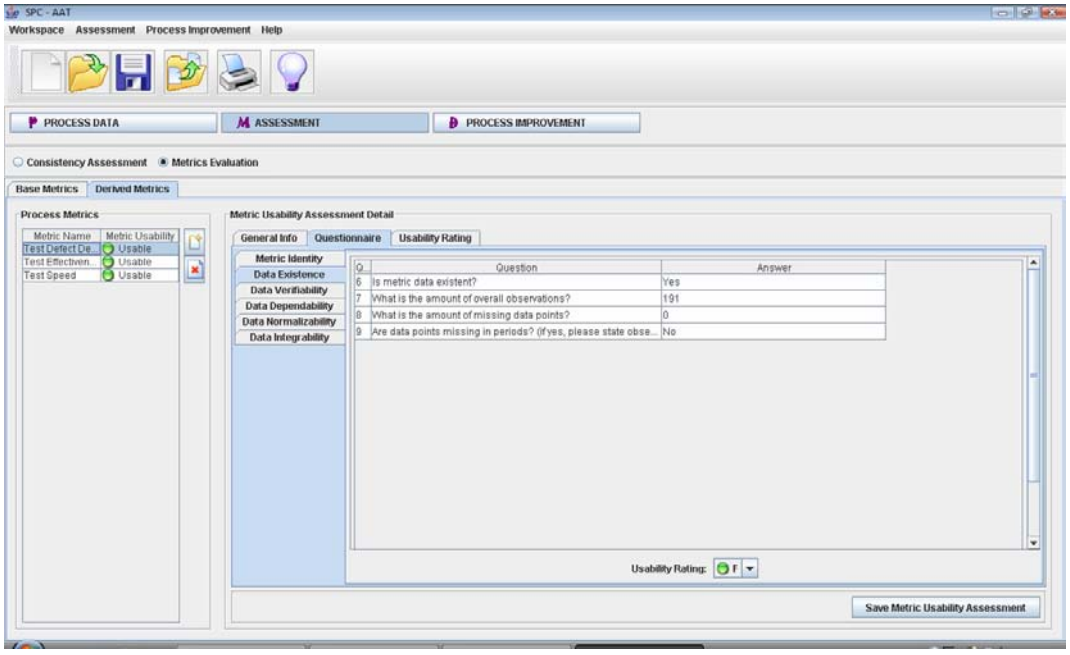


Figure 237 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

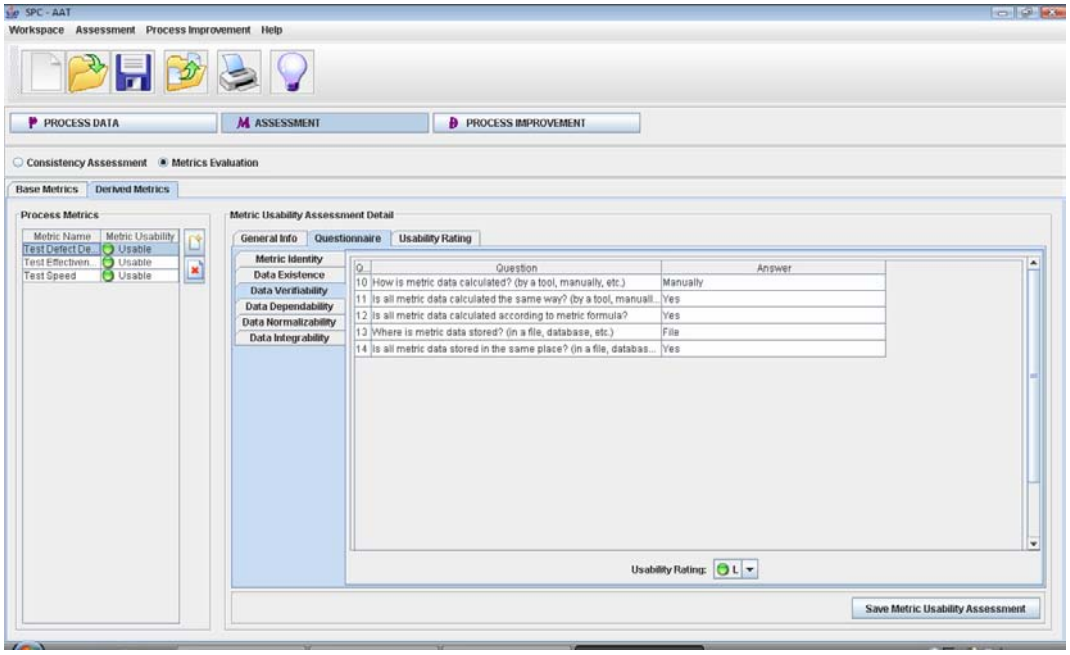


Figure 238 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

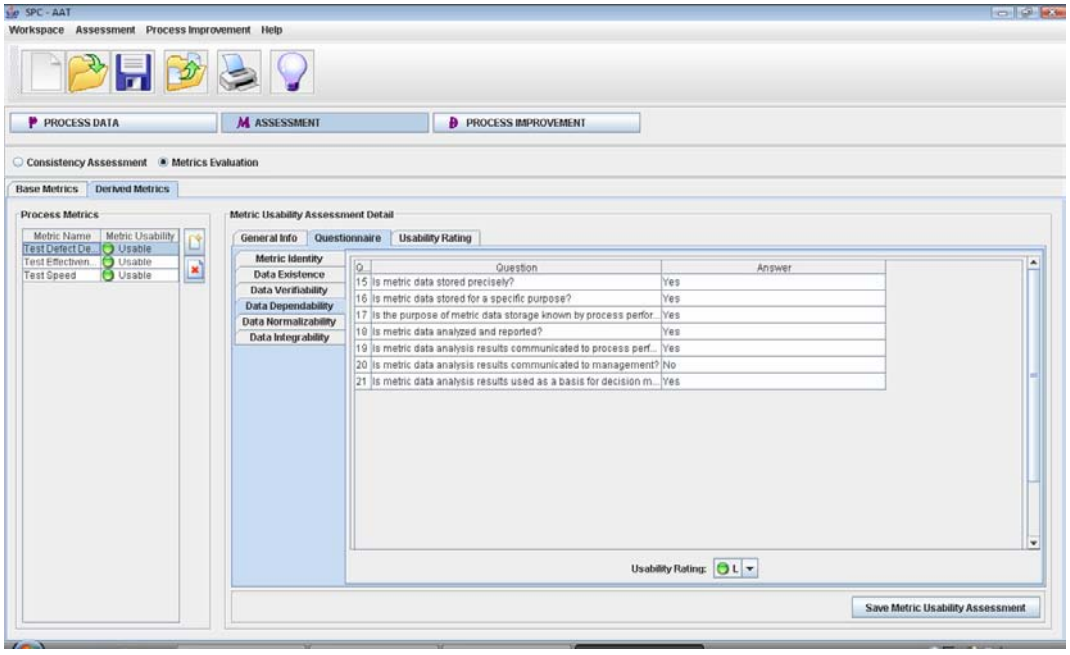


Figure 239 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

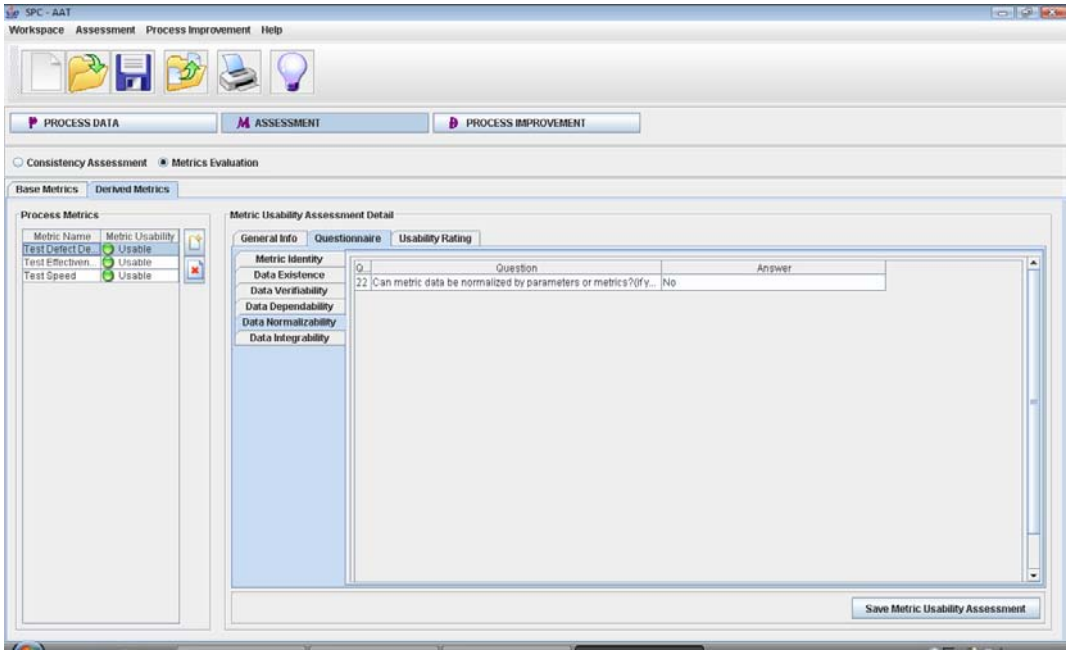


Figure 240 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

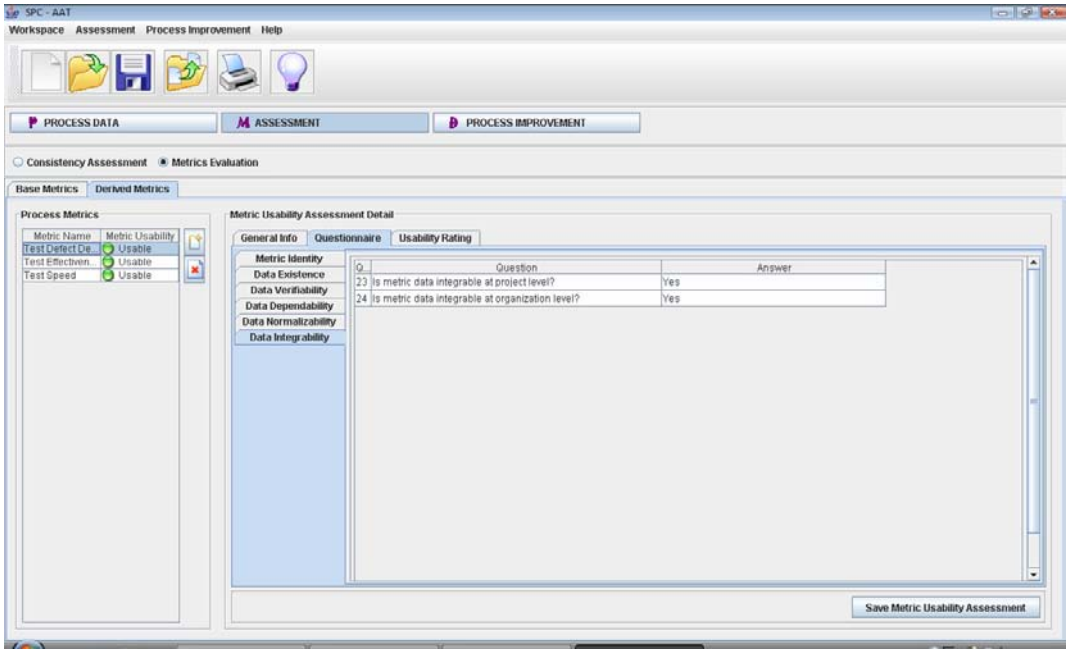


Figure 241 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

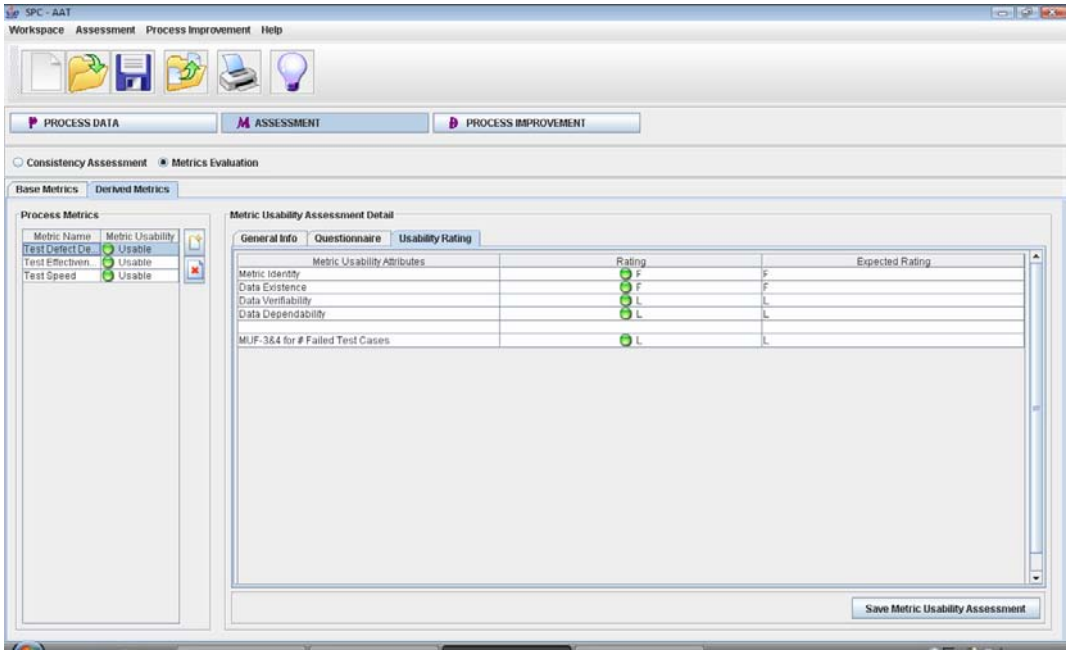


Figure 242 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

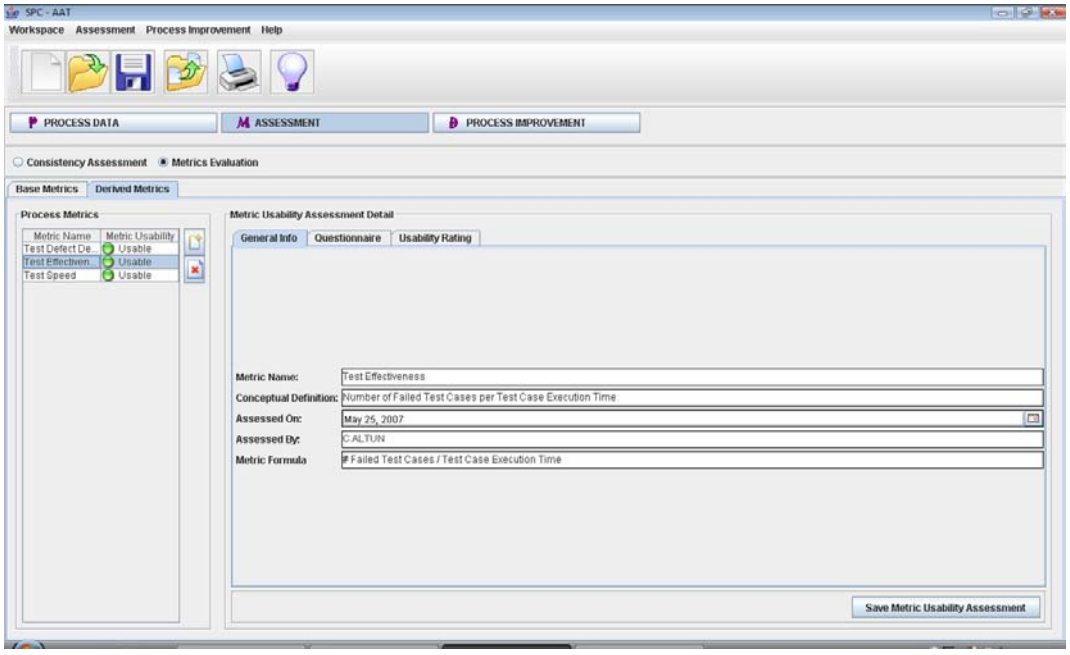


Figure 243 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

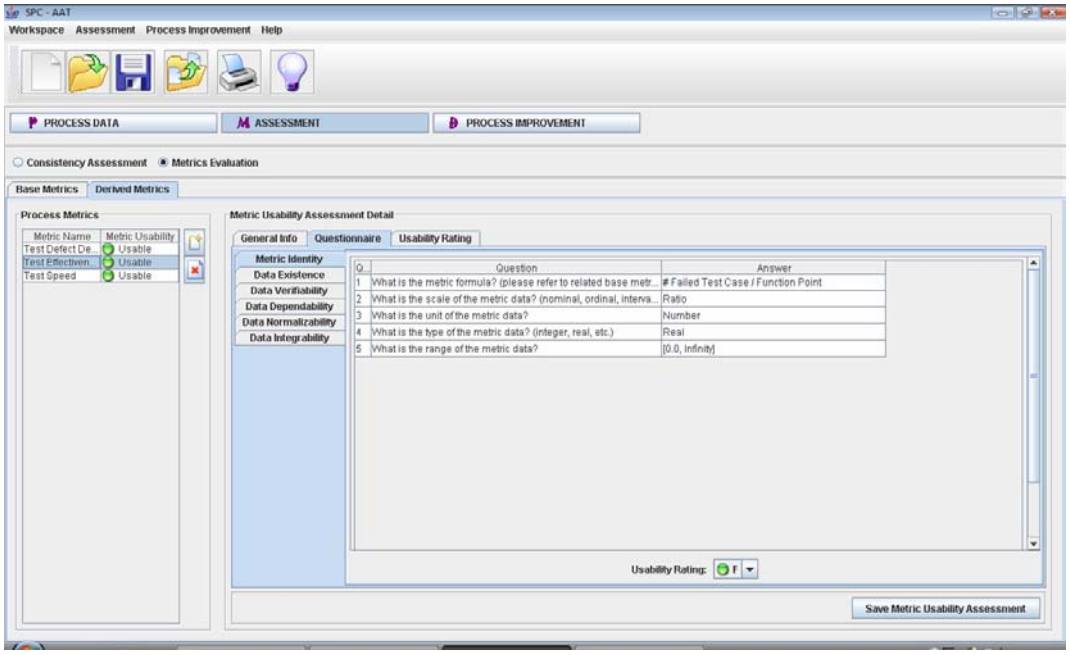


Figure 244 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

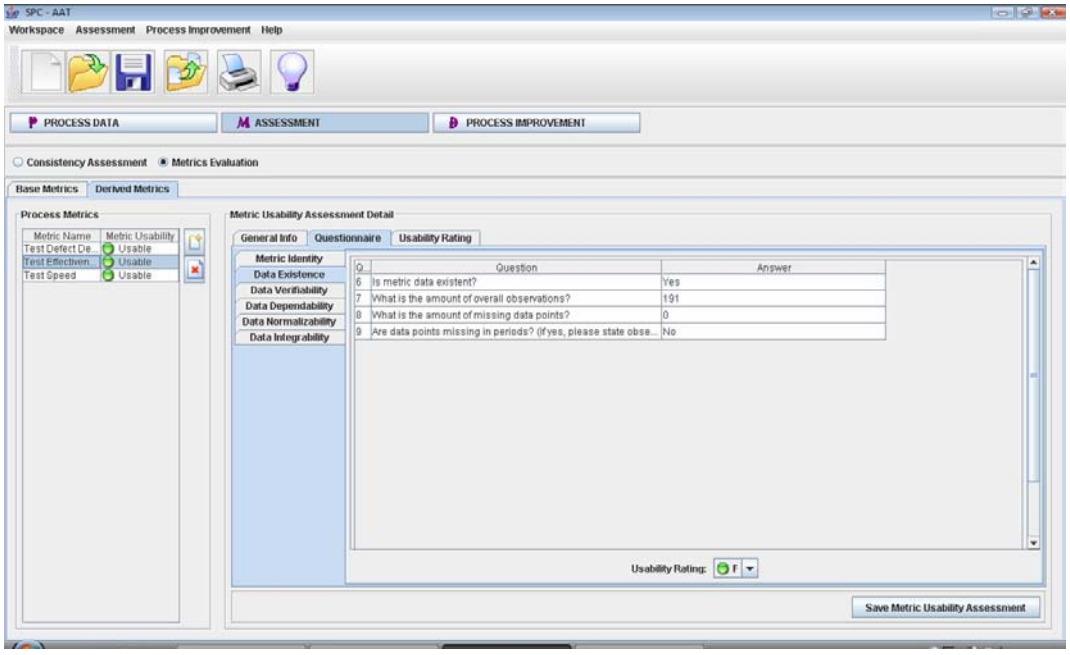


Figure 245 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

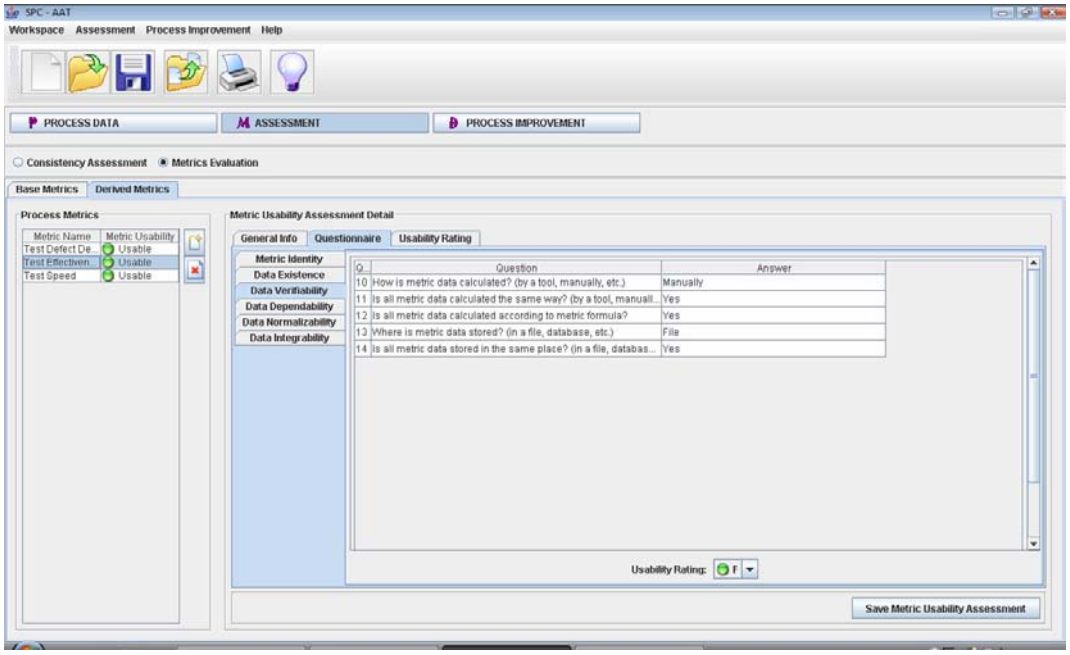


Figure 246 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

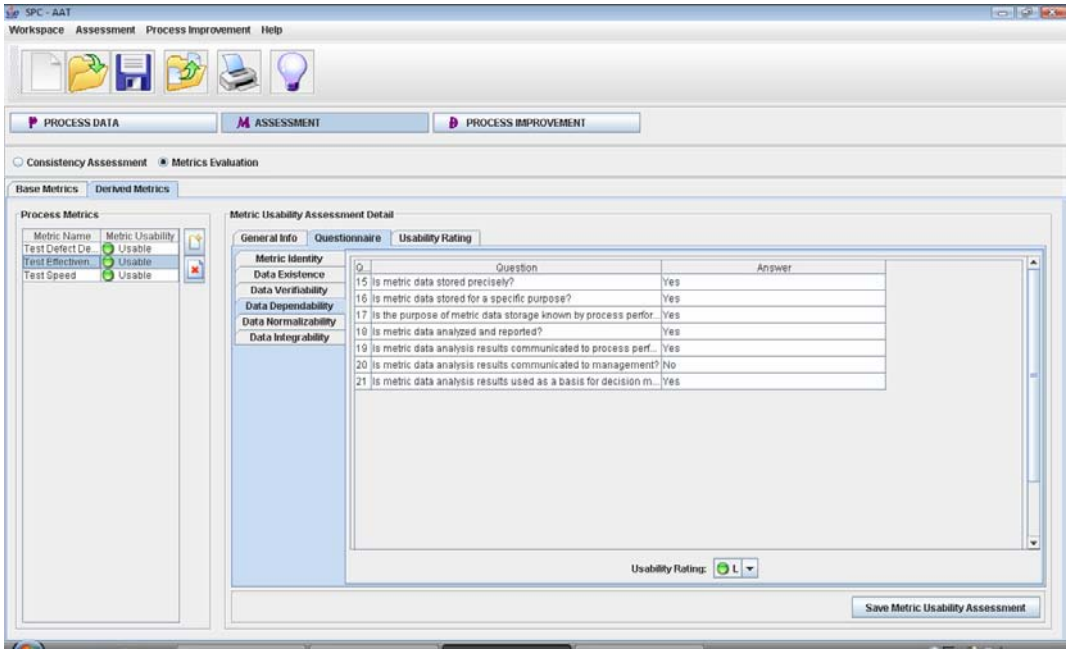


Figure 247 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

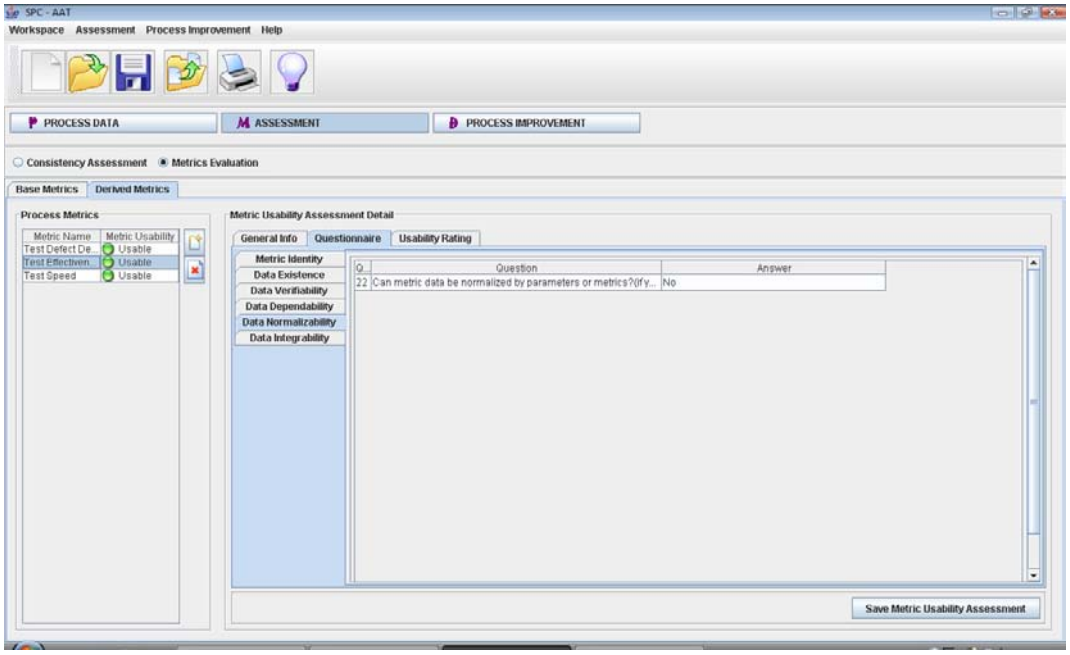


Figure 248 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

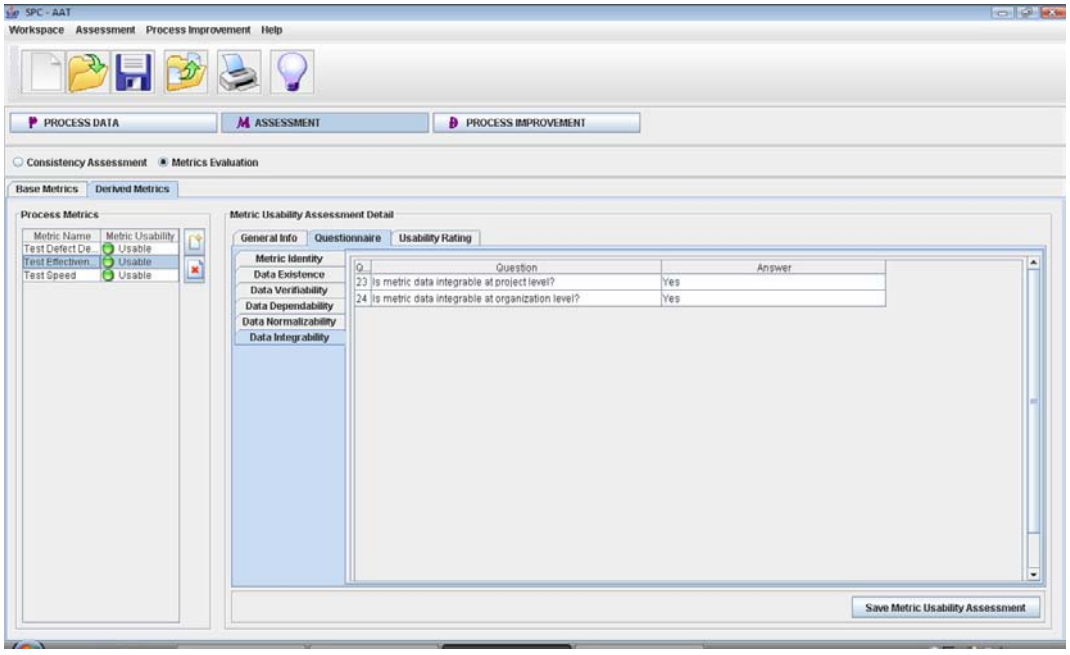


Figure 249 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

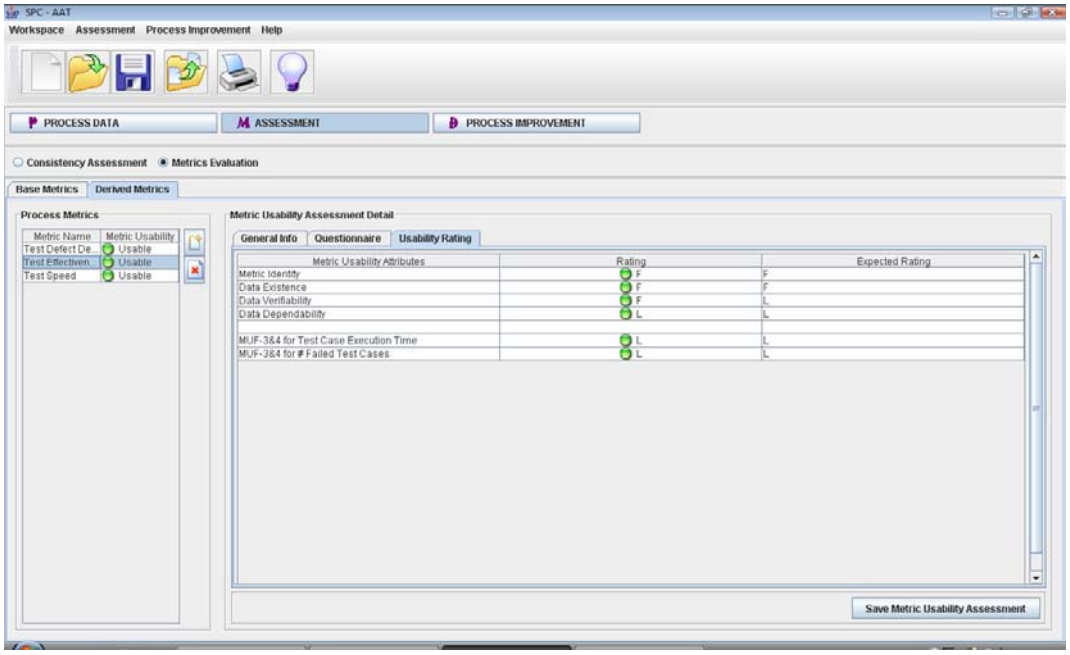


Figure 250 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

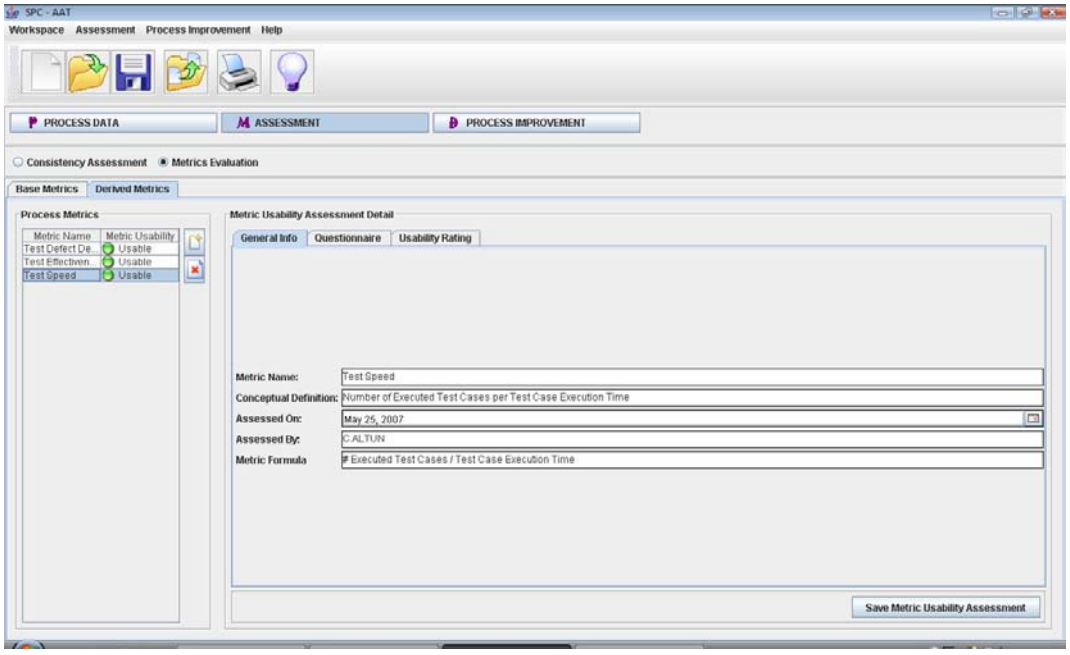


Figure 251 Metric Usability Questionnaire of “Test Speed” Derived Metric

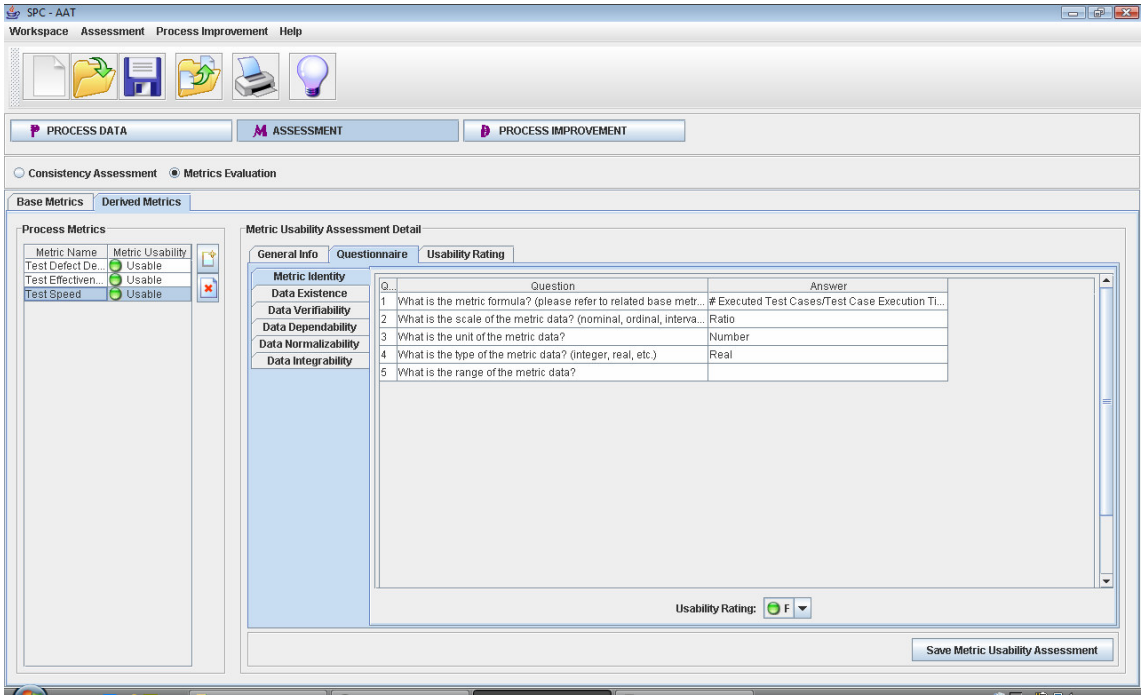


Figure 252 Metric Usability Questionnaire of “Test Speed” Derived Metric

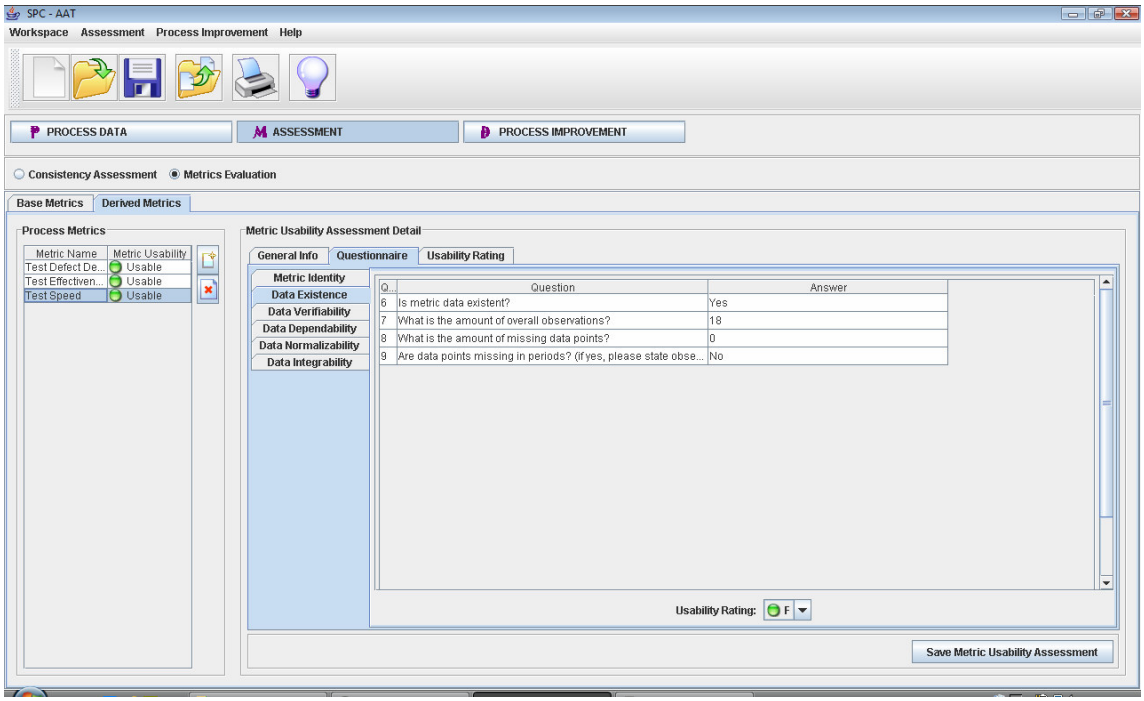


Figure 253 Metric Usability Questionnaire of “Test Speed” Derived Metric

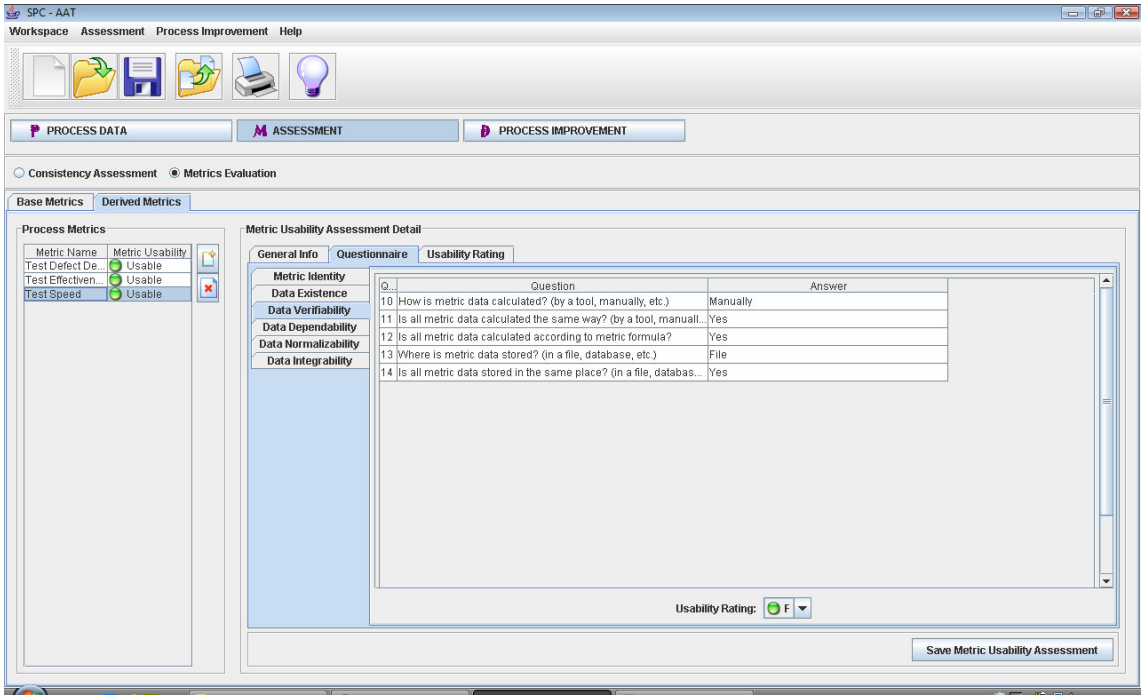


Figure 254 Metric Usability Questionnaire of “Test Speed” Derived Metric

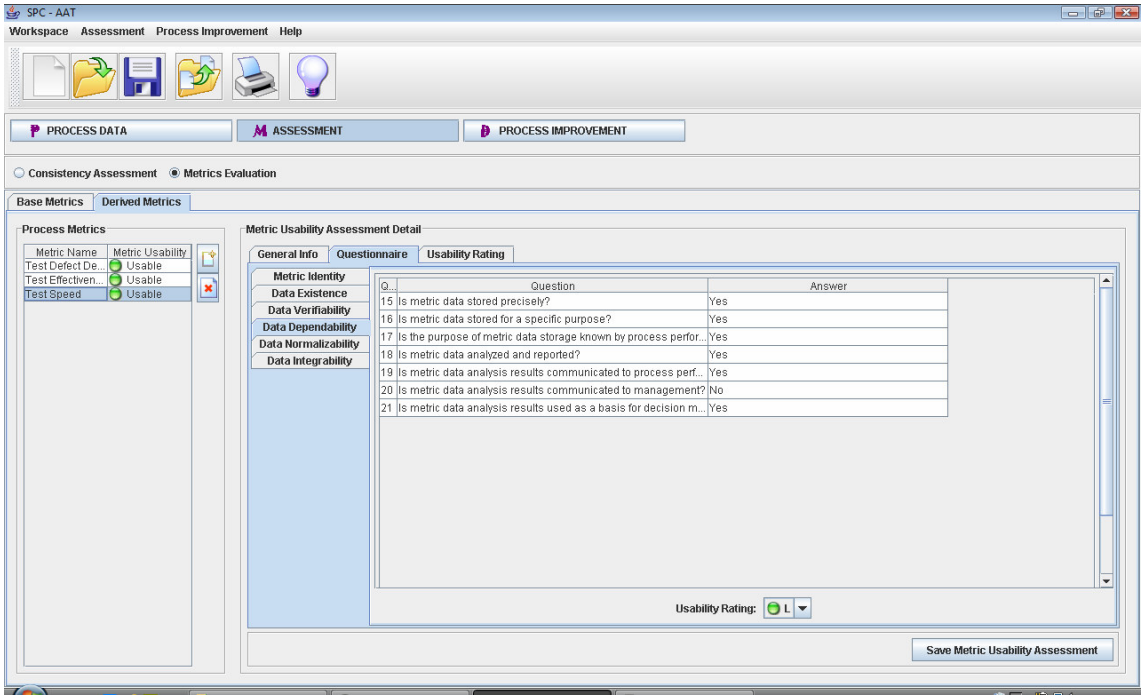


Figure 255 Metric Usability Questionnaire of “Test Speed” Derived Metric

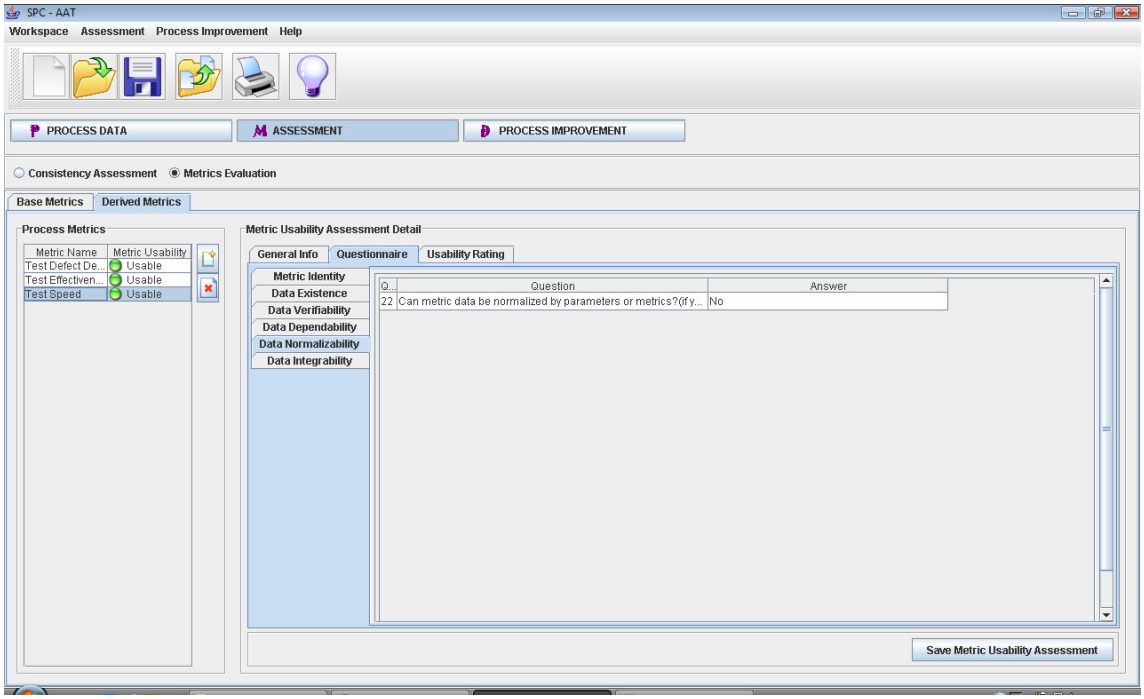


Figure 256 Metric Usability Questionnaire of “Test Speed” Derived Metric

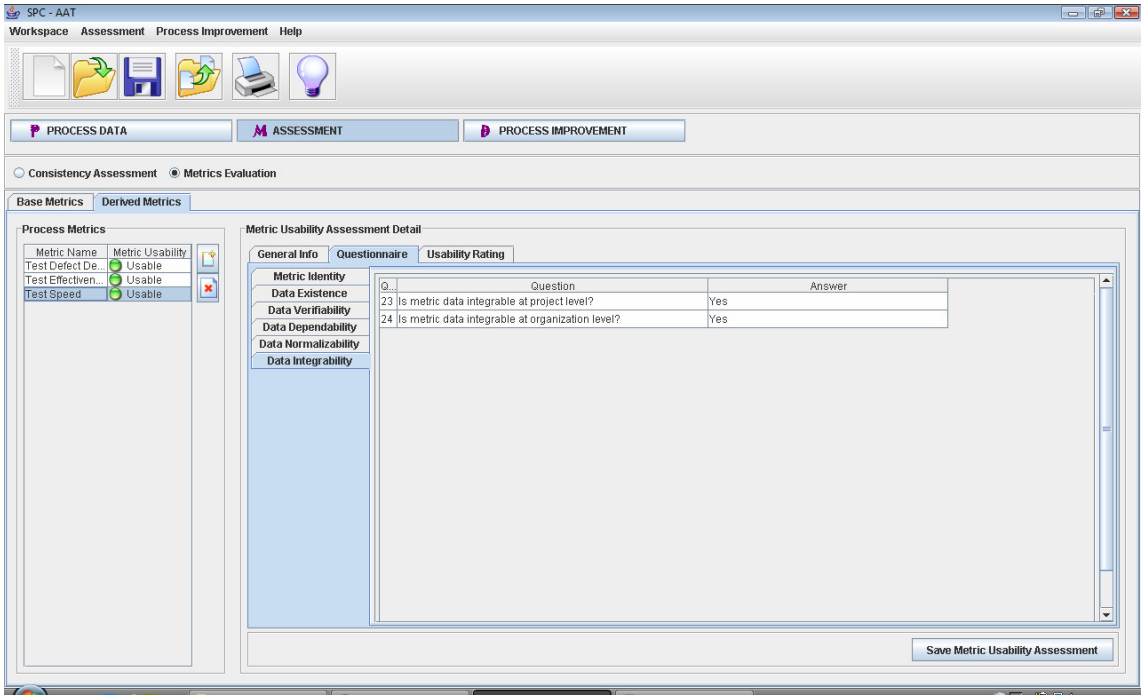


Figure 257 Metric Usability Questionnaire of “Test Speed” Derived Metric

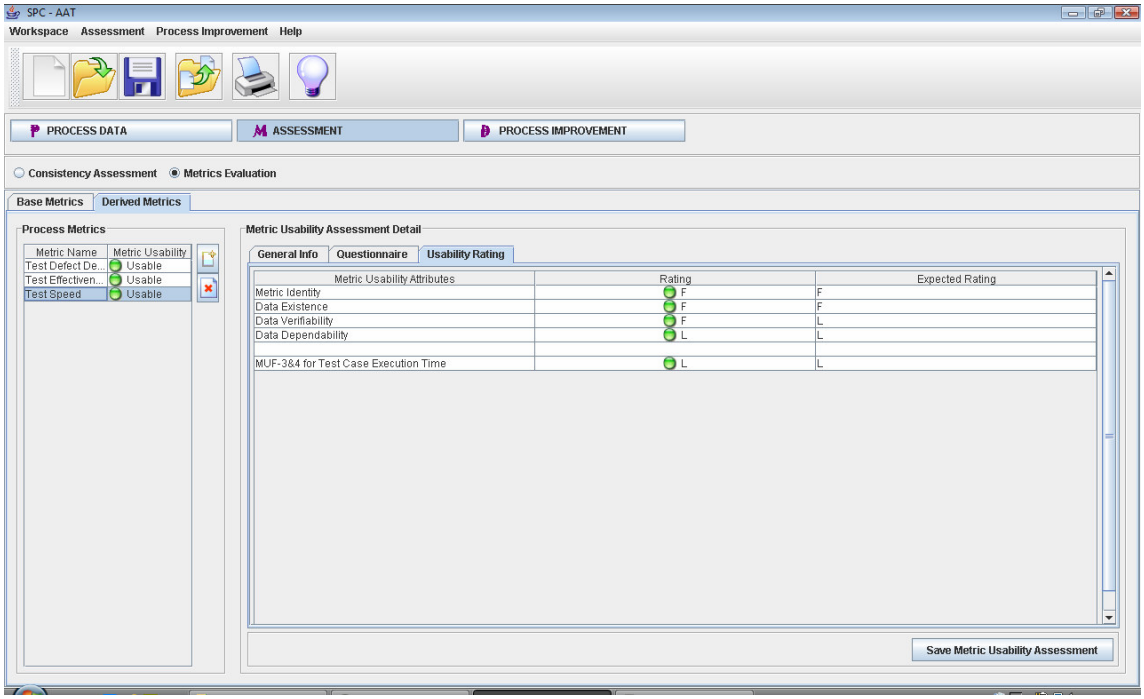


Figure 258 Metric Usability Questionnaire of “Test Speed” Derived Metric

Metric Usability Questionnaires of Case Study B

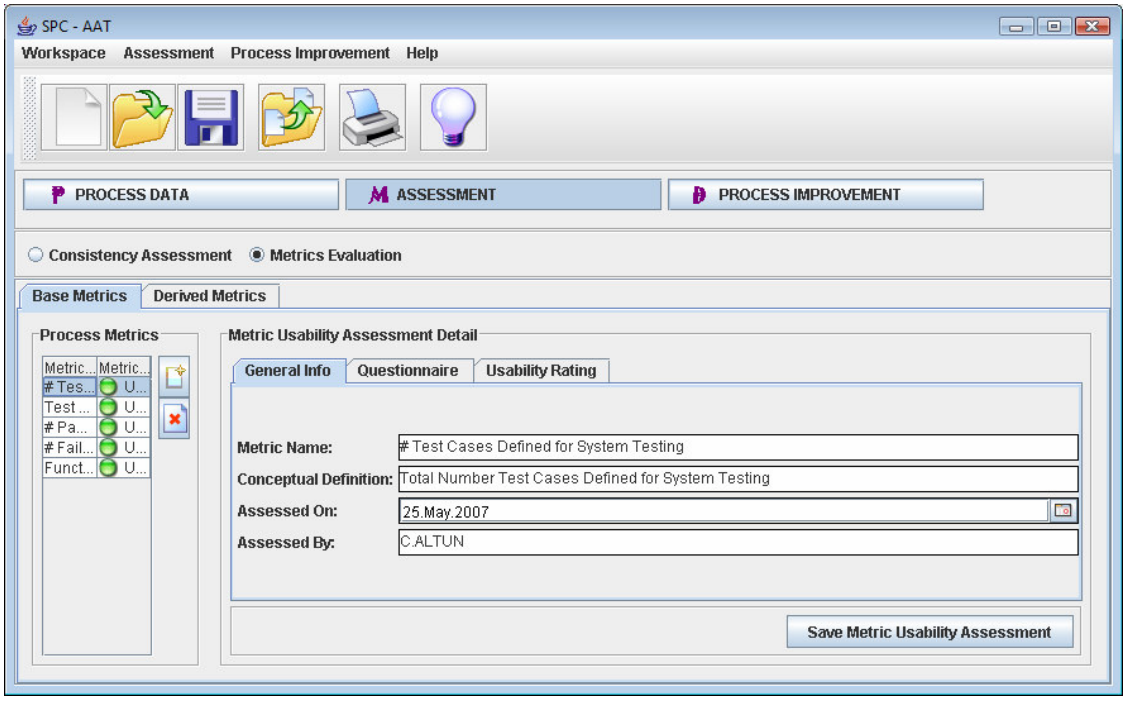


Figure 259 Metric Usability Questionnaire of “# Test Cases” Base Metric

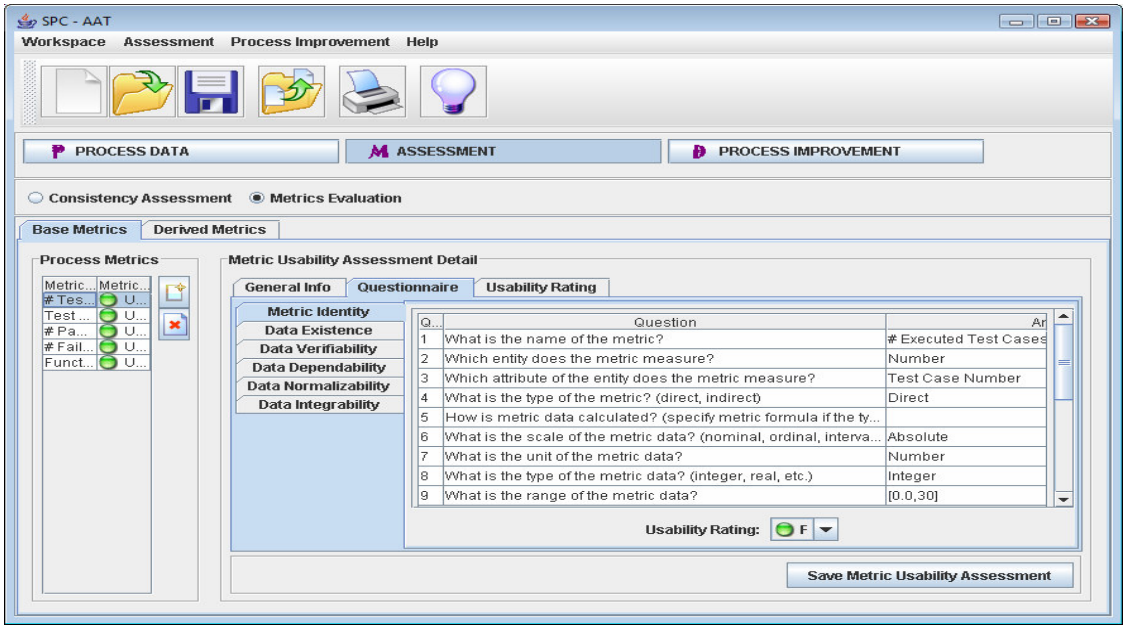


Figure 260 Metric Usability Questionnaire of “# Test Cases” Base Metric

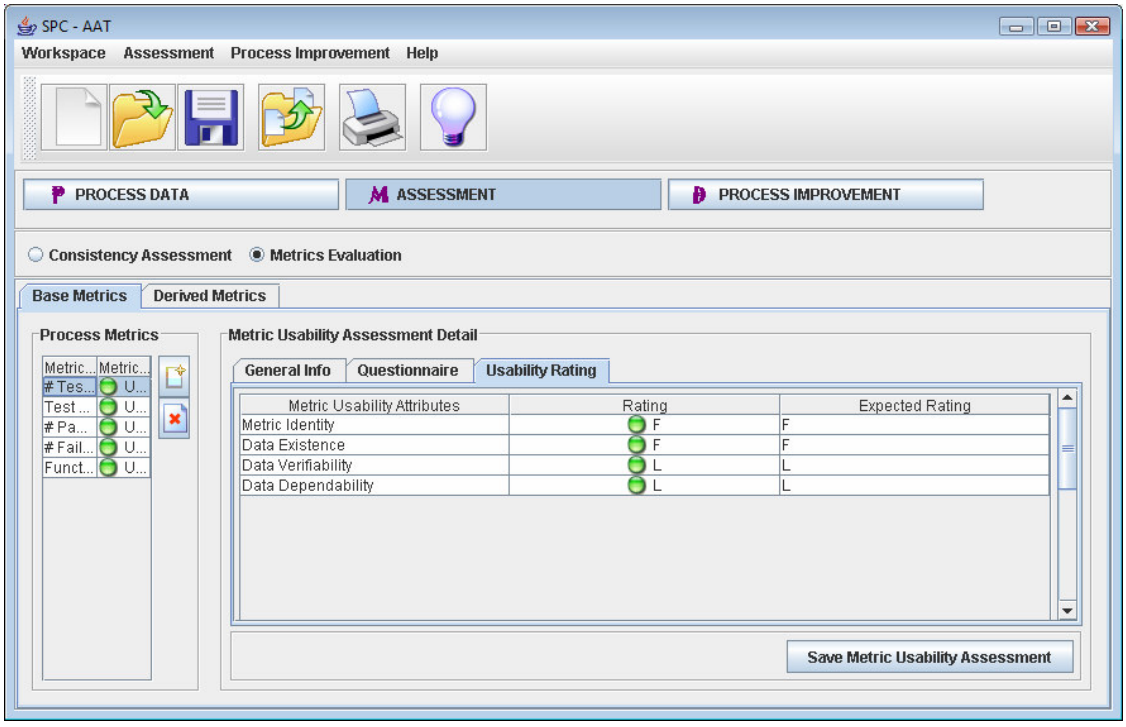


Figure 261 Metric Usability Questionnaire of “# Test Cases” Base Metric

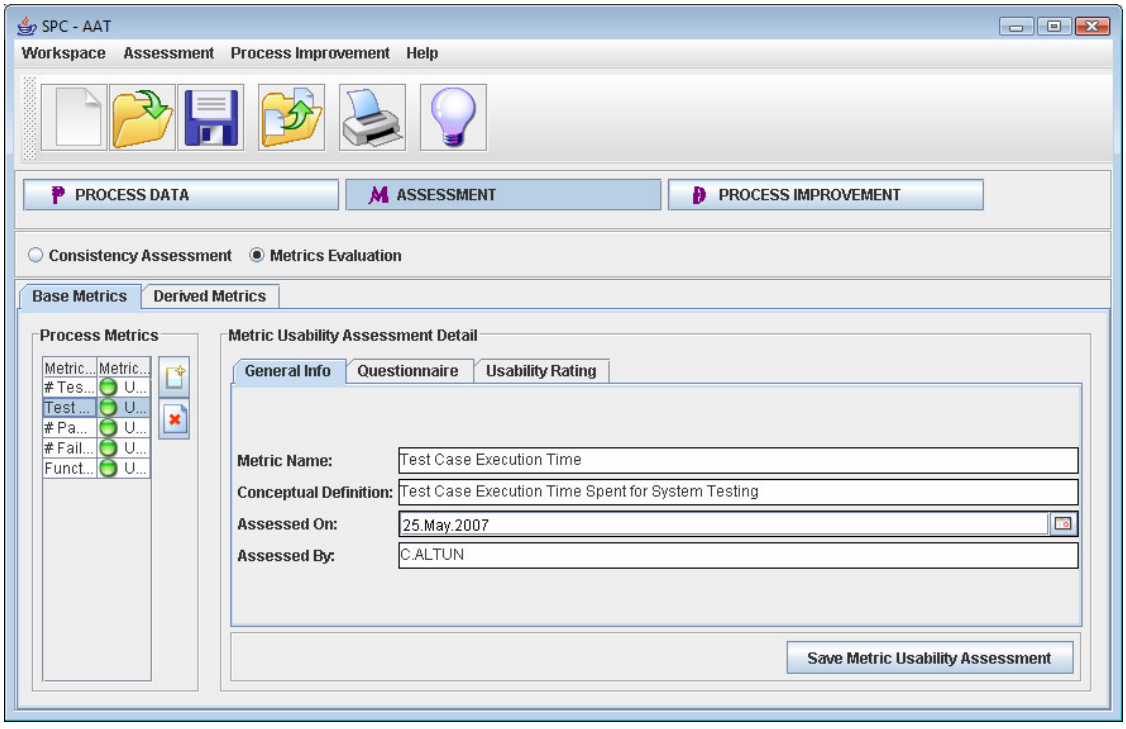


Figure 262 Metric Usability Questionnaire of “Test Case Execution Time” Base Metric

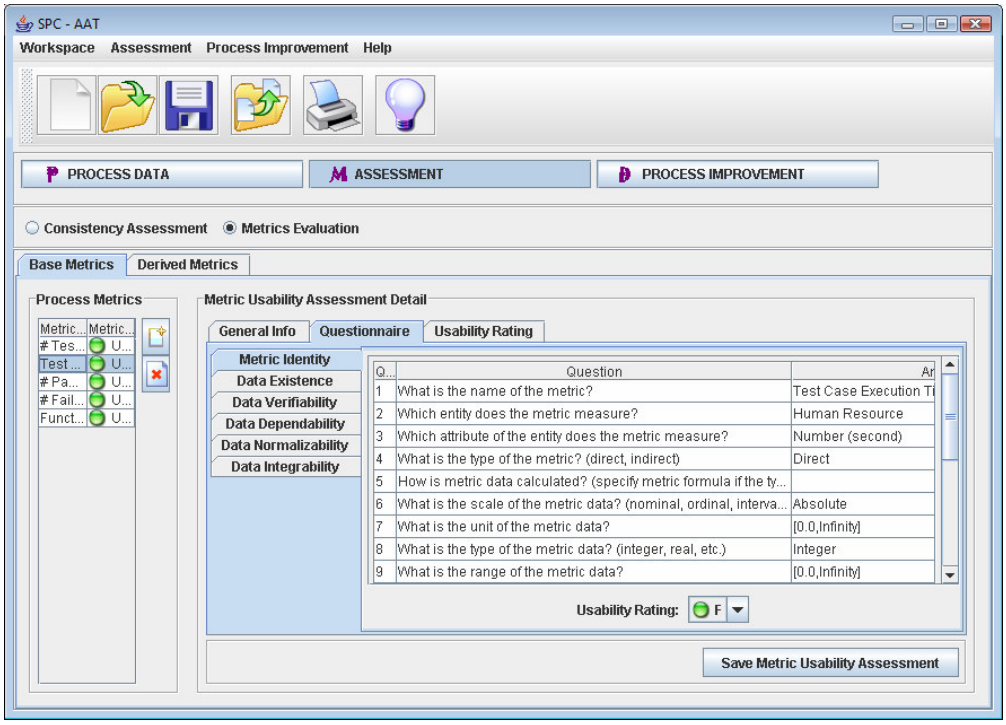


Figure 263 Metric Usability Questionnaire of “Test Case Execution Time” Base Metric

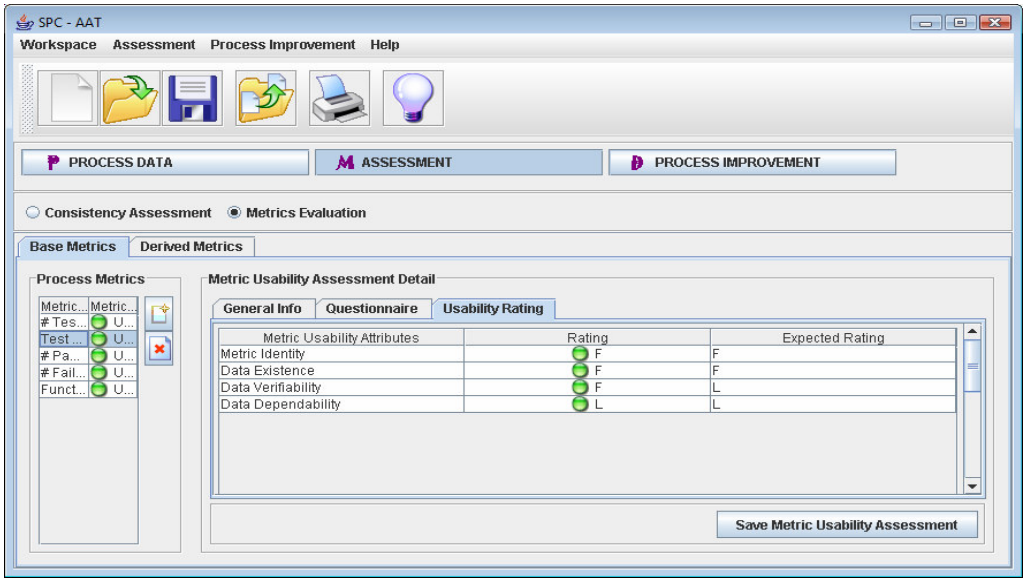


Figure 264 Metric Usability Questionnaire of “Test Case Execution Time” Base Metric

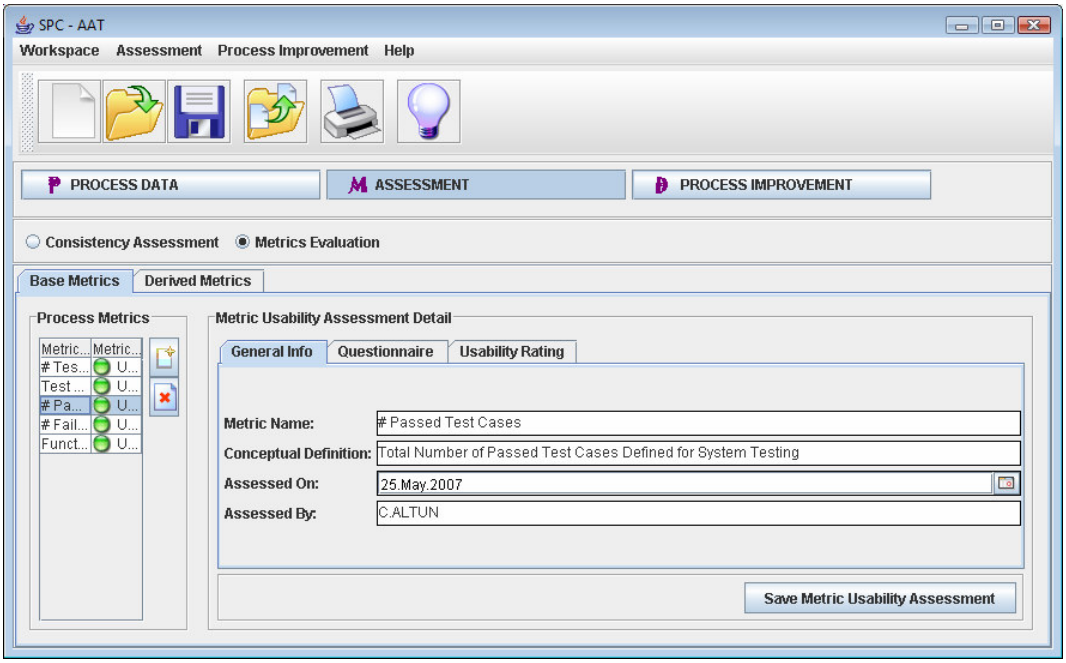


Figure 265 Metric Usability Questionnaire of “# Passed Test Cases” Base Metric

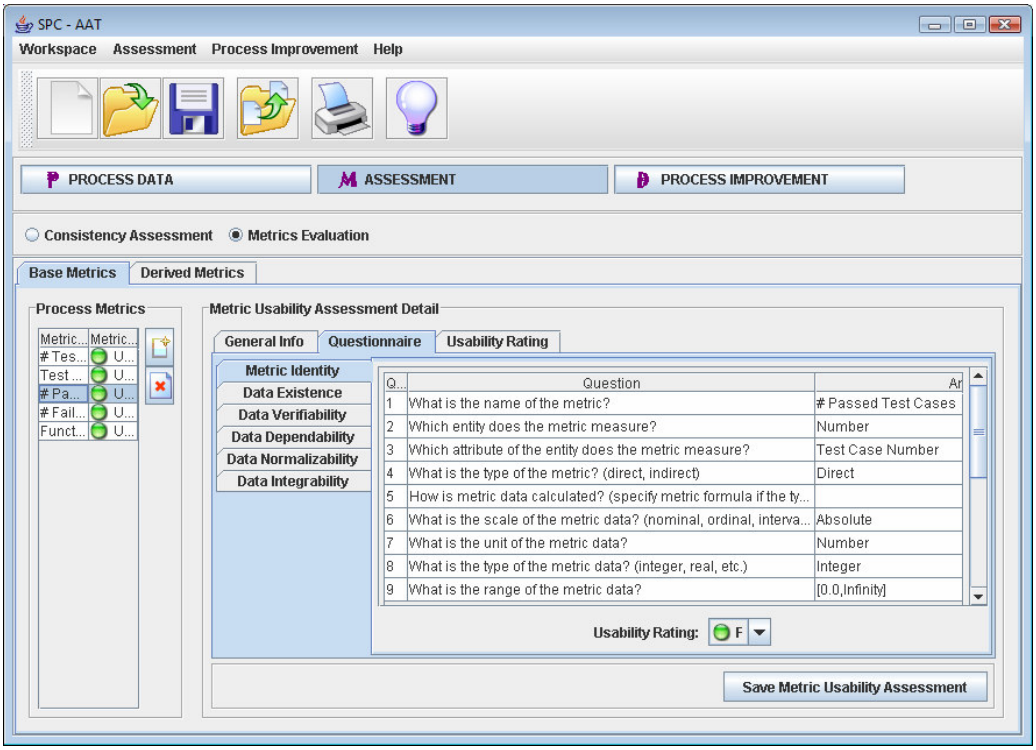


Figure 266 Metric Usability Questionnaire of “# Passed Test Cases” Base Metric

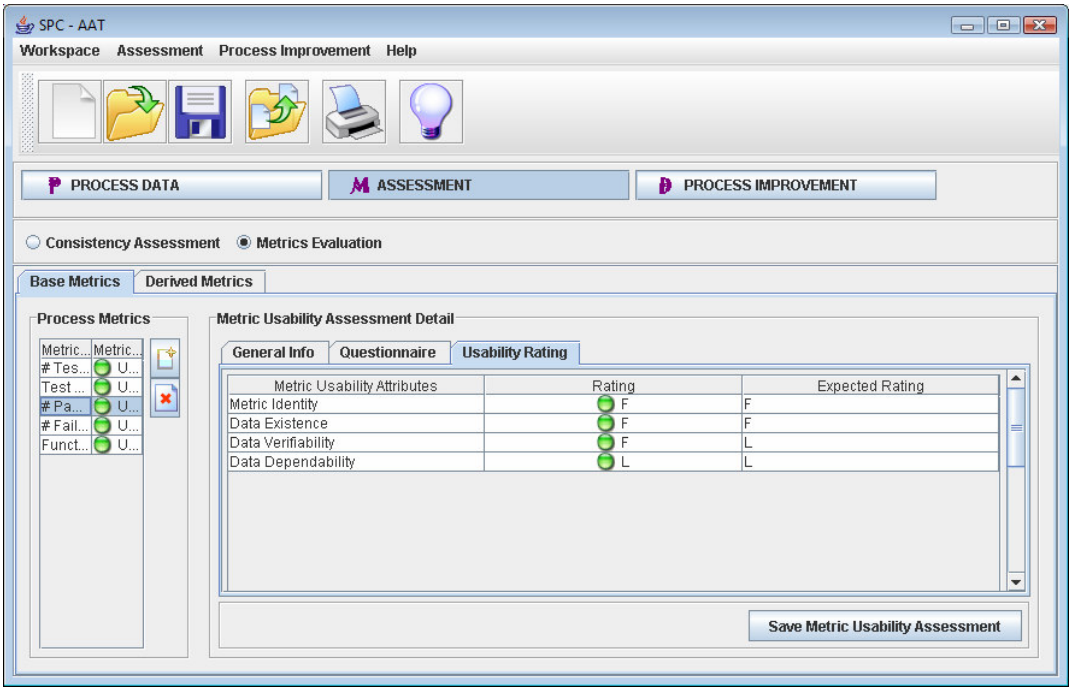


Figure 267 Metric Usability Questionnaire of “# Passed Test Cases” Base Metric

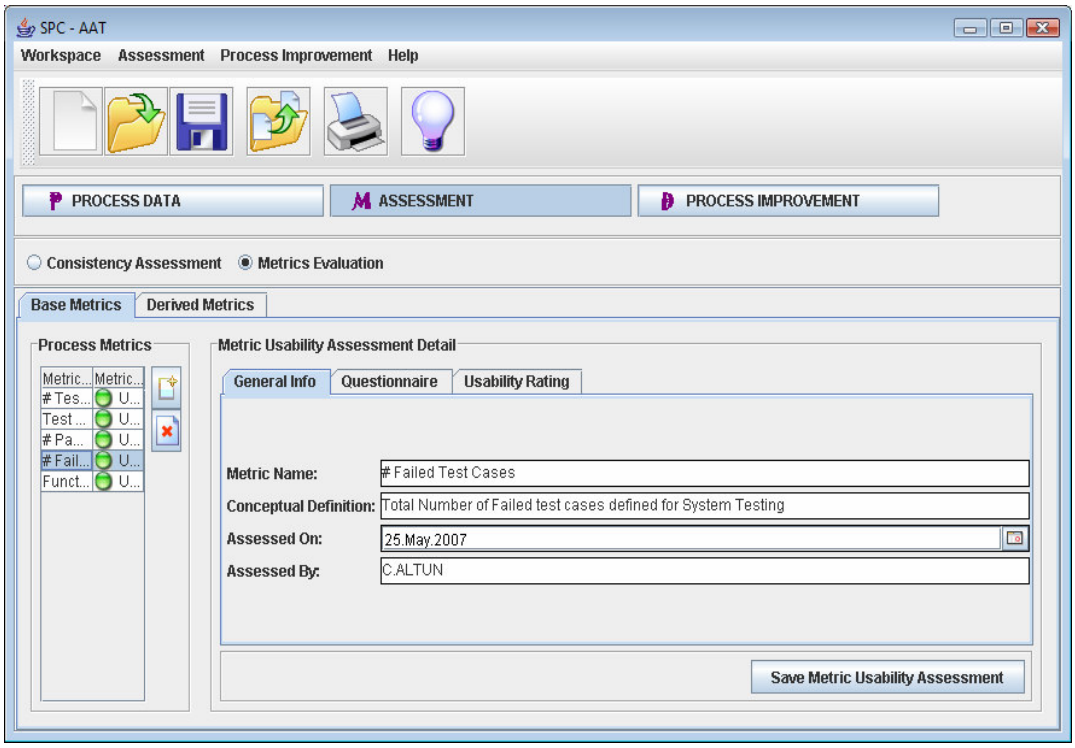


Figure 268 Metric Usability Questionnaire of “# Failed Test Cases” Base Metric

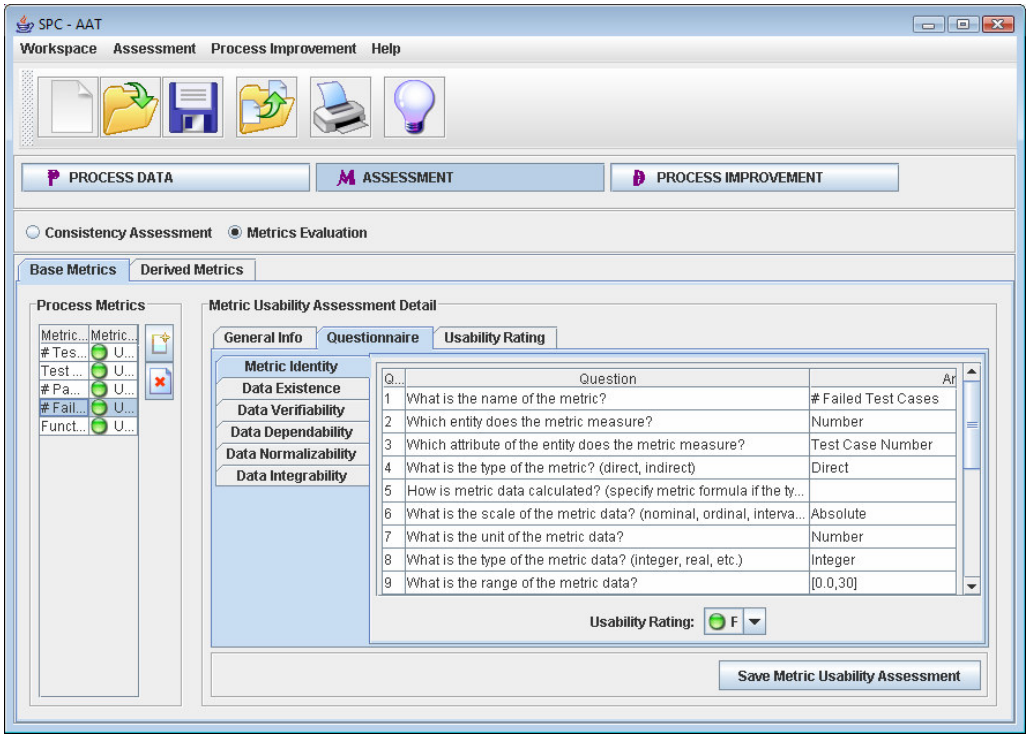


Figure 269 Metric Usability Questionnaire of “# Failed Test Cases” Base Metric

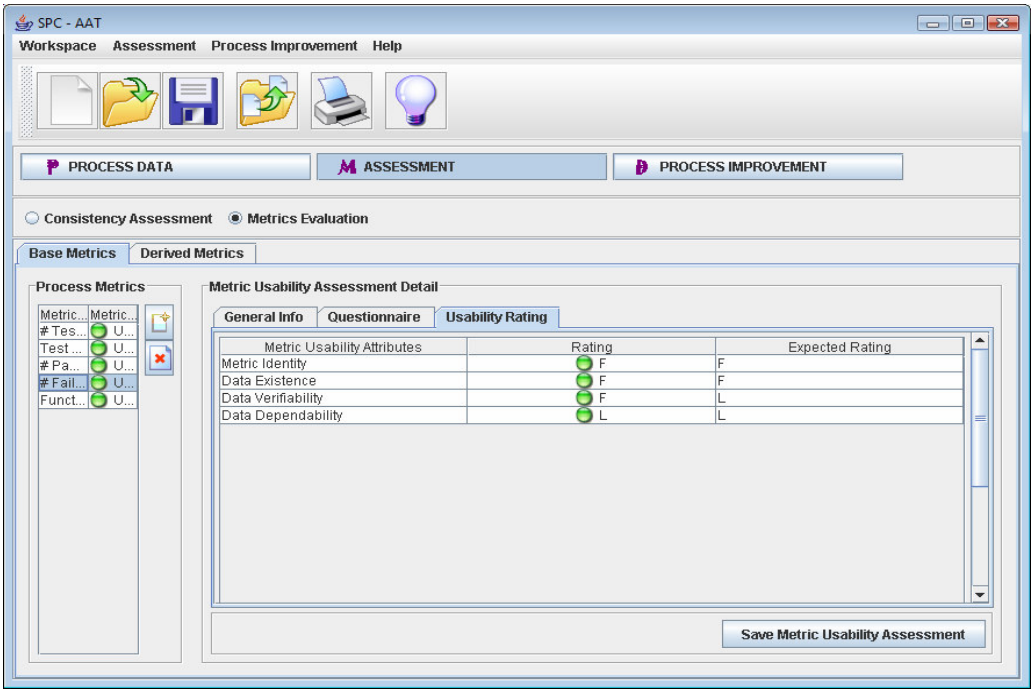


Figure 270 Metric Usability Questionnaire of “# Failed Test Cases” Base Metric

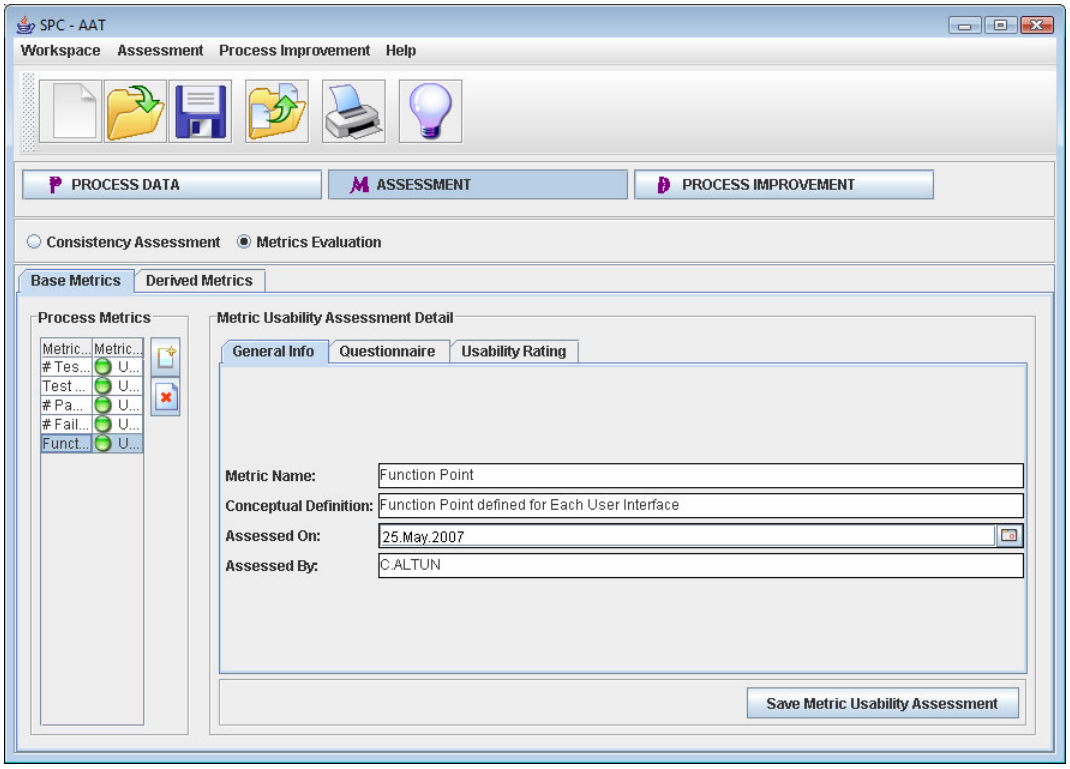


Figure 271 Metric Usability Questionnaire of “Function Point” Base Metric

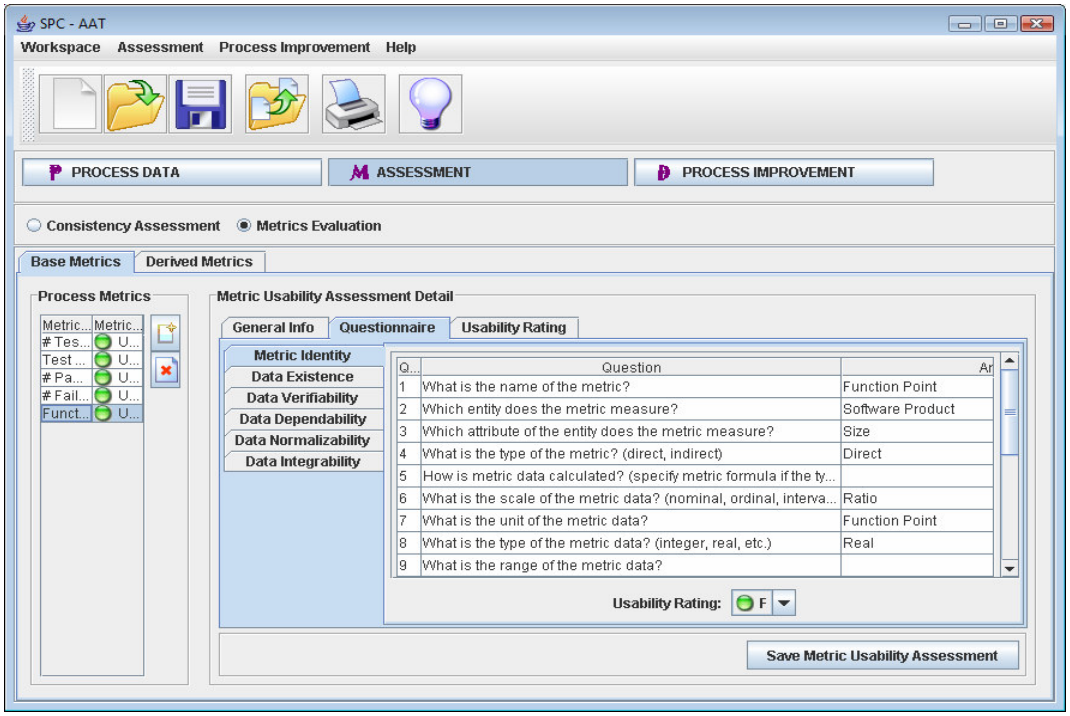


Figure 272 Metric Usability Questionnaire of “Function Point” Base Metric

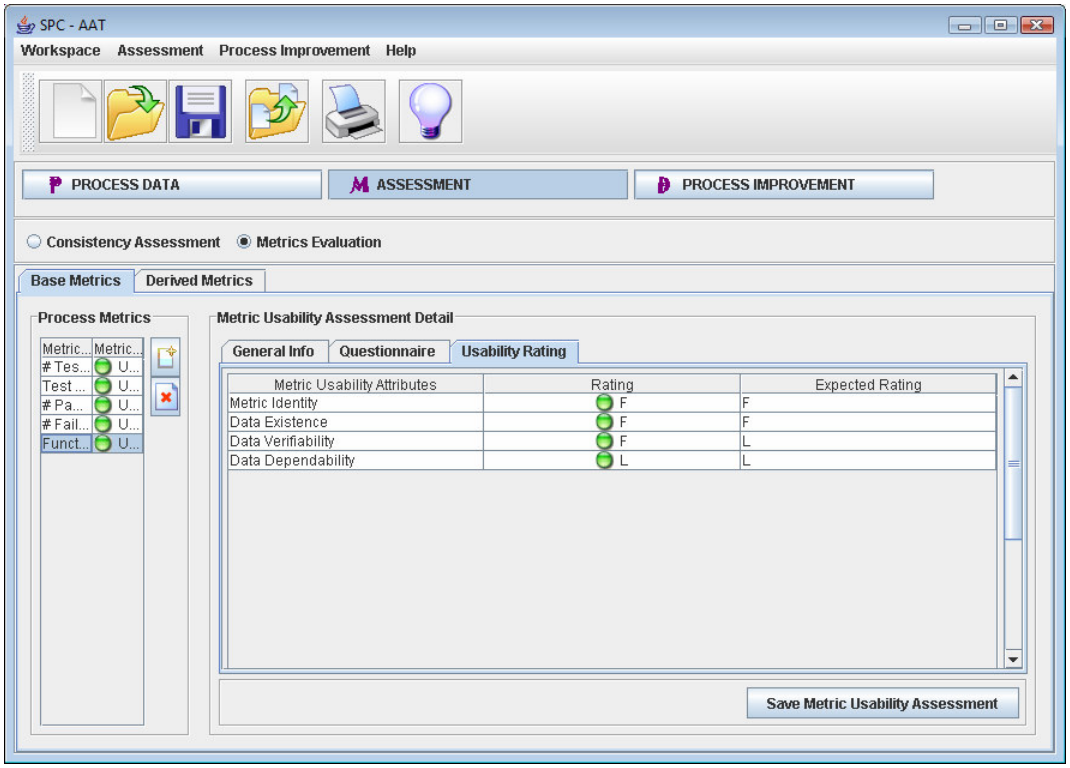


Figure 273 Metric Usability Questionnaire of “Function Point” Base Metric

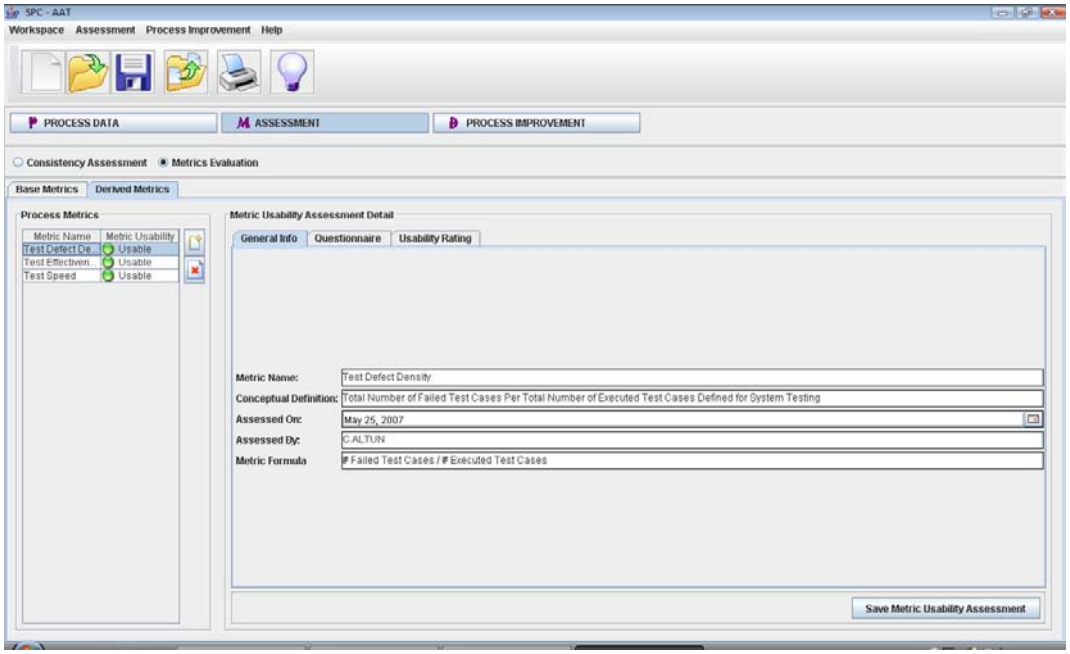


Figure 274 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

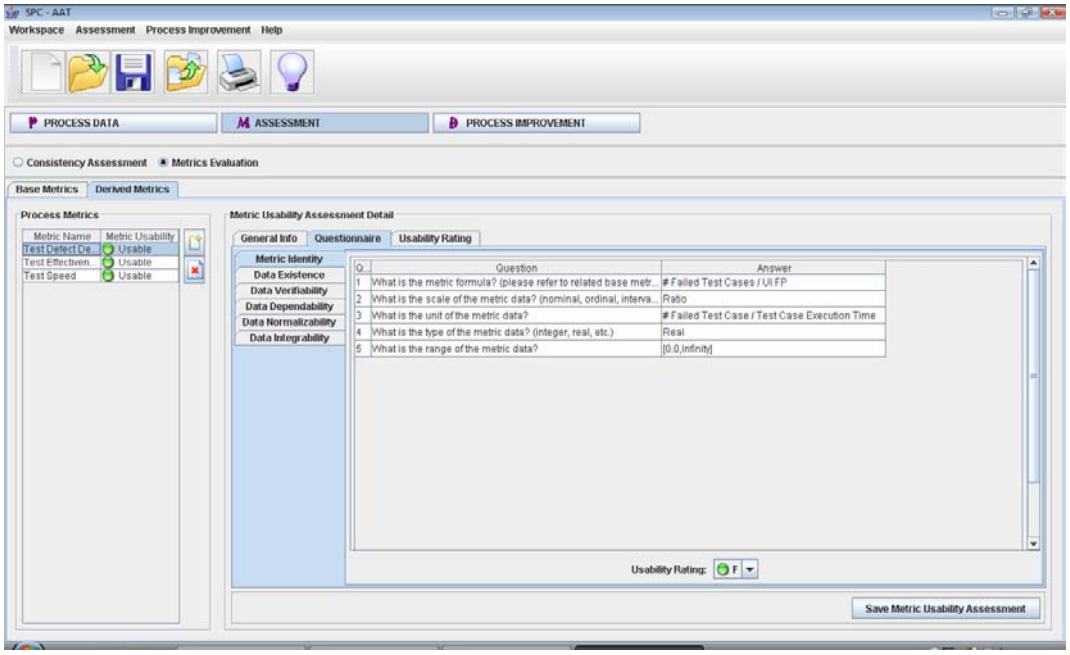


Figure 275 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

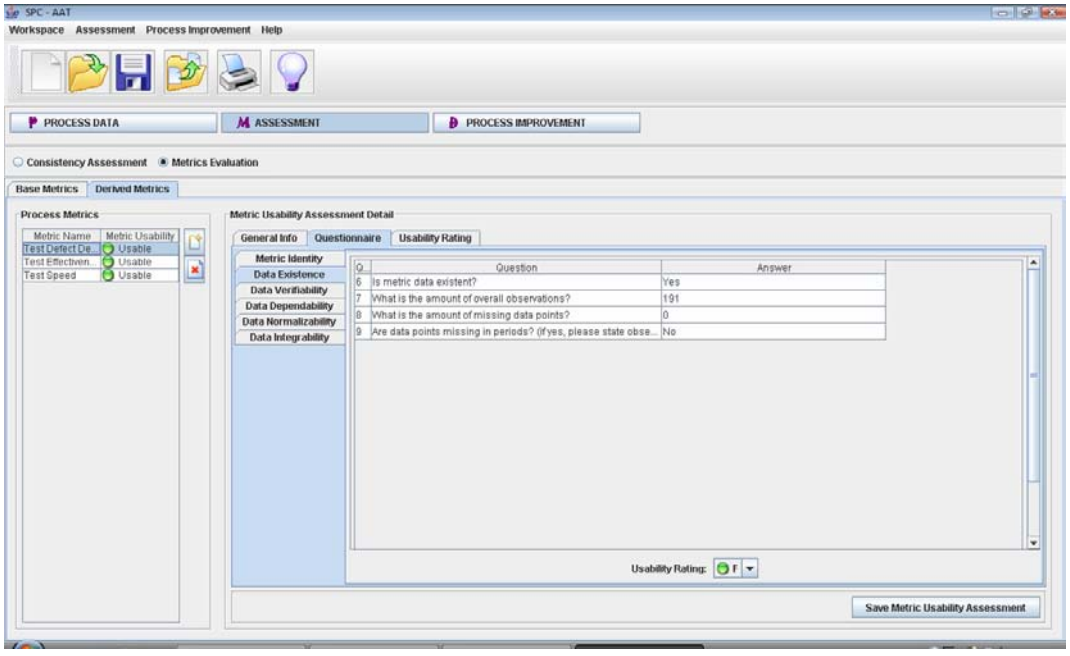


Figure 276 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

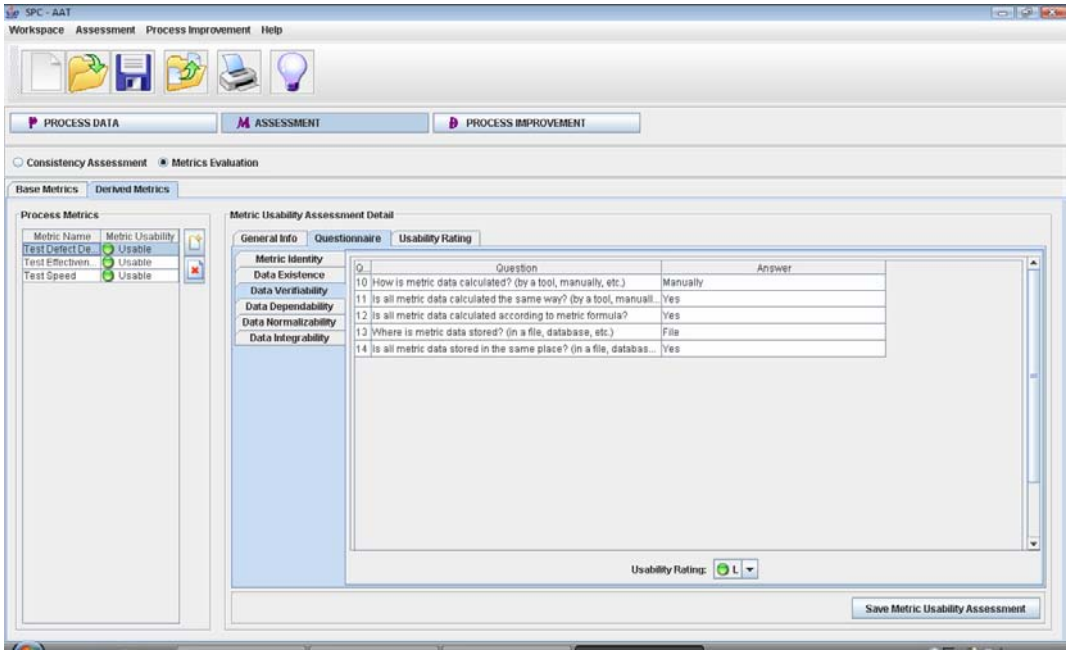


Figure 277 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

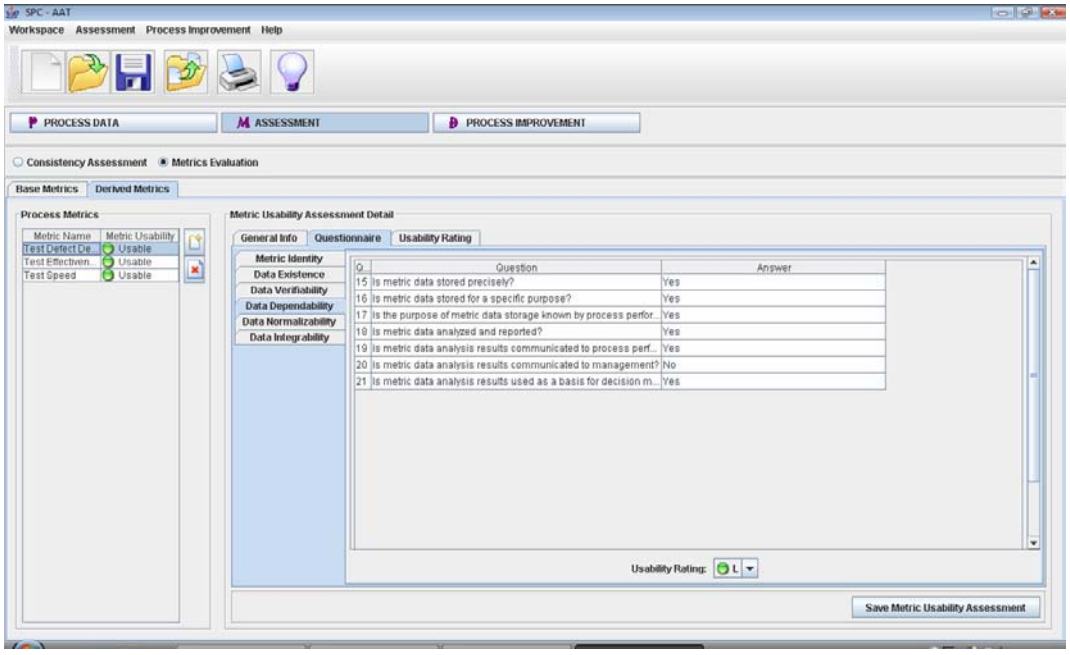


Figure 278 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

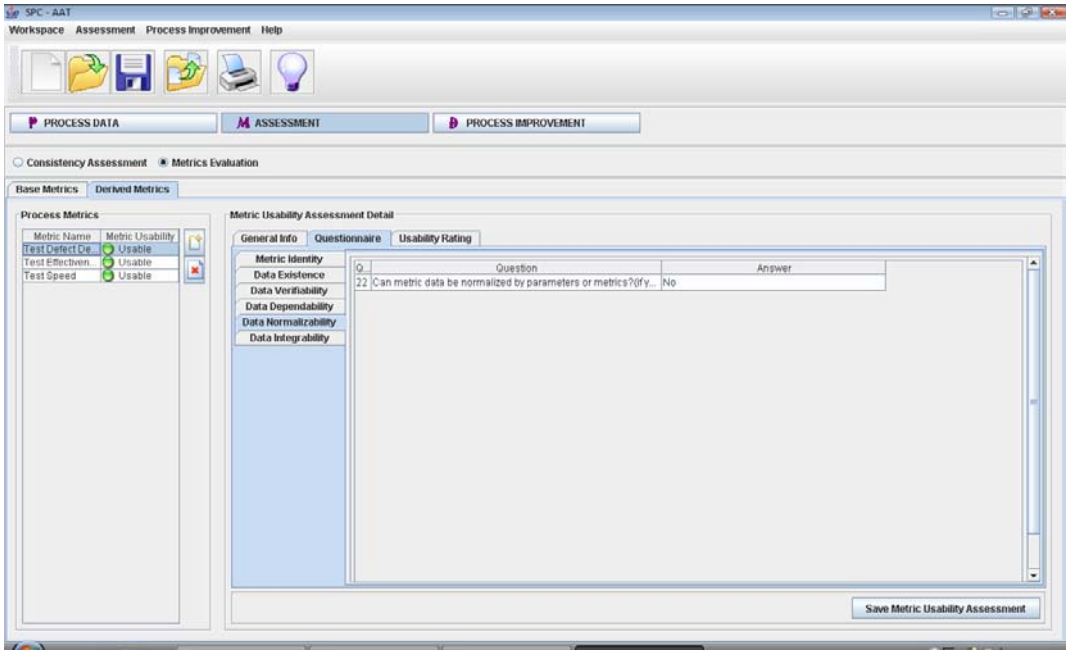


Figure 279 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

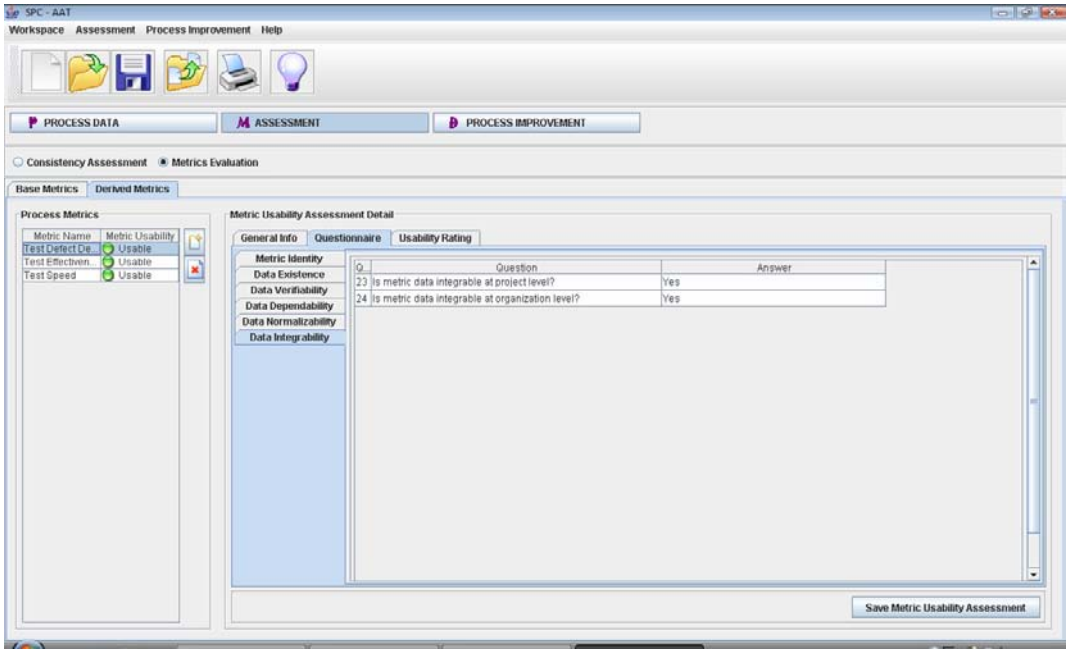


Figure 280 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

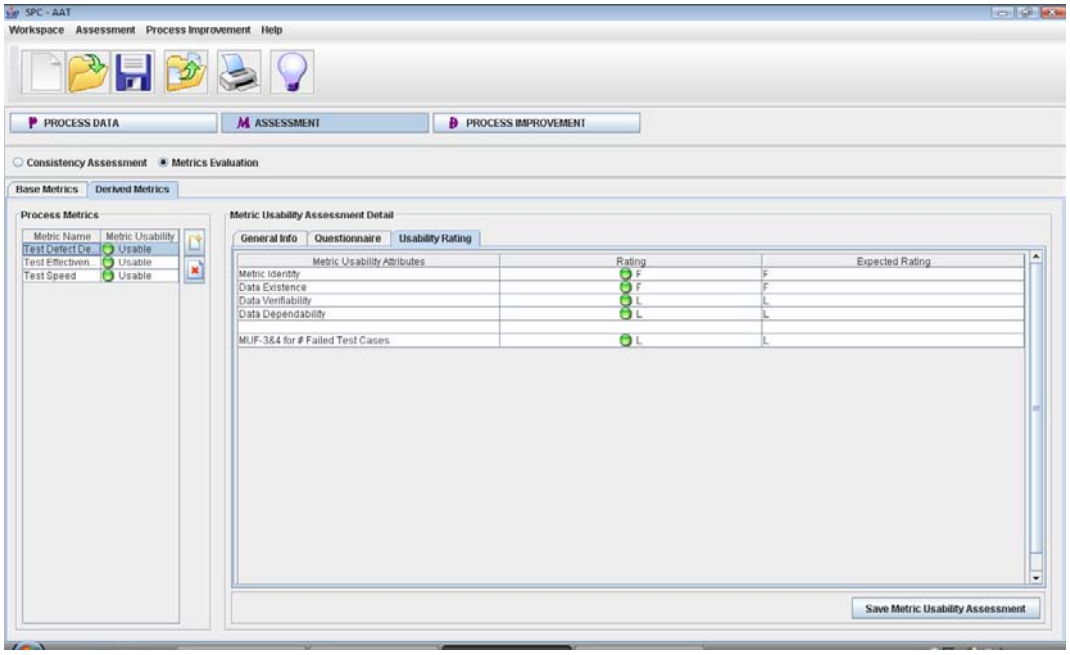


Figure 281 Metric Usability Questionnaire of “Test Defect Density” Derived Metric

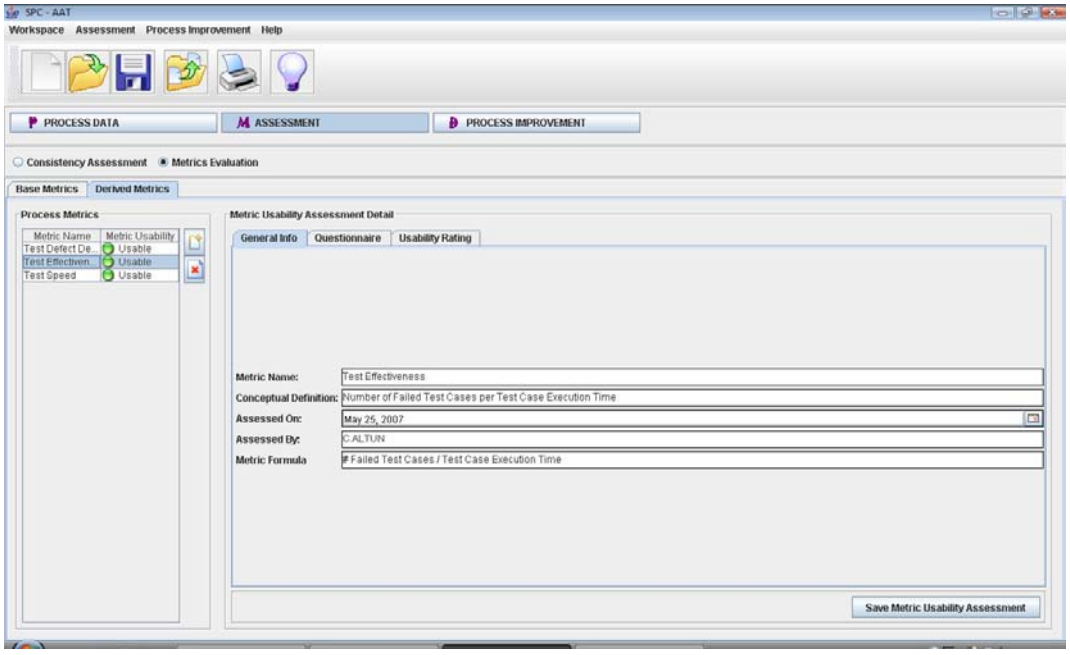


Figure 282 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

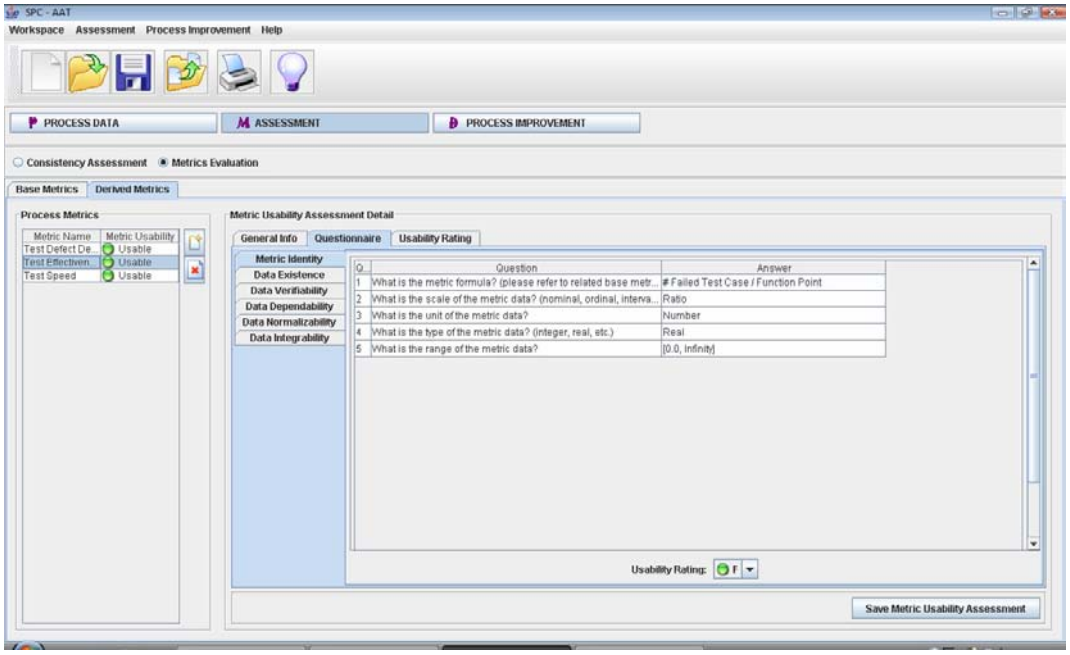


Figure 283 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

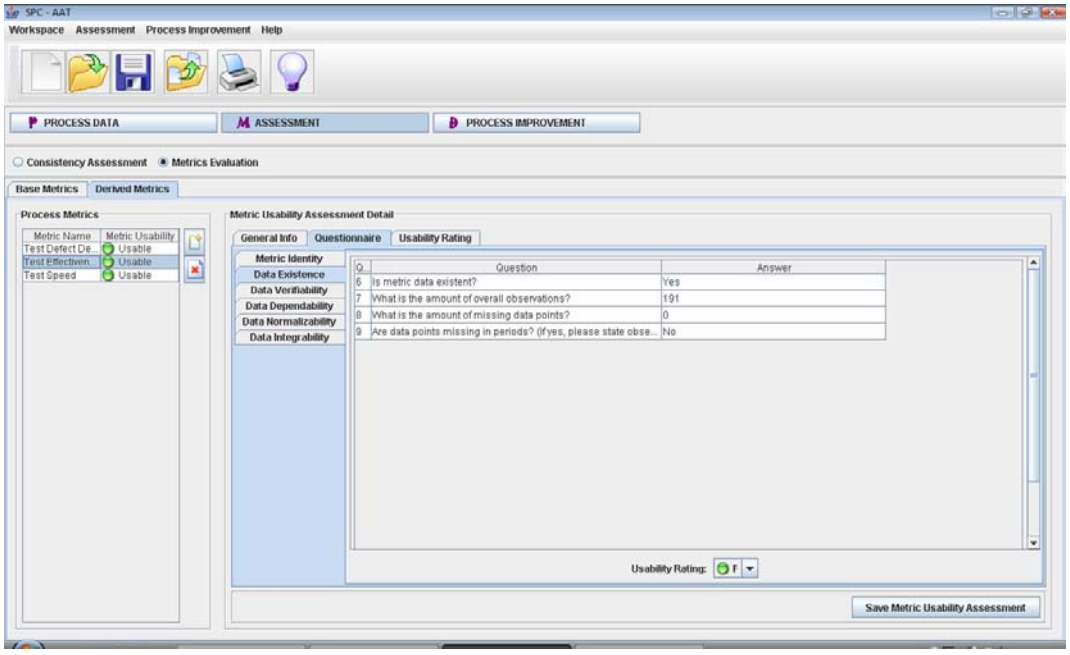


Figure 284 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

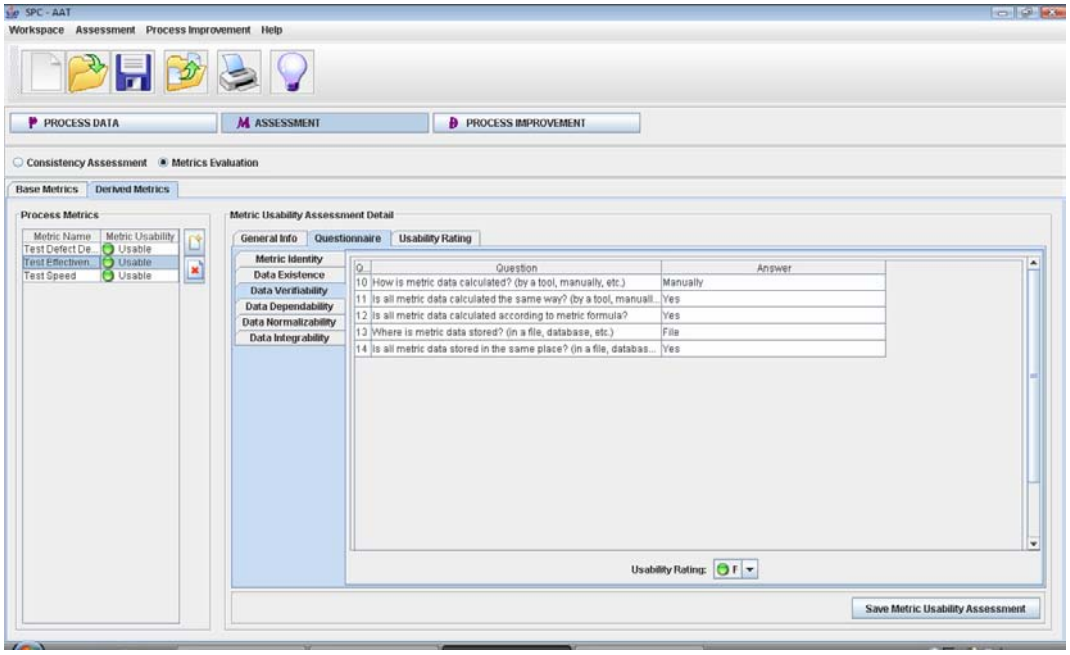


Figure 285 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

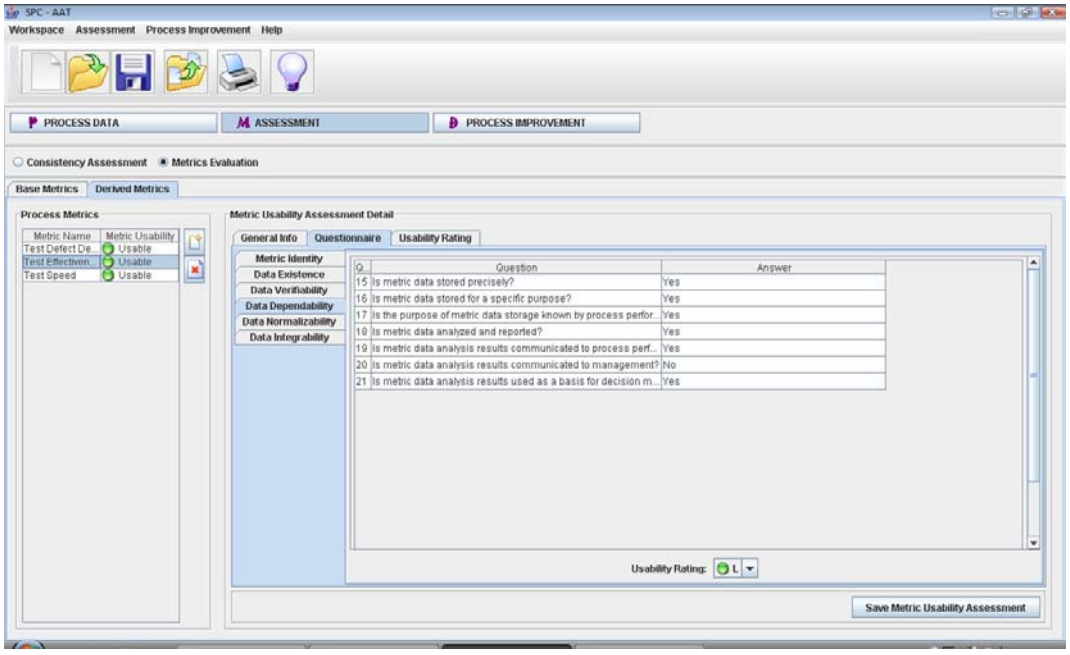


Figure 286 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

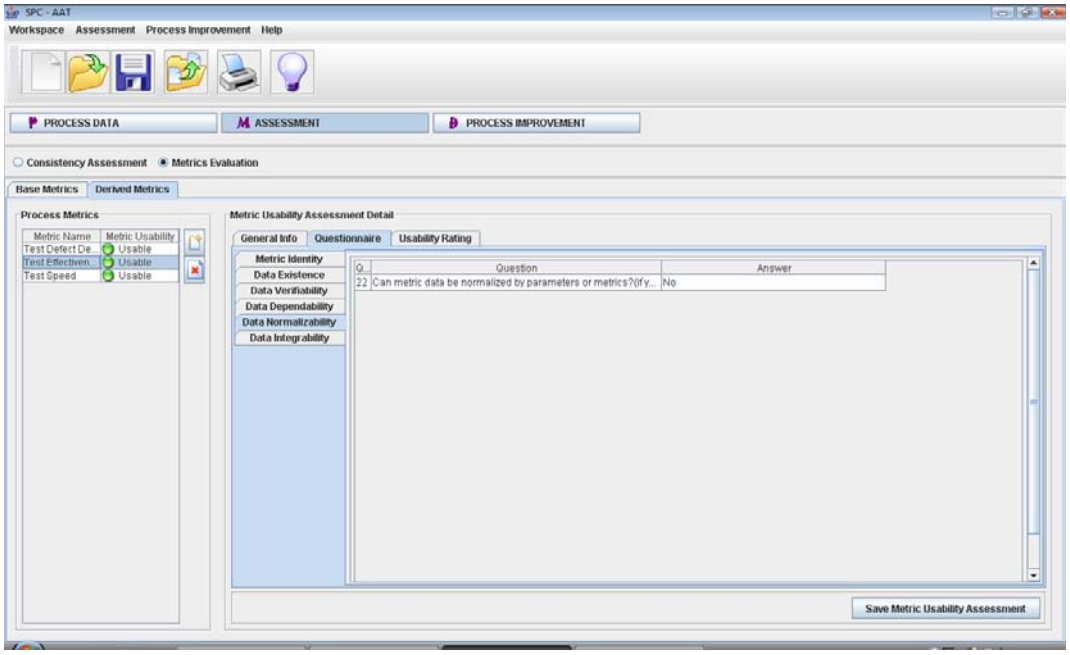


Figure 287 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

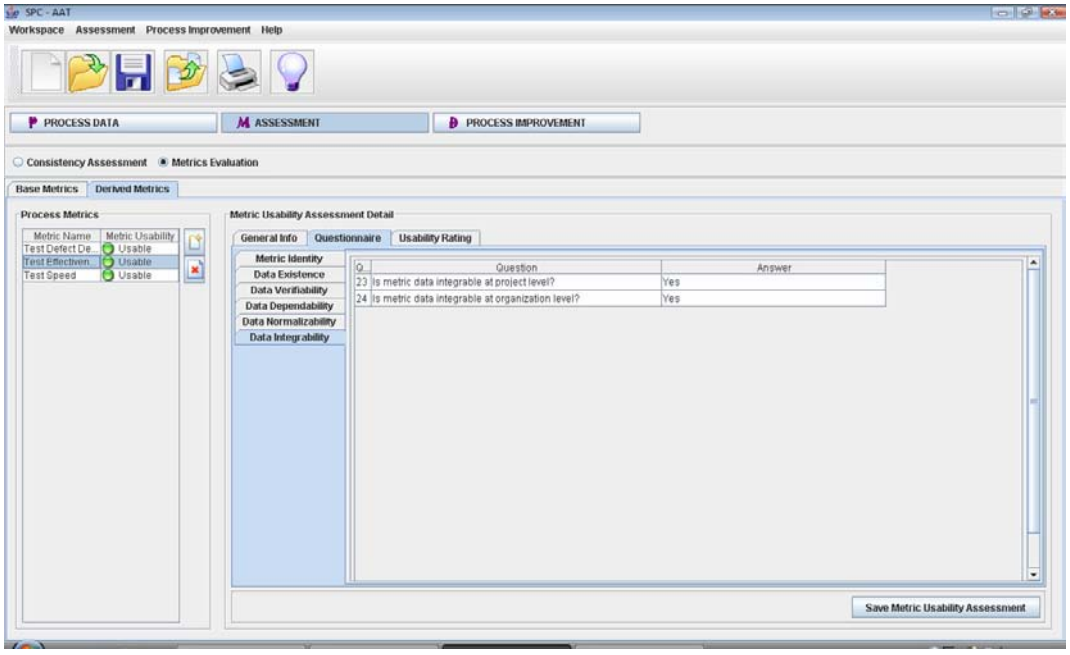


Figure 288 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

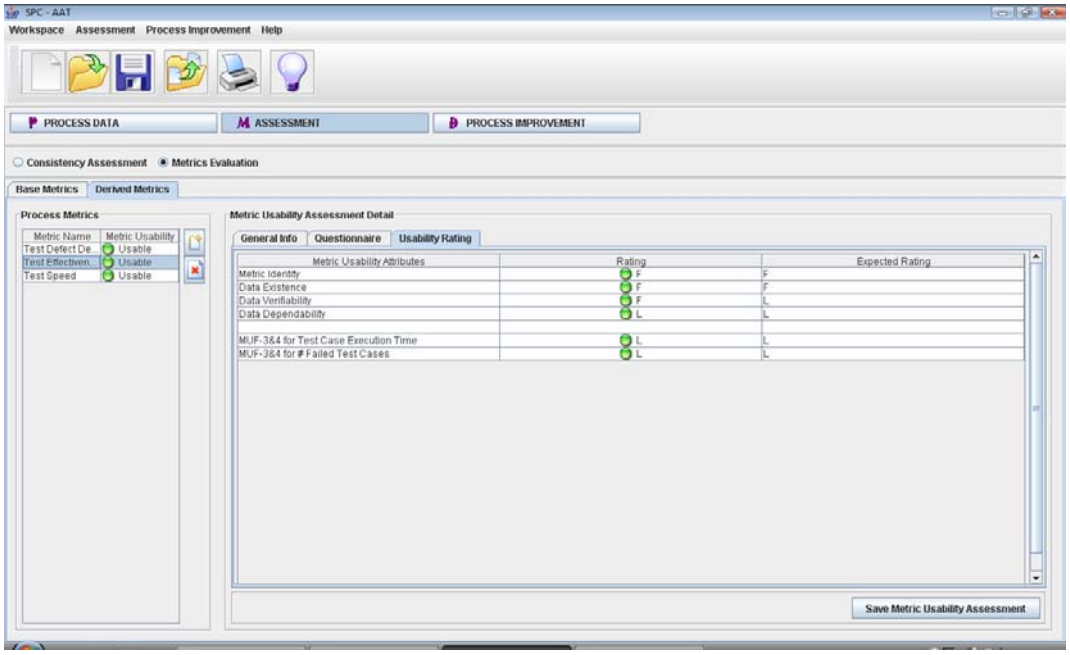


Figure 289 Metric Usability Questionnaire of “Test Effectiveness” Derived Metric

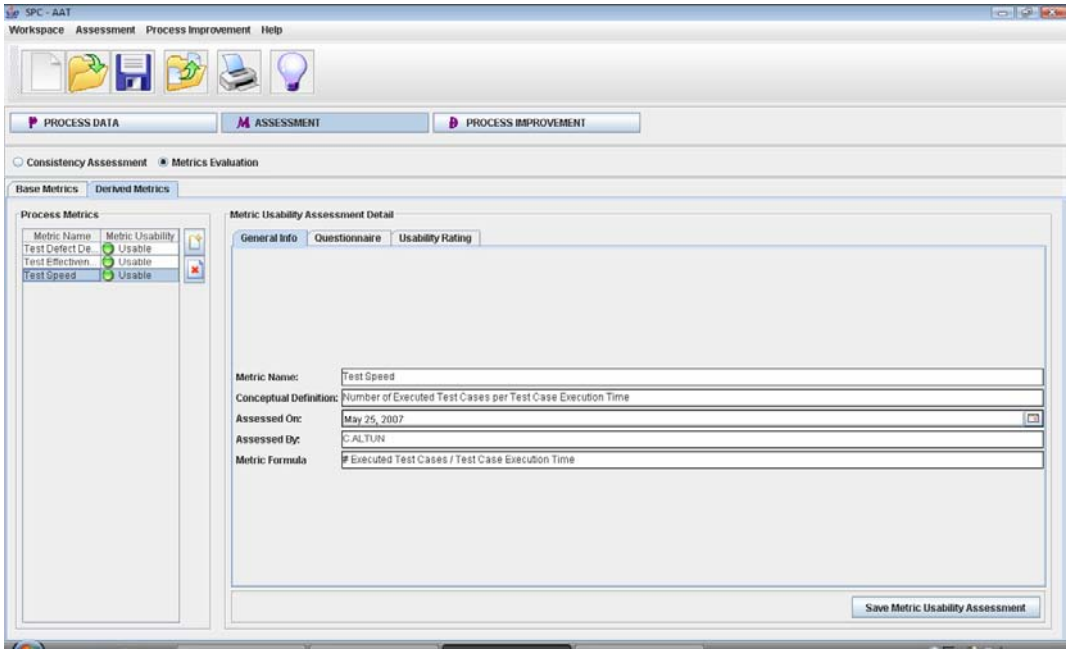


Figure 290 Metric Usability Questionnaire of “Test Speed” Derived Metric

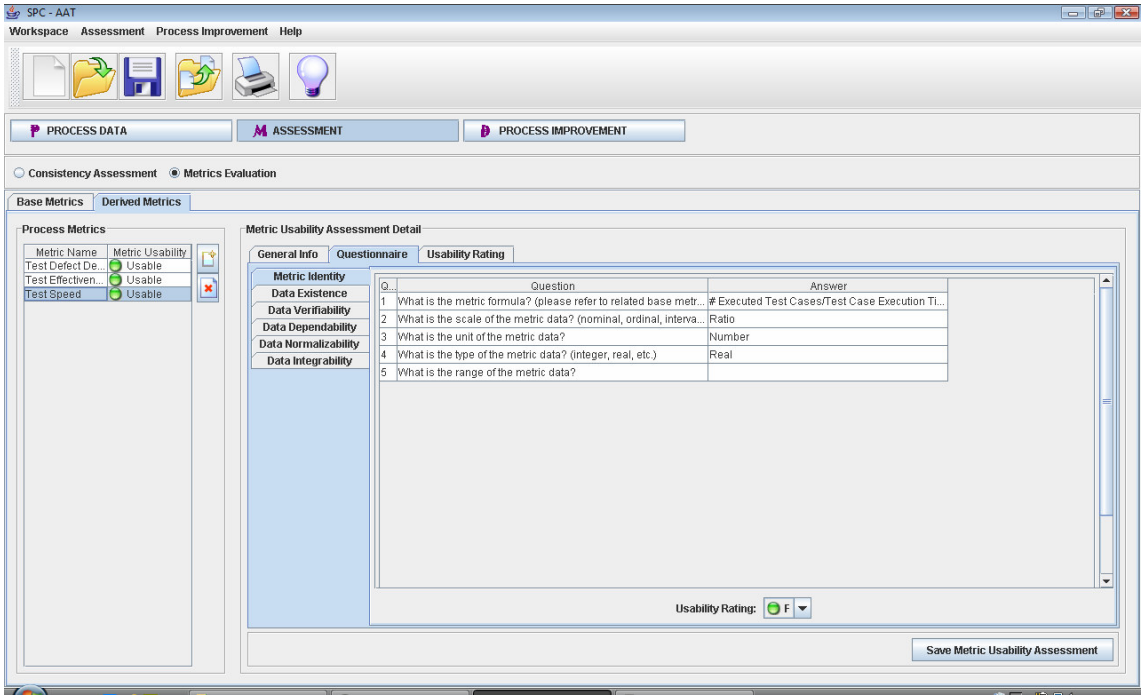


Figure 291 Metric Usability Questionnaire of “Test Speed” Derived Metric

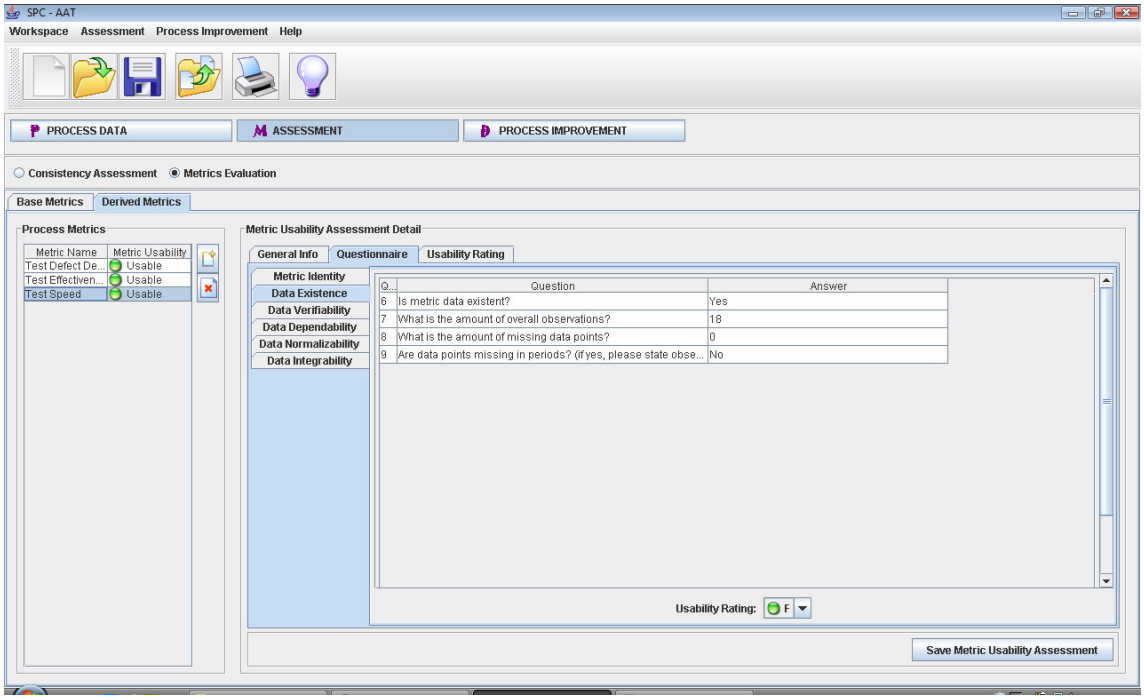


Figure 292 Metric Usability Questionnaire of “Test Speed” Derived Metric

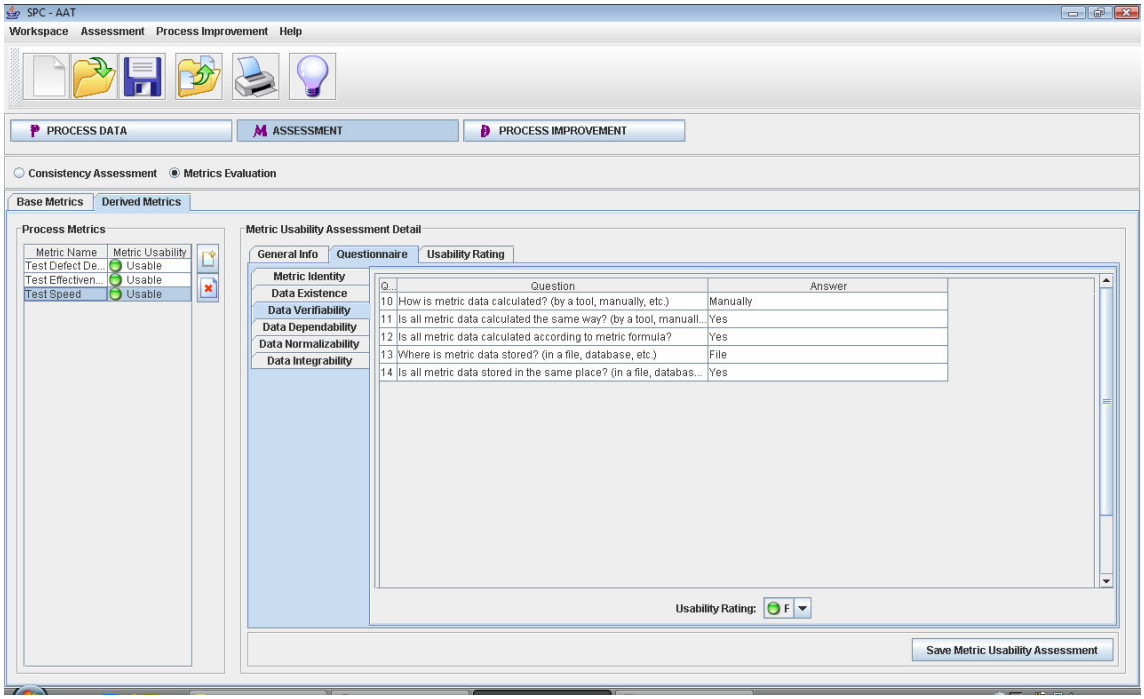


Figure 293 Metric Usability Questionnaire of “Test Speed” Derived Metric

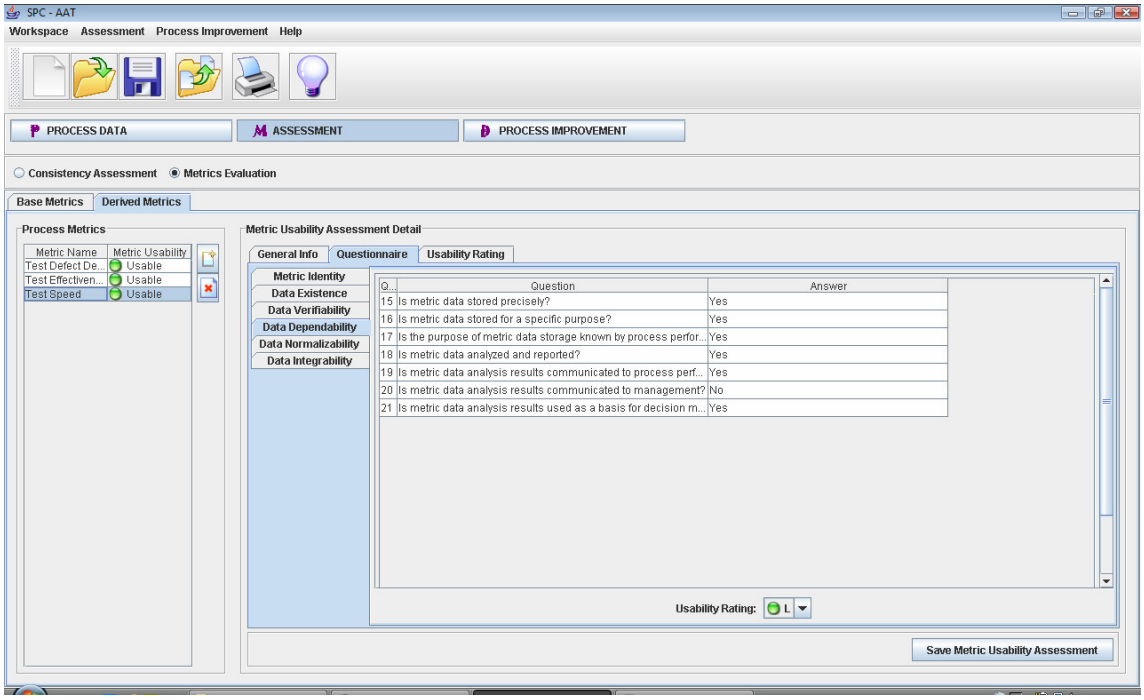


Figure 294 Metric Usability Questionnaire of “Test Speed” Derived Metric

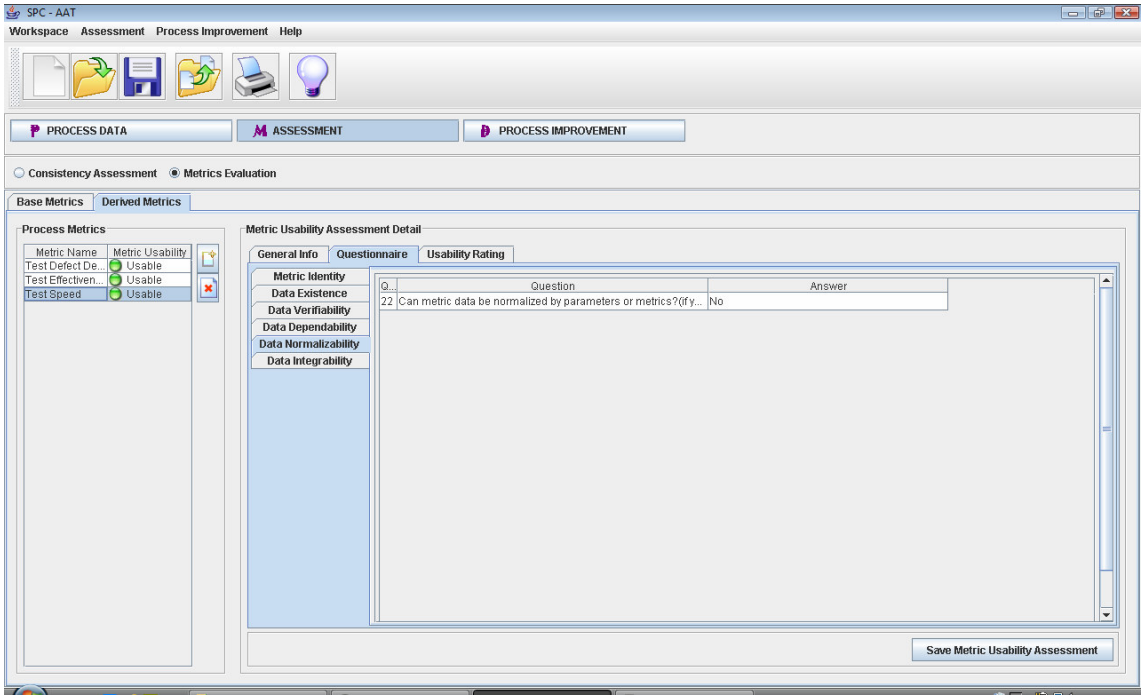


Figure 295 Metric Usability Questionnaire of “Test Speed” Derived Metric

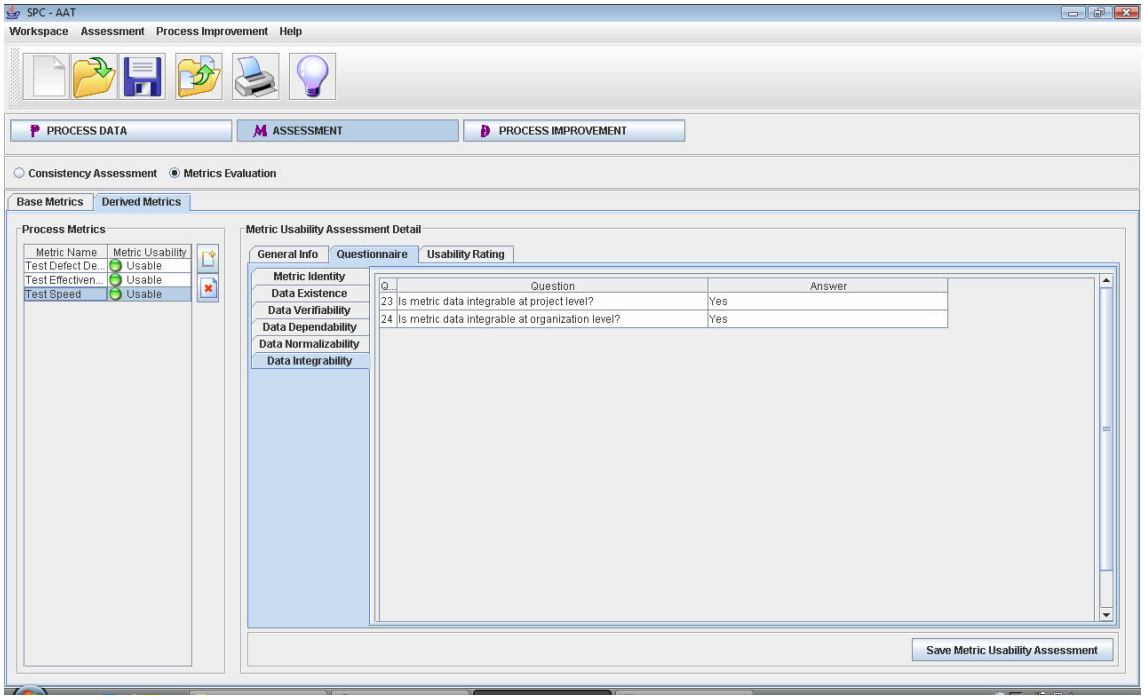


Figure 296 Metric Usability Questionnaire of “Test Speed” Derived Metric

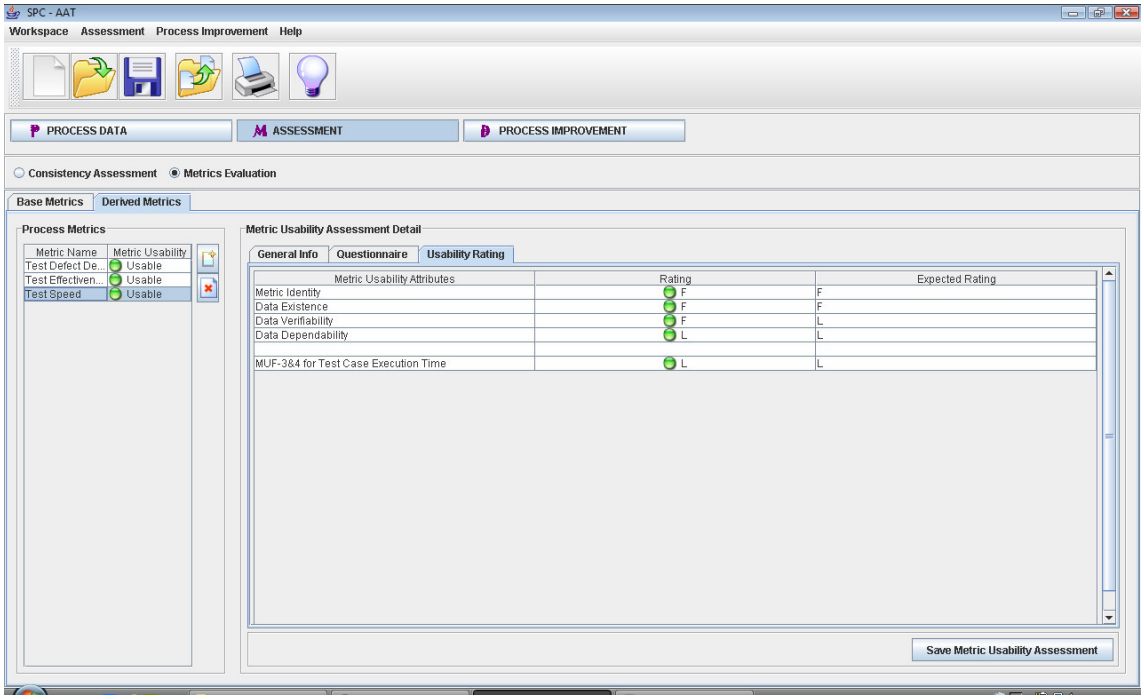


Figure 297 Metric Usability Questionnaire of “Test Speed” Derived Metric