Process mining applied to the change control board process

discovering real processes in software development process

Urena Hinojosa, E.

Award date:
2008

Eindhoven, February 2008

Process Mining Applied to the Change Control Board Process

Discovering Real Processes in Software Development Process

by:

Edward Ureña Hinojosa

Student identity number 0602310

in partial fulfillment of the requirements for the degree of

Master of Science
in Operations Management and Logistics

Supervisors:
dr.ir. J.J.M. Trienekens, TU/e, TM/IS
dr. A.J.M.M. Weijters, TU/e, TM/IS

TUE Department Technology Management.
Series Master Theses Operations Management and Logistics

Subject headings: data analysis, workflow, software development, modeling

Preface

This document is the final assignment to gain my master in Operations Management and Logistics (OML) at the Eindhoven University of Technology. The content of this document deals with the application of Process Mining techniques in the Change Control Board of a Software Development Project in the Netherlands.

During the evolvement of this master project I had the opportunity to apply theoretical notions of Process Mining in real data. This work has been a real gain of experience that reinforced my knowledge about the content of this master program, especially in the area of Business Process Management, Data Analysis and Process Mining.

I would like to thank all the people that has support me during this final assignment, especially Paul Siemons, Jos Trienekens, Ton Weijters, and Rob Kusters for their interest in the research of the project. My gratitude goes to Jesse Schobben, and Maria Dolores Lara for helping me with code of the program used in this project. I would like also to thank the people from the Information Systems of Technology Management department for helping and supporting my work during this time. Last but not least, I would like to thank to my family and friends for all the continuous support.

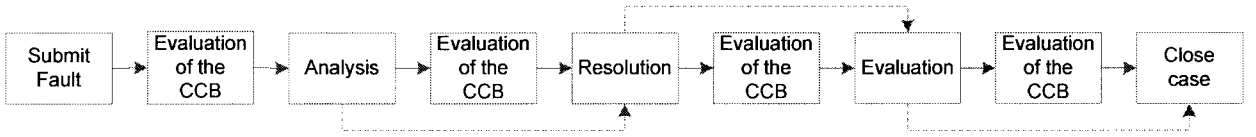
Executive Summary

Research question

Software is a collection of computer programs, procedures and documentation which final goal is to provide the capability to perform tasks within a computer system. The process to create Software is called Software Development Process (SDP) which consists on an "ordered set of activities that, after being completed, result in a software product that is delivered to a customer" (N. Whitten, 1995). SDPs use a specific methodology called "Life Cycle Model" in order to carry out the steps of the development. The Life Cycle Model is composed on a series of phases that contains the product specification, the design of the architecture, the programming or coding, the integration and the testing of the product. This methodology is executed by entities within the organization, each one in charge of executing specific activities within the SDP. A Software product is split in many Software objects and grouped in Configuration Items (CIs) in order to facilitate their handling. The entity in charge of managing CIs is the Software Configuration Management (SCM) entity. The SCM is divided in two sub-entities: the Library Management and the Change Control Board (CCB). The first is in charge of documenting information related to the CIs during the life cycle of the project. The second is in charge of handling changes requested by testers concerning improvements to the CIs that compose the Software product. This process of handling is called "CCB process".

The theoretical understanding of the CCB process is presented as a stiff linear sequence of tasks that handle all the CI's defects submitted to the CCB, like the structure of process model shown in the following figure.

Sequence of tasks of the CCB process – Theoretical Process Model



However, each CI's defect is different; it is to say that each CI's defect can have different levels of priority, severity or it might require different handling actions to be taken. Therefore, it is argued that the structure of the CCB process changes in order to overcome the specific characteristics of the CI's defects submitted. At the same time, because of the complexity of solving a CI's defect, the throughput time might change as well. Thus, the present thesis project has formulated the following research questions:

1. What is the real structure of the CCB process shaped by the CI's defects and their characteristics?
2. What are the CI's defect's characteristics that influence the behaviors within the CCB process?

Two additional questions try to investigate in-deep the influence of the CI's characteristics:

3. What is the difference of structure and performance between different CCB processes shaped by different sets of CI's defects with different characteristics?
4. Is it possible to infer rules and patterns in the CI's defects according to the way the CCB process behaves?

In order to answer these questions, the project will be analyze a data set coming from the CCB process of a middleware software project. The data set is composed by 10 snapshots taken in a monthly basis to the database where all the information about the handling of CI's defects in the CCB process has been recorded. The techniques selected for the analysis is Process Mining that consist on "methods for

distilling a structured process description from a set of real executions" (Aalst et al, 2003). Thus, based on these research questions and the type of analysis proposed the objective of the present report is:

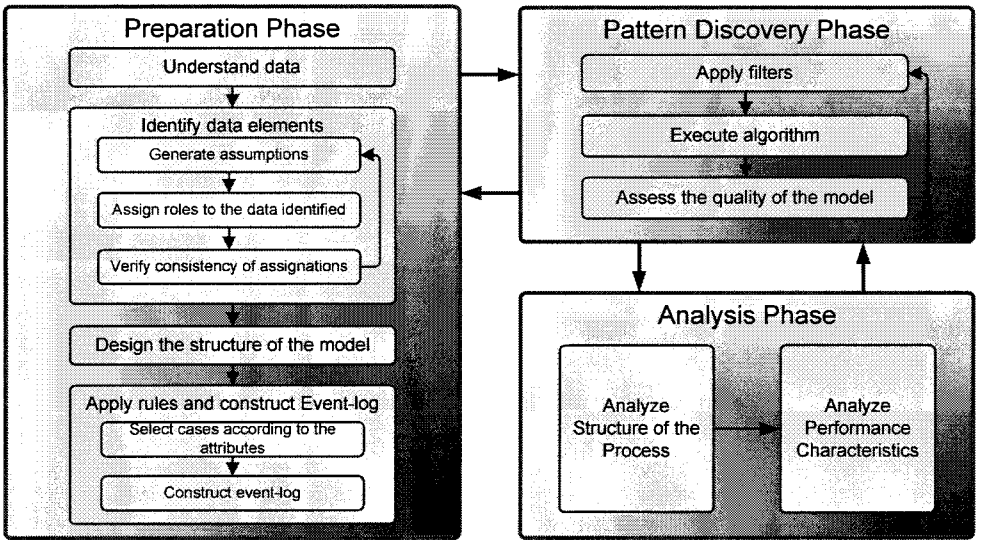
"Carry out the discovery of the real process structure and behavior of the Change Control Board process by understanding the Software Development Process and applying Process Mining techniques in a data set of CI's defects handled by the Change Control Board "

Method

The method proposed for the present project thesis is based on a sequence of steps extracted from the literature, in this project these steps shape what we called Process Mining framework. It consists of three main phases: (1) Preparation phase, (2) Pattern Discovery phase, and (3) Analysis phase. This framework is used in an experiment that will be applied in the data set. The Process Mining framework can be seen in the following picture.

Methodology of the experiments applying the Process Mining Framework

Based on:[13], [14], [15].



During the Preparation phase the information contained in the data set is analyzed and an event-log is produced. In this phase the 60 to 80% of the work from the total work in the experiment is carried out. The main steps in this phase are: (1) understand the data set and it elements, (2) identification of the elements that will be used by the Process Mining techniques, (3) Design the structure of how the process model will be built in the event-log, and (4) built the event-log. For this phase a small program was used for creating the traces of each of the CI's defect found in the data set. The outcome of this program is an event-log that will be used in the next phases of the experiment.

The Pattern Discovery phase is in charge of generating process models by using the event-log generated in the Preparation phase. This phase is carried in automatically by using ProM tool which is a Process Mining tool developed by the Eindhoven University of Technology. The main steps of this phase consist of: (1) application of filters to the event-log in order to select groups of CI's defects with specific values in their attributes, (2) it applies the Heuristics algorithm in order to generate process models, and (3) an assessment of the quality of the process models generated.

The Analysis phase is carried out from two perspectives that answer the research questions formulated in the present project thesis: (1) the structure of the CCB process influenced by the attributes of the CI's defects, and (2) the performance characteristics of the CCB process affected by the attributes of the CI's defects.

During the experiment two attempts were carried out for the Preparation and Pattern Discovery phase. These two attempts are based on two criteria of the sources of information in order to generate the event-log. The first attempt used only the current status of each CI's defect as the unique information about the tasks that shape the process. The second attempt used additional data elements that record the handling of the CI's defects in each of the tasks that compose the CCB process.

Results

After generating the process models for the analysis it has been possible to establish that the theoretical process model represents only a part of the CI's defects handled in the CCB process. Moreover, a more complex structure has been found, which is sensitive to the attributes of the CI's defects that are submitted. It was found that the sequence "submit→analysis' is not the only path that is followed by the CI's defects during their handling, instead of that the path: "submit→resolution" happens in 70% of the cases. Additionally, rework has been detected in the tasks: "analysis", "resolution", and "evaluation". Analyzing the performance characteristics, it was found that the CCB process has an arrival time of CI's defects of 10.72 [cases/day]. When measuring the average waiting times that CI's defects have after a task completed the handling shows that waiting time after the task "evaluation" takes 65.96 days, which is in average ten times the other average waiting times found for the other tasks.

When grouping CI's defects according to the values of their attributes, there is evidence that different value of attributes of CI's defects cause that the CCB process handles them in different way. This behavior has been observed in all the attributes observed in the present project thesis, so either the structure or the performance of the CCB process change from one attribute value to other.

The results have confirmed the initial assumption that the CCB process is sensitive to the attribute values of the CI's defects that are handled. This analysis will benefit further research in the area, where information coming from this research project is useful in order to carry out more refined analysis of the behavior of not only the CCB process but the entire SDP.

Contents

Chapter 1

1 Introduction to the Research

Software is a collection of computer programs, procedures and documentation which final goal is to provide the capability to perform tasks within a computer system. The development and production of software is carried out in projects where teams of skilled people work together in order to deliver software components which will work according to the specifications defined for the software product.

The area where these projects evolve is called Software Development Process (SDP) defined as an ordered set of activities that, after being completed, result in a software product that is delivered to a customer [1]. These projects are knowledge-intensive work, it is to say, that while developing a software product, the most important asset is the knowledge applied in all the elements that compose the product. Robillard [2] states that Software development is "the progressive crystallization of knowledge into a language that can be read and executed by a computer". The source of the knowledge used during a software project comes from the people involved in the development of the product. Thus, in order to control the Software Development Process it is not only necessary to control the technical issues that appear when developing a piece of software, but also to control the way people perform the work within the process of development.

This fact has contributed to the creation of new techniques which are in charge of providing solutions to the control of Software Development Projects. The task of these techniques is the creation of a structured SDP environment, where there is a controlled design, program, and analysis of the components of a software product. These techniques are grouped in a new area of science called Software Engineering that takes into account the issues of software production. The activities that involve software engineering are: managing, costing, planning, modeling, analyzing, specifying, designing, implementing, testing, and maintaining [3].

The initial work of Software Engineering has studied these activities separately and timidly integrating them in the structured processes. However, with the emergence of the concept of maturity of the SDP, the interest to know how well organized the processes within the organizations are has taken place. The study of processes in SDP has shown that classic analysis of operations is not adequate because of the clear differences between producing hardware and producing software [4].

A novel area of research called Process Mining (PM) has developed techniques capable to discover the true structure of complex processes by using the data produced in these processes. Their advantage is that they can be applied to process that produce either physical or non-physical products, because these techniques are based on an analysis of the sequences of activities that compose the process, and not the type of product created in the process. These techniques take advantage that nowadays it is possible to store big amounts of data related to the execution of the activities within a process. They make the discovery by analyzing the instances that pass through different transformations executed by the activities in a process.

In this report it will be intended to apply Process Mining techniques to one of the most interesting processes within SDP: the Change Control Board (CCB) process. The CCB is the entity of the SDP in charge to control the evolution of the software components grouped in different levels of Configuration Items (CIs).

1.1 Research Questions and Objective of the Project

The way that the CIs are handled by the CCB is by the processing of the CI's defects, and because the heterogeneity of their characteristics, it is argued that the CCB process does not have a unique structure for handling all the different CI's defects. Moreover, it is proposed to discover the differences of handling the different types of CI's defects that enter to the CCB process and create rules for predict the behavior of the CCB when a new CI's defect with specific characteristics enters to the process.

Thus, the research questions for the present project are presented as follows:

5. What is the real structure of the CCB process shaped by the CI's defects and their characteristics?
6. What are the CI's defect's characteristics that influence the behaviors within the CCB process?

The first question helps us to see the possibility to discover the real CCB process used in the project. The second question allows to identify potential ways to classify the CI's defects and their influence in the CCB process. Using these two concepts two additional research questions are formulated in order to analyze the CCB process in-deep.

7. What is the difference of structure and performance between different CCB processes shaped by different sets of CI's defects with different characteristics?
8. Is it possible to infer rules and patterns in the CI's defects according to the way the CCB process behaves?

Based on these research questions the objective of the present report is presented:

"Carry out the discovery of the real process structure and behavior of the Change Control Board process by understanding the Software Development Process and applying Process Mining techniques in a data set of CI's defects handled by the Change Control Board "

1.2 Research Outline

In order to achieve this goal, it is studied the data from a middleware software project in the Netherlands, this project is called MTR-A. Two types of data sets are analyzed: (1) Change Control Board data, (2) data concerning the product and the project itself, that is called "Documentation data" for the project. These data sets have been extracted for a Classical Software Engineering analysis, so here it will be tried to adapt them in order to use Process Mining techniques.

In order to apply Process Mining it is proposed to make a series of experiments based on what we call the Process Mining framework. This framework is composed by steps extracted from the literature and organized in a sequence in order to carry out a process mining. It is composed by three main phases: (1) preparation phase, (2) discovery phase and (3) analysis phase.

In Figure 1 the objective of the project is graphically presented. It content five elements: (1) the project and all its information, (2) the data collected for the classical Software Engineering analysis, this data is composed by two types of data sets previously mentioned, because these three types of data are interrelated and many of their variables belong to more than one type of data, they are presented as only one set of data, (3) the data that can be used by Classical Software Engineering Techniques, (4) the data suitable for Data Mining techniques, and (5) the data suitable for Process Mining techniques.

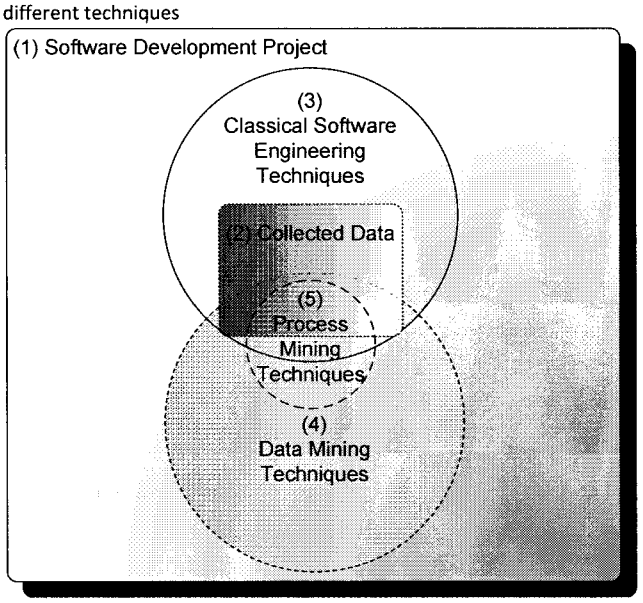
Process Mining cannot be directly applied in the data received. Thus, the thesis project will follow some preliminary steps that will help in the application of the Process Mining framework in the data from the project. Four steps are formulated for the present project:

1. Understanding of the environment where the CCB process is carried out.
2. Identification of suitable data for Process Mining techniques.

3. Formulation of an experiment that applies the Process Mining framework in the data and answers the research questions.
4. Execution of the experiments and analysis of the results.

1.3 Structure of the Report

Figure 1. The relationship between the data in the project and the different techniques



Based on the research outline presented in the previous section, the present thesis project is organized as follows: Chapter 2 will address a short description of environment of the SDP in the MTR-A project and the role of the CCB process, where a brief description of its components will be provided, and the limitations that present its current analysis. In Chapter 3, an analysis of the problems and the way in which Process Mining can help to overcome the limitations for a better understanding of the CCB process will be presented. Thereafter, Chapter 4 is dedicated to formulate the design of the experiments, in this chapter a brief description of the Process Mining Framework is made, and the way it will be used for the experiments. In next three chapters a description of the implementation of the experiments is presented. Chapter 5 and 6 describe the preparation and discovery phase of the two attempts carried out for these phases. Chapter 7 presents the analysis phase. At the end, in Chapter 8 the conclusion and further directions are given.

Chapter 2

2 The Change Control Board Process

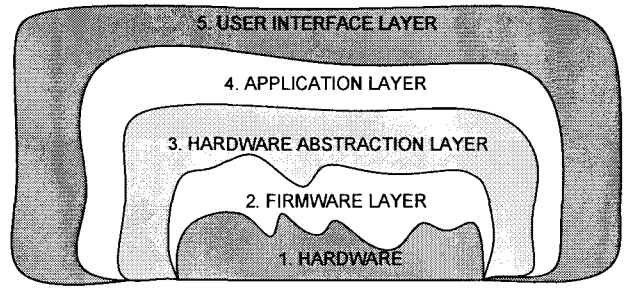
2.1 Introduction

The goal of the present project is to increase the accuracy of description of the Change Control Board (CCB) process in the MTR-A software development project. This chapter is dedicated to present the theoretical description of the project and the CCB process. This chapter is organized as follows: first a general description of the components of the MTR-A project is carried out. Thereafter, it will be introduced the CCB process describing its role in the MTR-A project and the way it is executed. Next, a brief description of the current analysis carried out to the MTR-A project and its CCB process will be presented. This chapter concludes with the limitations that these current analyses have if they consider the CCB process in an incorrect manner. Because of confidentiality reasons, here it is not authorized to use real names for the project nor for the product in this report.

2.2 The MTR-A Software Development Project

The MTR-A project is a Middleware Software Project of a company in the Netherlands. This project is focused on the development and production of software for an electronic device that will be released in the near future. The following description of the project is based on discussions with the consultant in charge of carrying out the analysis and evaluation of progress of the project.

Fig. 2.1 Classification of the layers of software in embedded products (MTR-A project)
Based on: Interview with the consultant



The software consists of multiple layers which are presented in figure 2.1. As a usual practice in this type of business, the hardware and the first layer above the hardware are developed by the hardware fabricant. In general, the hardware presents a diversity of features that have to be adapted by the software, that is why the irregular representation of the hardware in the figure. The layer known as Firmware improves the functionality of the hardware but it still requires improvements in order to make the device suitable for the market; thus, the objective of the next layers is to extend and correct the functionality of the device by filling in the gaps of software that controls the hardware.

The MTR-A project is in charge of developing these functionalities by developing the next the three upper layers (3, 4, and 5 layers). Layers 3 and 4 in Figure 2.1 are part of the normal MTR-A project. Layer 5, the user interface layer is developed in a different way for each family product. In addition, the User interface developed by the MTR-A project is meant for demonstration and testing the total content of the electronic device. It also can happen that the product is sold without this last layer, and the buying company will develop their own user interface for the electronic device.

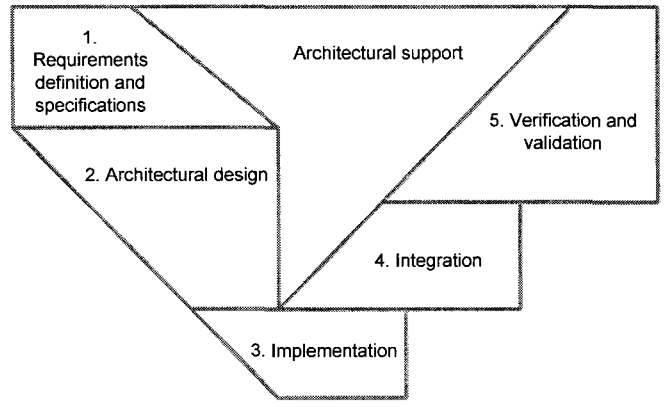
The processes used in order to develop these software layers are assessed according to the Capability Maturity Model (CMM) [5]. The latest assessment states that the organization is found in level 3 of the CMM. It means that the organization is capable to define the processes and their intermediate activities, where the inputs and outputs for each of these activities are known and understood. Due the possibility of identification of these elements, the intermediate activities can be examined, measured, and assessed. This

level of maturity allows the tracking of defects which can be used in order to plan the remaining phases of the project and also obtain prediction of the outcomes of the processes [3].

2.3 Life Cycle Model of the MTR-A Project

The Life Cycle Model used by the MTR-A project is based on the V-shape model. This model is characterized by a sequential course of execution processes. It is to say, that each phase must be finished before the next phase initiates. However, on the contrary to other sequential models like the Waterfall model, the V-shape model puts more emphasis on testing [6]. The V-shape model used in the MTRA project can be seen in Figure 2.2. The description of the phases of this model can be found below..

Figure 2.2. V-shape model used in the MTRA Project
Based on: Interview with the Consultant



1. The first phase included the development of requirements definitions and product specifications.
2. The second phase involves the architectural design of the product; this phase is started simultaneously with the first phase; however, it starts progressively according to what is defined in the first phase.
3. The implementation or coding of the software components is performed in phase 3. This phase starts in about the middle of progress of the second phase.
4. Phase 4 involves the integration of the components. This phase starts when the architectural design has been completed, and the coding phase is about the middle of progress. It is basically composed by integration tests of the components.
5. The fifth phase is the execution of verifications and validations about the functionality of the integrated components. This phase starts half way phase 4; additionally it is expected that some coding is carried out in case that the components' verification do not pass the assessments.

Additionally, there is a permanent link between the phases called Architectural support, which interrelates phases in order to provide information and support of one phase to the others.

It is important to notice, that the V-shape model shown in figure 2.2 differs from the V-shape life cycle model presented in the theory. Two main differences are found: (1) in the V-shape model of the MTR-A project, the phases might start simultaneously or in the middle of progress of the previous phase. (2) The "Architectural support" allows the link between any possible phases. This differs with the theoretical approach where the links relate only phases with the same kind of element worked, for instance "Design of components" are only linked to "Component test".

2.4 Complexity of the Process

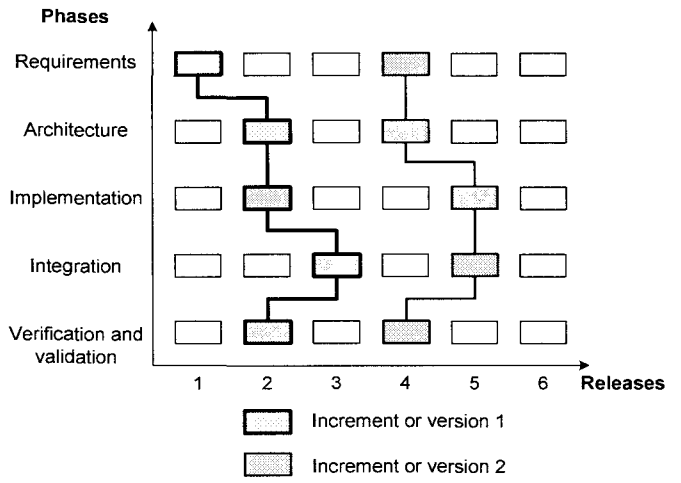
The SDP of the MTR-A project is also influenced by three elements that increase the complexity of the processes executed.

1. Releases, which are the updates made to the CIs within a single V-shape structure due the modifications required after performing tests in the phases of coding, integration, or verification and validation.
2. Increments or versions, which are the updates made to the whole product. The increments take different CI's releases in order to carry out the update of the whole product.

3. The MTR-A project develops a family of products, it is to say that there is a parallel creation of software products with different levels of functionality that depend on the use of the device that will be released to the market. Every product contains its own line of integration of the CIs depending on the functionality provided by the CI's releases.

The work in different component's releases, different increments or version and different family products is developed in parallel by the company. This situation creates a complex work environment, where new versions or increments start without having to wait that the previous version or increment is finished.

Figure 2.3.Increment formation based on the location of different CI's releases and the phases in the V-shape model.
Based on: [7].



Each new increment will use a combination of CI's releases, which may come from different phases of development in the V-shape structure. Figure 2.3 shows a schematic view of this behavior for only one family product. Each small box is a set of CIs found in a specific phase and release, which is integrated with other sets of CIs in order to assemble an increment of the product.

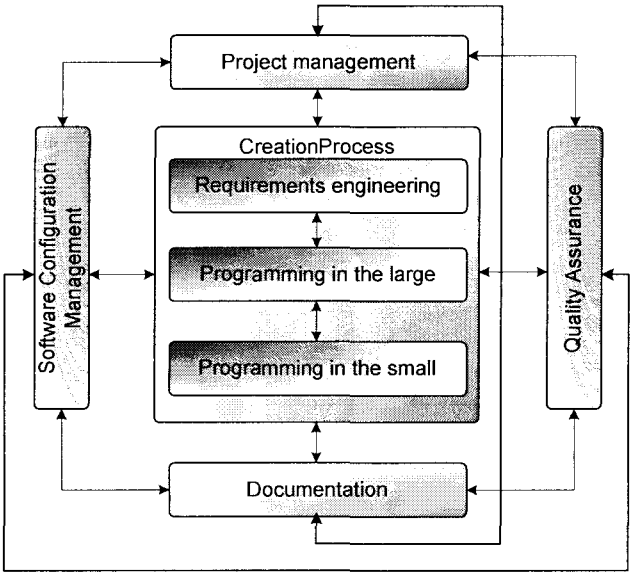
Other characteristic of the MTR-A project is that based on the initial Increment, called Basic Profile, the next generations of increments and family products are defined. Thus, the work on the MTR-A project allows the release of different family products, and incremental versions of each of them based on a basic initial profile of the product.

2.5 Core Life Cycle Model

In order to move the phases from the life cycle model to the entities that shape the organization, the approach made by Westfechtel and Conradi [7] is used. They defined in a simplified way the organizational structure of a Software organization that applies a Life Cycle model. This structure is called "Core Life Cycle Model" and it is composed by seven entities that execute a specific type of activity within the organization. These activities are related to the phases within the V-shaped structure of the life cycle model used in the MTR-A project.

In this model two main blocks are identified: the first block that we call "Creation Process" is composed by the entities focused on the creation of the software components. The second block is composed by the entities in charge to support the entities of the first block.

Fig 2.4. Core Life Cycle Model. Presenting the minimal structure of the interrelationships between components of a SDP
Based on:[7].



These entities provide the structure and organization for the whole project; we call this block of entities "Organizational Support". These entities and their competences are described below:

Creation Process block – Entities are focused on create the software product.

1. **Requirements engineering** that tackles the problem domain or "what has to be built".
2. **Programming-in-the-large** that comprises with the solution domain or "how the system is to be built" which includes the process of design of the architecture of the product.
3. **Programming-in-the-small** which writes the code according to the specifications made in the two previous entities.

Organizational Support block – These entities are focused in provide a structure and organization to the SDP of the organization.

4. **Project management**, which deals with the coordination issues such as planning, organizing, staffing, controlling, and leading projects.
5. **Software Configuration Management** dedicated to the management of all software objects including: definitions, software architecture models, component implementations, and test data. It is composed by the Library Management and the Change Control Board. They will be explained in-deep in the next section.
6. **Quality Assurance** which involve activities of validation, verification, inspection, and testing.
7. **Documentation** includes two types of documents: technical and user documentation.

Using this framework provides the possibility to understand the interconnections between the content of phases and entities within the organization, this helps to explain the flow of the elements in the project.

2.6 The Software Configuration Management

The Software Configuration Management (SCM) in the MTR-A project has as the main task the maintaining of a good organization of the elements that flow within the project. The definition given by the Software Engineering Institute (SEI) is the following: *"A discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements."* [8].

Basically the work of the SCM is the management of Configuration Items (CIs). The definition given to a CI by the SEI is: *"An aggregation of hardware, software, or both, that is designated for configuration items that make up a product."* [8].

A CI in the SCM of the MTR-A project is independently identifiable, stored, tested, reviewed, used, changed, delivered and/or maintained. In addition, the CIs within the MTR-A project are organized in a tree-like structure where a Top-level CI is composed by many Sub-Level CIs and these CIs can also be composed by a lower level of CIs.

The migration of a CI from one increment to the next in the SDP is based on the request of improvement for the features of the CIs. These requests for improvement will be called "CI's defects" in the present project. In order to organize the work of the CIs within the SCM of the MTR-A project, the SCM entity has divided the work over the CIs in two main activities carried out by two sub-entities: (1) the Library Management, and (2) the Change Control Board.

2.7 The Library Management

The Library Management (LM) is concerned with the following activities:

- Identify interrelation between elements of the software, and group them in CIs.
- Define a structure capable to store CIs preserving their interrelationships. This structure is called "Configuration library" or repository.
- Maintain the CI's status accounting, which consist on the record and reporting of information needed to managing a configuration effectively.

The LM has a strong link with the Change Control (CC) because it stores the changes of status of the CIs based on a list of CI's defects handled in the Change Control Board (CCB) process. The LM is the link between the SCM and the Documentation entity of the Core Life Cycle model in figure 2.4.

So far, in the present chapter it was presented the environment that surrounds the Change Control, this provides an idea of the complexity of the MTR-A project. The next section makes a description of the Change Control, the team in charge of executing its tasks (the Change Control Board), and the process that the CCB follows.

2.8 The Change Control and the Change Control Board

The Change Control (CC) is the element of the SCM in charge of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification [8]. As it was mentioned before in section 2.7, the CI's defects are the elements used by the CC in order to control the changes on the CI. The team in charge on performing these activities is called the Change Control Board.

The arc that connects the SCM and the entities of the Creation Process block in the Core Life Cycle model (figure 2.4) represents the work of the Change Control sub-entity over the Creation Process block.

The CC defines special attributes to the CI's defects, which affect the behavior of the CCB process. These attributes are part of the information of the CIs managed by the LM.

2.8.1 CCB Process

It is the set of activities used by the Change Control Board for handling requests of change (CIs' defects) for the CIs. The steps of this process are carried out by entities of the Creation Process block. The role of the CCB is to distribute tasks related to change the CIs in the Creation Process block, and later evaluate the outcomes of the processing made to the CI with respect to the change requested.

The CCB works with special attributes to the CI's defects that are submitted for a change. These attributes are the following:

- Priority – This is the assumed priority for correcting the CI. The evaluation depends on the need of the CI by the entities and the remaining components of the software product.
- Severity – This is the seriousness for a change in a CI given the importance of the CI in the structure of the product.
- Request type – It is the type of the change requested at the moment that the CI is submitted to the CCB, three possible types of requests are distinguished:
    - PR – Problem report or defect
    - CR – Change request
    - IR – Implementation request
- Caused during – The approximate V-structure's phase in which the defect was caused in the CI
- Discovered during – The phase and its activity in which the defect was found
- Team - The team in charge to carry out the work over the CI

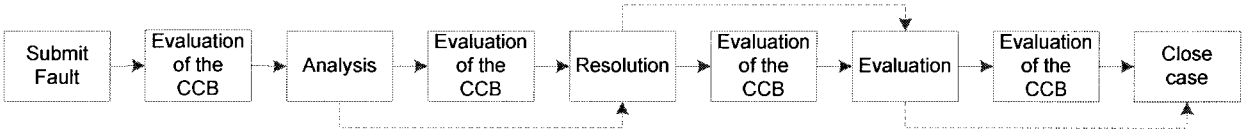
The analysis of these attributes can tell a lot about the SDP of the MTR-A project. For example it allows analyzing the evolution of the components of the software product (at the level of CIs), identify which CIs are crucial in the development of the project, the effort spent on the evolution of the CIs, and the identification of components that bring more difficulties for resolving them in the SDP.

Based on the explanation made by the consultant, the CCB process carries out as follows:

1. A tester finds a defect in a CI
2. The defect is submitted and it receives a CI's defect Id in the CCB system.
3. The CI's defect created goes to the CC Board, typically composed by a chairman and personnel involved in the project, depending on the case sometimes the client participates in these meetings. The team does a pre-selection of which "CI's defects" will be handled; this step is done in a weekly basis, but depending on the amount of "CI's defects" submitted so this activity is carried out in a more frequent basis.
4. The next step is an analysis of the defect in the CI. In this step an analyst is assigned in order to identify the solution for the request submitted about the CI. Once the analysis is finished depending on the importance of the CI's defect, two possible steps can happen: (1) the CI's defect goes again to the CCB for an evaluation of the results, before the next step starts. (2) The CI's defect directly follows the next step in the process.
5. The next step is the resolution; in this step a coder or programmer is assigned. Once this step is finished again two possible steps can be done like in the analysis step.
6. Thereafter, the CI's defect already resolved pass through a process of evaluation. Like in the two previous steps after the evaluation is done, the CI's defect can return back to the CCB for a judgment about the work performed and once accepted the result, the case is considered completed, or in case that the CI's defect is of minor importance, it is directly closed without a CCB board evaluation.

The way that the organization understands the CCB process is shown in figure 2.5 with the continue-line arcs. It is argued that it is a basic sequence of the activities where is not possible to see all the possible sequences that the handling of a CI's defect can follow. To complete the previous description, it were used dashed-line arcs that describe the other possible sequences of tasks handling a CI's defect in agreement with the previous description.

Figure 2.5. Sequence of tasks of the CCB process – Theoretical Process Model



This representation assumes a quasi linear of the execution of four tasks: CI's submitting, Analysis, Resolution, and Evaluation. The evaluation of the CI's defects by the CC Board is considered as a task that does not consume effort in the process, but it is assumed that between tasks is possible to execute this task. It is argued that this representation does not show the real structure of the process, it only shows what would be expected to happen in a completely successful execution of the tasks within the CCB process.

2.8.2 The Current Analysis Made Using the CCB Process and its Data

The analyses made by the consultant bring three types of information about the project:

1. The productivity and performance of the entities of the organization that are carrying out the project.

In order to analyze the CCB the consultant uses the process model shown in figure 2.5 as the base of understanding of the CCB process, and calculates the number of CI's defects in each of tasks of the CCB process.

2. An analysis of the evolution of the size of the components.

The consultant uses the Effort spend in the handling of the CI's defects as the base for these calculations.

3. Trends about the severity and the status of the defects in the project in a global basis.

These analyses are made by quantifying the amount of CI's defects that enter to the CCB process according to the time. Then they are represented in histograms that show trends about amount of CI's defects with different levels of severity that are being handled in the CCB process.

The methodologies used for these analyses are based on classic Software Engineering theory, which applies statistics techniques that explain the behavior of the product and its components at different levels, including components of the architecture of the product, CI's defects, and individual files of the product.

2.8.3 Limitations of the Current Understanding of the Process in the Analysis

It is argued that the current understanding of the CCB process limits the way to carry out the further analysis of the MTR-A project. Two perspectives were found in which the current analysis of the CCB process contains limitations: (1) the structure of the process influenced by the CI's defects and their attributes, and (2) the performance of the process affected by the CI's defects and their attributes.

- **Structure of the process**

The current analysis made to the CCB process takes into account a stiff linear structure of the process, which does not change due the influence of the CI's defects that enter to the process. It is to say that it is assumed that the process is invariable and it is not being influenced by the attributes of the defect submitted. It is not known if the CCB process executes an activity more than one time, or if there are activities that are skipped.

Is it possible to have an execution of activities in parallel? For example a CI's defect easily fixed could have been resolved and evaluated at the same time, or a CI's defect that after passing through an analysis is identified as a duplicated defect, and therefore directly closed after only having used the analysis activity of the process. Would it be possible to find CI's defects that after being incorrectly analyzed started their resolution with unsuccessful results and being forced to return again to the "analysis" task for a new analysis? How often does this situation happen? Is this situation affecting the assessment of the performance of the CCB process? Is there a possibility to infer rules according to these patterns?

It is argued that the attributes of the CI's defects submitted to the CCB process influence the structure of the process. An accurate analysis of the process can provide information of which type of defects requires more complex structures of the CCB process.

- **Performance of the activities within the process**

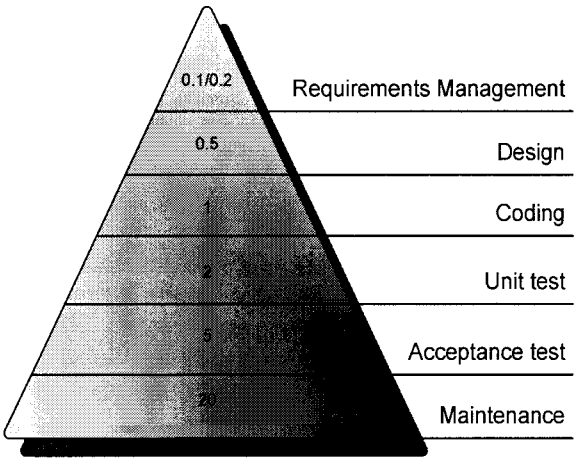
The analysis of performance of the CCB process has been neglected with the idea that it is very difficult to measure the creative process that is required at the moment to perform the activities within the process. As it was mentioned before the activities involved in the process creation of software are mainly carried

out by people and as Brooks in "The mythical man-month" [4] states: "software systems are perhaps the most intricate and complex man's handiworks". Thus, coming up with a creative solution for solving the problems submitted to the CCB may have a very irregular timing. However, it is argued that there must be an influence of the CI's attributes at the moment to come up with a solution. An easy problem submitted could not take the same time for resolution that a complicated problem when the problems are being analyzed by skilled people with large experience in the solution of this type of problems.

The current analyses carried out to the CCB process has been focused on trends about the severity of the CI's defects submitted to the CCB based on the historical description of the evolution of the CI's defects. These analyses are made in a global perspective, and they only consider the average lead times in the CCB process. It is argued that there are easy to solve defects but there are also defects difficult to solve, and an average lead time cannot describe the difference of handling the differences between defects. How much time does it take the resolution of a CI's defect with high level of severity? Are the teams performing similar when they are handling similar IC's defects? Are there activities more occupied than others within the CCB process?

In the theory of Software Engineering, it is assumed that the relative costs to repair a defect within SDPs increases according to the phase in which these defects are found [9]. Figure 2.6 shows this theoretical assumption.

Fig. 2.6 Relative cost to repair a defect at different lifecycle phases Based on: [9].



Because the phases of developing software in a life cycle model are carried out in a time line stream. It is possible that an analysis of the evolution of the CI's defects in a correct process model can capture the effort spend on a CI's defect and relate it to the phase that it belongs. In this way it would be possible to validate the theoretical assumption.

In conclusion, an improved analysis of the process taking into account the characteristics of the CI's defects submitted can increase the knowledge about the SDP in the MTR-A project, providing with typical patterns in the structure and behavior of the CCB process, and providing information for defining rules within it.

Chapter 3

3 Analysis of the Problems and the Research Proposal

3.1 Introduction

In chapter 2 it was presented a description of the environment of the SDP of the MTR-A project with emphasis on the Change Control and the Change Control Board process. It described how the CCB process is understood by the organization and how it can affect the analyses that are currently carried out to for the MTR-A project.

This chapter explains the benefits of having the real model of the CCB process before carrying out analyses of the SDP of the MTR-A project. Here it is stressed the need to discover how the attributes of the CI's defects submitted to the CCB process influence its structure. This chapter is organized in the following way: first an analysis of the benefits of a correct understanding of the CCB process based on the findings of section 2.8.3. Next, a brief description of the data provided by the MTR-A is done. At the end of the chapter an introduction to Process Mining is made, describing the type of information that it can provide and the data requirements needed in order to execute this type of analysis.

3.2 Benefits of a Deeper Understanding of the Process

The benefits of having a deeper understanding of the CCB project are be captured in an increase of the capabilities of analysis of the process' details. As it was mentioned in section 2.8.3 the aim of this report is to increase the understanding of the process from two perspectives: (1) the structure of the CCB process influenced by the CI's defects and their attributes, and (2) the performance of the process affected by the CI's defects and their attributes.

From the point of view CCB process' structure, the benefits are based on an increase in the capability to identify different paths shaped by causal sequences of tasks or activities within the structure of the process. Moreover, if the analysis of performance of the CCB process takes into account a real structure influenced by the attributes of the CI's defects, it can provide accurate productivity calculations that correctly describe the tasks of the process.

The improvements to the current analyses carried out for the CCB process can only be achieved when it is possible to apply techniques that can understand the data generated in the process including the attributes of the CI's defects, and the way that they affect the evolution of handling the CI's defect within the process.

3.3 The Data Available

The data analyzed in the present project is coming from two different sources:

1. Software Configuration Management → Change Control Board data
2. Documentation → Documentation data

They will be briefly described in the next sections.

3.3.1 Change Control Board Data

The CCB tracks and records the paths of each CI's defect from its entry until its exit of the CCB process. This information is stored in a database in coordination of the Library Management. The data is basically the status of the CI's defect. It is to say, the CI's defects' attributes, and dates when the tasks of the process were executed in the CCB process. This data set is presented in the form of snapshots taken in a

monthly basis to the database. A snapshot is a copy of the status of the content of the database as it was at a particular point in the time. It contains three interesting characteristics:

1. The data set is composed by snapshots taken in a weekly basis. This allows having records of the path of the life cycle of CIs' defects in the CCB process since they were introduced, passing through the activities of the process, until closing the case created for the CIs' defect.
2. It contains a list of attributes for each CI's defect submitted to the CCB process.
3. The snapshots show cases handled in previous dates to the date when the snapshot was taken, so it is still possible to have references of all the CIs' defects closed in the last snapshot.

An in-deep explanation of this data set will be presented in section 5.1.1.1.

3.3.2 Documentation Data

It is general information about the project that supports our understanding of the activities. Documentation data contains all the information concerning the product and the project, among others this data includes information about the characteristics of the product, components, specifications, quality policies and the planning of the activities within the project. In this sense, this data contains descriptions of the procedures carried out by the different departments or the organization in the MTR-A project. Two important sets of information are the Change Control Board work procedures, and the Quality system of the organization. Documentation data does not record data from the project, but it is the compass and map of the project because it contains all the necessary information for achieving the goal of the project.

It is argued that one of the most appropriate analysis techniques that can help to analyze the CCB process is Process Mining. In the following section a brief description of this technique is presented.

3.4 Process Mining

Process Mining is specific area of Data mining, focused on discover the characteristics of the process and the instances that pass through it. Aalst et al defined Process mining as "methods for distilling a structured process description from a set of real executions" [10]. The Gartner Group mentioned them as one of the new techniques that can help to produce a mapping of the processes within the industry [12]. In order to carry out a Process Mining analysis, the following elements are required:

1. The concept is defined as the hidden real process that has to be discovered.
2. The instances or "cases" that have entered to the process and execute a determined sequence of tasks before their processing is completed.
3. The attributes used by Process mining are basically three:
    a. The tasks that have been performed within the process
    b. The tasks' timestamps
    c. The resources that execute the tasks in a specific timestamp
4. Additional attributes in order to classify the instances.

These elements form of what is called "event-log" or "audit-trail".

One of the principal characteristics of this type of techniques is that the discovery of the process is done by using automatic tools that read the data and shape the structure of the process model. However, the preparation of the data used by the tool requires a manual handling.

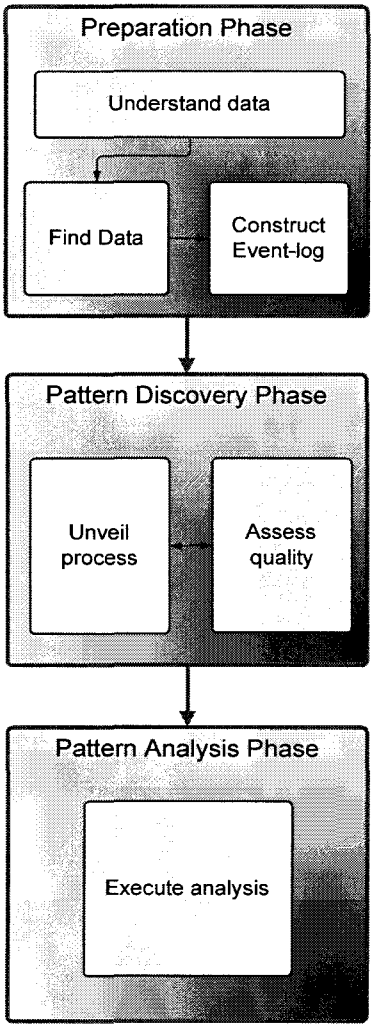
Not all the authors describe explicitly all the steps carried out during their experiments; thus, there is not a standard procedure of how to carry out Process Mining. Thus, based on the description provided by researchers in the literature it has been created a Process Mining framework which is shown in figure 3.1. It is composed by a sequence of steps used by different researchers for executing Process Mining. It contains three main phases which are described below.

3.4.1 Preparation Phase

The success of the application of these techniques depends on the effort spend on having a suitable data set which will feed the remaining phases of analysis. Researchers in the area of data mining estimate that this phase can take from 60% to 80% of the time of the whole process [16]. In the preparation phase an understanding and definition of suitable elements for the analysis is carried out. The preparation phase basically carries out the following activities:

- Understanding of the data
- Identification of the data elements suitable for the analysis
- Construction of an event-log composed by the instances and their process attributes, including these activities:
    - Cleaning of the data
    - Data transformation to an appropriate format for the analysis.

Fig. 3.1 Phases of the Process Mining Based on:[12], [13], [14], [15].



3.4.2 Pattern Discovery Phase

The function of the Pattern Discovery phase is, by means of applying algorithms or techniques on the data set, find patterns and rules that govern the instances contained in the data. In recent years there is an intensive development of algorithms in order to deal with the issues presented in the data that is analyzed. The issues are mainly referred to noise in the data set, and incompleteness of the event-logs which could not be eliminated during the preparation phase. Weijters et al [11] suggest that a way to tackle these problems is by applying of Heuristics approaches, which are based on the construction of dependency/frequency relations between the data.

3.4.3 Assessment of the Quality of the Process

Once the Discovery phase has been completed, a necessary step is to know how well the model created resembles the real process. Aalst et al [10] stated that an important criterion about the quality of a mined workflow model is the consistency between the mined model and the traces in the workflow log. They experimented in discovered models in order to identify factors that validate the quality of the model. The most important results obtained were the following:

- As more noise, there is less balance in the model which causes a negative effect on the quality of the result. In addition, they found that levels of noise higher than 10% make the model invalid.
- If high levels of noise increase, the ability of finding parallelism decreases.

These results help to validate a model in order to carry out the next phase of the analysis.

3.4.4 Analysis Phase

Once having a valid process, it is possible to start with the analysis phase of the model discovered. Here there is a combination of automatic steps carried out by analysis algorithms and manual calculations in case that automatic techniques are not available. However, the final results are made by the analyst that has an understanding of the process. In this phase three types of analysis are possible:

- Process perspective

This perspective studies the order of the activities within the organization or the department focused on the "control-flow" of the processes. "The goal of mining this perspective is to find a good characterization of all possible paths, e.g., expressed in terms of a Petri net" [17].

- Organizational perspective

It is focused on analyzing the flow of the activities from the point of view of the resources' exchanges in the process. The outcome of this analysis is the construction of a social network that shows the relations between individual performers, and classifies resources according to their roles, and organizational units.

- Case perspective

"Cases can also be characterized by the values of the corresponding data elements" [17].This perspective carries out an analysis of the cases according to their path in the process or the executors that worked on treating these cases. The main assumption of the case perspective analysis is the existence of various individual attributes linked to each case that influence the activities being executed, and the resources working on the case. Thus, by the use of clustering techniques it would be possible to find correlations between attributes.

Usually before starting with the Process Mining analysis, a decision is taken about what kind of analysis is going to be carried out, so it is possible to identify the most interesting data elements for the type of analysis that is going to be executed with the process discovered.

3.5 Conditions Required for Executing Process Mining

Each phase of the Process Mining framework requires specific elements. For the Preparation phase, it is required that the four types of data elements described in section 3.4 are present in the data analyzed. In addition, these data elements need to be reflected in a functional structure capable to describe a process composed by patterns of sequences of tasks. Patterns are defined "as an arrangement of elements, deeply dependent by the interactions among them, and by the intrinsic nature of each element" [18]. The work of constructing the functional structure from the data set is organized in a sequence of steps defined in a methodology based on the Process Mining framework. For the Pattern Discovery phase, it is required that the event-log created does not contain high levels of noise, so the algorithms used do not misinterpret the causal relationships between tasks. In addition, for the present thesis project, it is required that the attributes of the CI's defects can be manipulated by filters in order to produce different CCB process models shaped by CI's defects with specific attributes values. The Analysis phase requires that all the process models generated in the previous phase are correct and sound, and provide information about the structure and behavior of the process.

The requirements mentioned in this section will be used in order to plan each phase of the experiment that will be described in the next chapters of this document.

Chapter 4

4 Design of the Experiments

4.1 Introduction

In this chapter the design of the experiment that will help to carry out the Process Mining analysis is explained. In this sense the understanding of the CCB process made in chapter 2, and the explanation of the Process Mining framework described in chapter 3 are applied. During this chapter it will be defined the objective of the experiment and it will continue with the description of the method that enables the production of an event-log suitable for discovery and analysis phase.

4.2 Objective of the Experiment

The experiment aims to increase the understanding of the CCB process in two perspectives: (1) the structure of the CCB process influenced by the CI's defects and their attributes, and (2) the performance of the process affected by the CI's defects and their attributes. As it was mentioned before Process Mining has been selected as the technique that will be used in order to achieve this objective. The perspective of Process Mining in charge of making the analysis of the structure and performance of a process is the Process Perspective.

The Process perspective tries to answer "How" a case is handled within a process. It is focused on the discovery of the "control flow" of the process. It is to say that the Process perspective discovers the order in which the tasks are carried out (structure). In addition, because during the discovery of the process the algorithms quantify the number of executions of each of the activities in the cases submitted, this information enables the possibility of assessing the activities from the point of view of their performance.

For enabling the use of Process perspective algorithms the experiment has to integrate the attributes of the CI's defects with the information of the tasks that compose the process. In this way the results provided by the Process Mining techniques will reflect the influence of these attributes in the structure of the process and in its performance.

The objective of the experiment is to obtain information that will overcome the limitations presented in section 2.8.3. Therefore, the experiment needs to use the data elements found in the CCB data set, in such a way that it is possible to produce an event-log composed by process data elements influenced by the attributes of the CI's defects. The next sections have to deal with this objective.

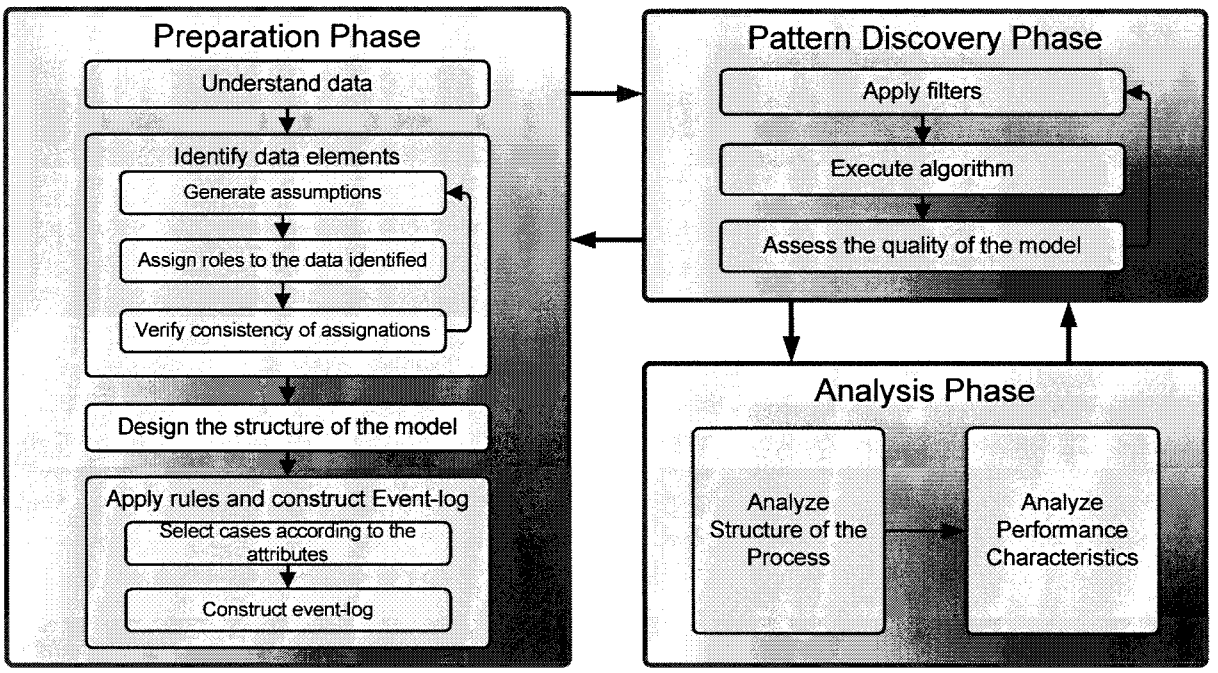
4.3 Methodology Used in the Experiments

In order to carry out the experiment two main elements are necessary to understand: (1) the CCB process and its data, and (2) Process Mining techniques.

The steps of the methodology are based on the Process Mining framework presented in figure 3.1 and a preliminary analysis of the CCB data set. Figure 4.1 presents an expanded Process Mining framework which includes the steps that will be used in the experiment.

It is argued that in practice Process Mining cannot be executed as a linear sequence of steps. There is interdependency between sequential phases. The Preparation phase needs to know what is needed in the Pattern discovery phase in order to prepare the event-log. The Pattern Discovery phase needs the content requirements of the model used in the Analysis phase. In addition, according to the requirements about the precision of the results of the whole process, a new cycle of the phases could be carried out having as a feedback the observations made to the results provided in the previous attempt.

Figure 4.1. Methodology of the experiments applying the Process Mining Framework
Based on:[12], [13], [14], [15].



The majority of the steps of the methodology are made within the Preparation phase. It is because this phase is the one with more manual steps, this phase involves the understanding of the process and the techniques used that can only be carried out by the analyst and not in an automatic way. Moreover, as it was mentioned before, between 60 to 80 % of the effort of the application of these techniques takes place in this phase.

The objective of the experiment is to provide sufficient information for an analysis of the influence of CI's defects' attributes in the process. A process model that contains only CI's defects with selected attributes can provide the information required for the analysis. Thus, it is argued that the experiment will need to generate different process models, each of them containing only CI's defects with only selected attributes. The following attributes are considered for the experiment:

- Priority
- Severity
- Type of request
- Teams
- Phase of the cycle model

In the following sections of this chapter the steps of the experiment will be explained.

4.3.1 Preparation Phase

The preparation phase is in charge to generate an adequate event-log file for the next phases of the Process Mining techniques. An event-log with suitable characteristics contains: (1) data elements for each process' component, and (2) data elements for each attribute required for the analysis. During the work in this phase, the quality of the data is one of the main problems. High levels of incompleteness and noise have to be tackled by using data cleaning techniques on the data set. The steps that include this phase in the experiments are the following:

4.3.1.1 *Understand Data*

In this step a preliminary inspection of the data is carried out. Two concepts are studied: (1) the global characteristics of the data set, and (2) the characteristics and behavior of the data elements. The first concept helps to study the data set from the point of view of its origin, size, and number of elements. The second concept helps to find roles to the data elements that compose the data set according to their characteristics and behavior.

4.3.1.2 *Identify Process Elements*

Here it is identified "what" represents each data element in relation to the process. The purpose of this step is to find the elements that shape the structure of the event-log used for the next phases of Process Mining. This step contains three tasks that need to be carried out:

1. Establish a set of assumptions that help us to identify the data elements. The assumptions are made in order to adjust the data elements to the CCB process.
2. Assign roles to the data elements identified according to the kind of information they contain in order to fill in the structure of the event-log.
3. Verify the consistency of the structure created with the data elements. In case that this verification is not satisfactory it will be necessary to come back to the first point in order to improve the fitness between the data set and the process.

In case that not all the elements that compose the process are present, the understanding of the CCB process and the assumptions made will be used in order to artificially generate the missing process elements. This is possible because it is assumed that the data elements that compose the even-log of a process are interrelated one to the other.

4.3.1.3 *Design a Hierarchical Structure between Process Elements*

The processes are composed by elements which are interdependent one to the other. So, in this step of the experiment an integration of the process elements is carried out. The idea is to design a hierarchical structure that will describe each of the tasks that compose the CCB process. This structure is the baseline needed in order to construct the event-log.

4.3.1.4 *Apply Rules and Construct the Event-Log*

The final step of this phase is the application of the rules created in the previous step to the data set. Because it is necessary to apply these rules to all the instances one by one, the previous steps are used in order to create a program that will be in charge of automatically construct the event-log. The program will be written in PHP and it will use a database. This will allow the handling of big quantities of information in a more structured way which is one of the requirements for this type of work.

The program will be in charge of the following activities:

1. Read the data sets and allocate their data elements in the structure of a database.
2. Create the data elements that are missing in the event-log's structure
3. Create the structure of the event-log

4.3.2 Pattern Discovery Phase

This phase of Process Mining is focused on the generation of a process model by using the event-log created during the preparation phase. The difference with respect to the previous phase is that this phase is carried out in a more automatic manner; moreover, the complete phase is executed in a software tool called ProM (Process Mining tool). This is an experimental software tool developed in the Eindhoven University of Technology with the objective to facilitate the application of Process Mining techniques to event-logs coming from data that belongs to different Information Systems that support the concept of

process. During the execution of this phase three steps were identified in order to create a suitable process model for the analysis phase.

4.3.2.1 Apply Filters

In case that the event-log still contains inconsistencies that neglect a correct application of the algorithms, ProM has the ability to filter the content of the event-log introduced. Two situations require the use of filters:

1. In order to generate process models that contains only complete cases in order to avoid wrong interpretations and calculations.
2. Select the cases that contain only the attributes to be analyzed.

4.3.2.2 Execute Algorithm

Once the event-log has been corrected the next step is the execution of a Process Mining algorithm that will produce a suitable process model for the perspective that will be analyzed in the next phase.

Weijters et al [11] suggest that a way to tackle with problems related to noise and incomplete logs is possible by the application of Heuristics approaches, which are based on the construction of dependency/frequency relations between the data. Because it is expected to find both types of issues in the event-log the use of the "heuristic" algorithm seems appropriate for the experiment. The outcome of this step will be a process model that reflects the content of the event-log.

4.3.2.3 Assess the Quality of the Model

In order to validate the process model discovered, it is necessary to assess its quality before accepting it as the real representation of the CCB process. Two types of validation will be used:

1. Assess the consistency of the process model created with respect to the cases contained in the event-log. For that the conformance checking algorithm available in ProM tool will be used.
2. Assess the logical sequence of the tasks within the structure of the process model. It is to say, that the process model will be accepted only if it contains an appropriate structure based on the principles that describe the CCB process.

Only if both validations are passed, the process model will be considered as a real abstraction of the CCB process in the MTR-A project. The final outcome of this phase is a set of various process models with different structure or performance characteristics depending on the attributes of the CI's defects used during their creation.

4.3.3 Analysis Phase

The Analysis Phase of Process Mining is in charge to make a descriptive analysis of the real processes discovered in the previous phase. Here are briefly described the aspects taken into consideration in the analysis of the results.

- Analyze the structure of the process - Here it is expected that the analysis will show the differences of handling different types of CI's defects.
- Analyze Performance characteristics – It will assess the performance of the CCB process in each structure discovered. With this analysis it is expected the find different ranges of processing times for each process model analyzed, and in this way improve the description of the performance of the CCB process.

4.4 How the Experiment will be Implemented

The implementation of the experiments is carried out in two parts: the first is focused on the Preparation and Pattern Discovery phases, and the second part is dedicated to the Analysis phase. The idea is to carry out the Analysis phase only in valid process models. A valid process model as it was explained in section 4.3.2.3 fulfills two conditions: (1) consistency with respect to its event-log and (2) contains a logical structure of the process model.

Additionally, during the execution of the experiment it was found that the understanding of the data set leads to suggest different possibilities of how to carry out the rest of experiment. Thus, for the present project two attempts of the experiment will be carried out but only in the first two phases of the Process Mining framework. This will allow having an adequate result that describes the general characteristics of the CCB process. The first attempt considers the most basic information found in the data set about the process. The second attempt will try to increase the accuracy of the results by increasing the number of data elements from the data set. The two attempts for the Preparation and Pattern Discovery phases will be described in Chapter 5 and 6 respectively, and the Analysis phase of the process models will be described in Chapter 7.

Chapter 5

5 First Attempt of the Experiment

This chapter describes the first attempt for executing the experiment of the project. The aim of this attempt is construct a process model with the most simple and intuitive elements of the CCB data set.

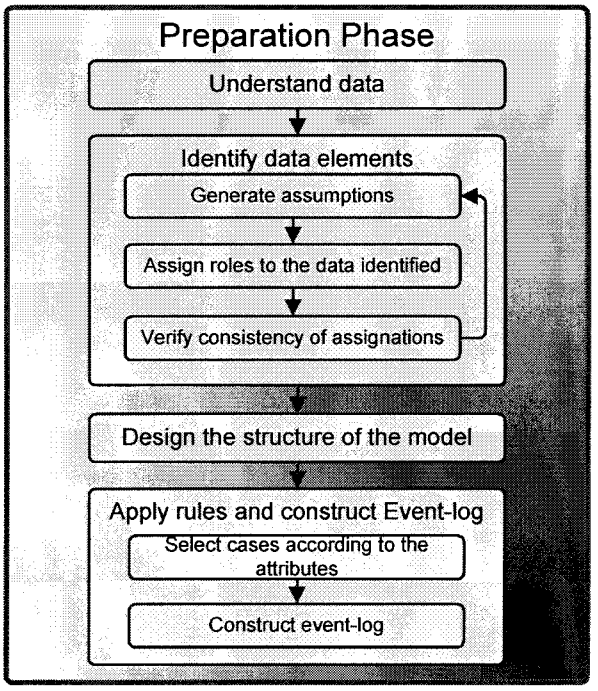
5.1 Preparation Phase

This phase is based on the structure presented in figure 5.1. It is divided in four steps: (1) the understanding of the data, (2) the identification of the data elements, (3) a design of the structure of the process model, and (4) the creation of the event-log.

5.1.1 Understanding the CCB Data

In order to identify the components of the event-log, the first step is to have an understanding of the CCB process data based on the way that the CCB works. Two elements are the background used: (1) the understanding of the environment where the CCB process evolves (chapter 2), and (2) a description of the procedures that govern the CCB, this information comes from the Documentation data available for the Study.

Fig. 5.1 Preparation phase and their steps
Based on:[12], [13], [14], [15].



The analysis of the data is made in two parts: (1) a global descriptions of the characteristics of the data, (2) the characteristics and behaviors of the data elements.

- **Global characteristics of the data set**

The CCB data is composed by snapshots taken from the database used by the CCB. These snapshots are taken in a weekly basis; however, for the present project only the snapshots taken at the end of the months for the period of one year are available. Each snapshot contains four types of data elements: (1) a general reference about the snapshot; (2) the CIs' defects introduced to the CCB process; (3) CI's defect's attributes; and (4) dates of execution of the activities for the CI's defect. Table 5.1 presents the data elements of a snapshot.

- **Characteristics and behaviors of the data elements**

Here a mapping of the data set that enables the manipulation of the data elements is created in order to produce the event-log. Each data element is of a specific type of measurement scale. The column "type" in table 5.1 gives the type of measurement scale of the data elements. The following types of data were found:

- Nominal data elements – they are used in order to give names to the attributes of the data elements.
- Ordinal data elements – they are used in order to establish a sequential order of appearance.

- Interval data elements – they are used in order to establish the order of appearance of the instances, but with the additional advantage that they allow arithmetic calculations.
- Absolute data elements – they have all the characteristics of the interval data elements with the additional advantage that all the possible calculations are meaningful.

In order to avoid ambiguity during the interpretation of the data elements, it is suggested that in the further research carried out in the same line of this project, these data elements will be analyzed one by one, and provide them an adequate name and more specific meaning.

Table 5.1. Elements of the Change Control Board data set
Based on: Internal documentation of the project

Element	Description	Type	Content
Snapshot reference			
Dataset	Name of the dataset	Ordinal	Project name + date of snapshot
History Date	Date this snapshot was taken	Interval	Date
System	System name	Nominal	String
Subsystem	Subsystem from which it was taken the snapshot	Nominal	String
CI's defect identification			
Problem number	unique ID assigned to the CI in the CCB	Ordinal	Integer
Product name	used differently by each project	Nominal	String (free text)
Product subsystem	Name of the subsystem where the IC's defect is stored	Nominal	String (free text) = Product name
Version	Version of the CI	Ordinal	Numeric =1 (always)
Release	CI's release label	Ordinal	String (free text) = Product name + Release Number
CI's defect attributes			
Priority	Priority given to the CI for its treatment in the CCB	Ordinal	String (with three levels)
Severity	Level of severity of the CI	Ordinal	String (with four levels)
Defect type	Description of the CI's defect	Nominal	String (free text) – not always available
Problem type	Depending the hierarchy of the CI, the CI can be parent or child	Ordinal	String (with two levels)
Request type	Identification of the CI's modification requested	Nominal	String - with three possible values
Caused during	when was the defect injected	Nominal	String – phase activities
Discovered during	when was the defect detected	Nominal	String – phase activities
Modifiable in	Name of the subsystem where the changes will be carried out	Nominal	String
Discovered on	The project	Nominal	String = MTR-A
Program	Program	Nominal	String – architectural components
Team	Team in charge of handling the CI	Nominal	String
Date of the activities executed for the CI's defect			
Current status	Current state of the CI's defect in the CCB process	Nominal	String
Create time	CI's creation of the record	Interval	Date
Submitted time	Date of submission of the CI to the CCB	Interval	Date
In analysis time	Date when the CI entered to the analysis process in the CCB	Interval	Date
Analyzed time	Date when the analysis ended	Interval	Date
In resolution time	Date when the CI entered to the resolution process in the CCB	Interval	Date
Resolved time	Date when the resolution ended	Interval	Date
In evaluation time	Date when the CI entered to the evaluation process in the CCB	Interval	Date
Evaluated time	Date when the evaluation ended	Interval	Date
Modify time	Latest change date of the CI's status in the CCI process	Interval	Date
Actual total effort	Estimated total effort spent on this CI	Absolute	Integer

In principle the type of data elements required by Process Mining techniques are nominal (case, task, and performer) and interval (time stamp). So the data set satisfy this requirement. All of the interval data elements are dates which have a strong link with the activities that are performed in the process.

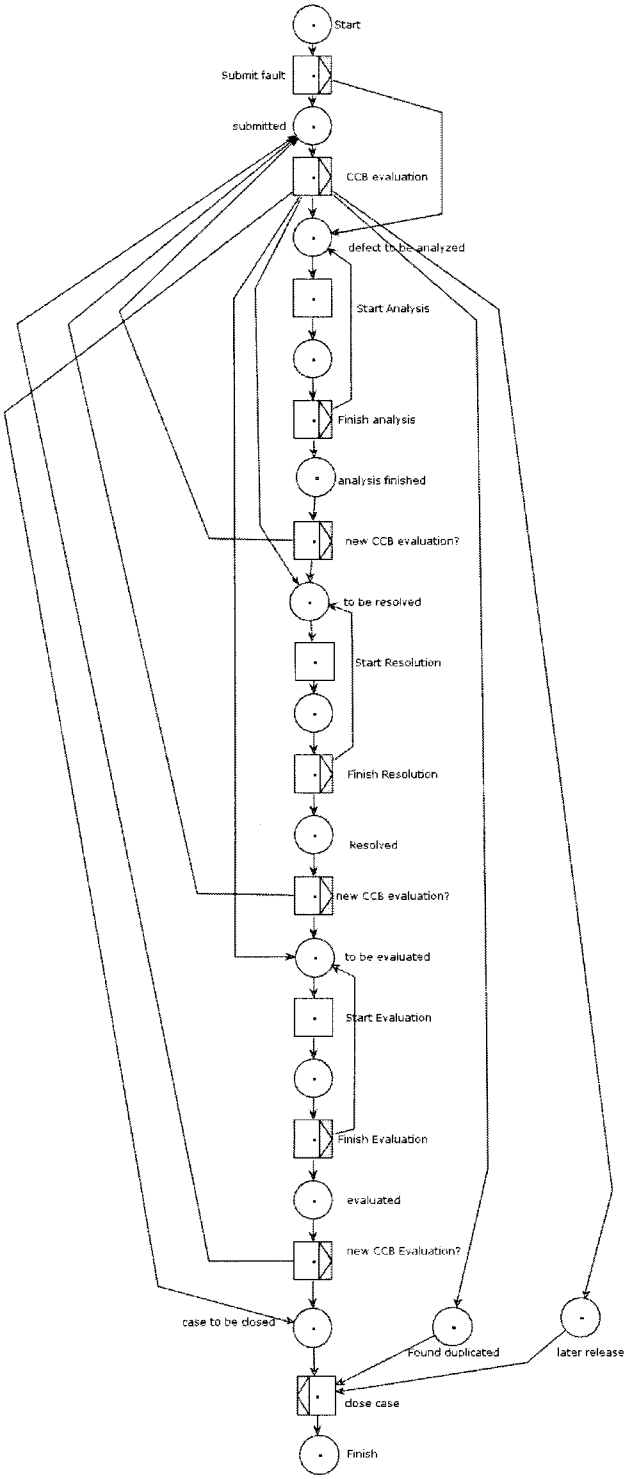
- **Behavior of the data set**

In every snapshot it is possible to find information about the elements that compose the process shown in figure 2.5 and its linear structure. However, after examining the sequences of CI's defects in all the snapshots, it was found that some instances contain activity's dates changing over the time. After asking about this observation to the consultant, it was found that only the successful executions of the activities are maintained in the database. Using this finding below it can be found an improved explanation of the CCB process taking into consideration the way that the data is recorded:

1. The CI' defect is detected and submitted with the same date, and then a "creation date" and "submitted date" are recorded. During this procedure the CI's defect receives an evaluation by the tester who submitted the CI's defect, this evaluation provides the attributes of the CI's defect; in addition, it establishes the importance of the CI's defect. If the CI's defect is important it will be evaluated by the CCB board (point 2); on the contrary, if the defect is not important the CI's defect will directly start with the "analysis" task (point 3).
2. The CCB board analyze the CI's defect and send it to the required task depending on the need (analysis, resolution, evaluation, or close case), with the following possibilities:
    a. The CI' defect is redirected to the "Close" if the defect is found: duplicated, expected to be fixed in next release or out of the scope of the functionality required.
    b. The CI's defect is redirected to tasks "Analysis", "Resolution", or "Evaluation" depending on the need.
3. The task assigned starts to handle the CI's defect, and then a date is introduced in the database (a date is introduced to the records in the field "in_..." of the task that will be executed). By defect, if the CI's defect is not important the task that starts the handling of the CI's defect is "Analysis".
4. When the task is completed, there are two possible outcomes:
    a. If the task's execution is successful, then a date is recorded in the field "...ed" that correspond to the activity, and the CI's defect is directed according to its importance to two possible tasks:
        i. An important CI's defect is reported to the CCB, and it waits to be redirected again to the next task, (it returns to point 2).
        ii. A not important CI's defect continues with the next logical task, for instance after Analysis it can be Resolution.
    b. If the task was not successfully executed, then the date in field "in_..." belonging to the task is removed, and depending on the importance of the defect, again there are two possible situations:
        i. An important CI's defect is returned to the CCB for a re-evaluation, (point 2).
        ii. A not important CI's defect is handled again by the same task (point 3)
5. Once all the tasks of the CCB process have been carried out, the case of the CI's defect is closed, and the data will not be changed anymore in the database.

The way of recording data in the MTR-A project affects the way that the CCB process has to be analyzed. An analysis of the CCB process that uses only one snapshot will bring as a result a process with a linear structure because only one snapshot does not take into consideration for example the possibilities of failing the execution of a task and having to execute it again. This part of the analysis of the data helps to reveal this behavior and it was used in order to make a preliminary conceptual model of the CCB process, this model is shown in figure 5.2.

Figure 5.2 – Conceptual Process model
Based on: Discussion with consultant and Internal report of Software Configuration Management



5.1.2 Identify Data Elements

The goal of this phase of the experiment design is the identification of the elements required by Process mining techniques. These elements are four:

1. The "case" that is handled in the process
2. The "tasks" that are executed when handling the "case" in the process
3. The "time stamp" of each "task" executed during the handling of a "case" in the process
4. The "resources" involved in the execution of each "task" during the handling of a "case" in the process.

As it can be appreciated in this list of elements there is a strong link between the data elements required for the analysis, it is known as "structured set of data" [16]. These types of data sets have the ability to describe common traces in the instances contained in the data set. These common traces are known as patterns that represent causal, hierarchical, and topological relations [18].

The structure of the content of the event-log is a composition of paths that handle "m" CI's defects.

$$Event\ \log = \{path_1, path_2, \ldots, path_{m-1}, path_m\}$$

In addition the structure of a path "j" is:

$$Path_j = \{case_j, [task_1 \ldots task_o], [attributte_1 \ldots attribute_p]\}$$

In addition, it is necessary to establish when a task has been either started or completed, and the resource that has executed it. In order to establish the status of execution of an activity here it is used the concept of event type. An event type tells the status of execution of a task in a specific time of execution. Then, the structure of a task "k" as follows:

$$Task_k = \{[event_1(time) \ldots event_q(time)], resources\}$$

Below it will be carried out an identification of these four elements one by one.

5.1.2.1 Assumptions for the First Attempt

For the first attempt of the experiment the following assumptions concerning the process were used:

- Each path created by the handling of a CI's defect in the process is composed a sequence of tasks.
- Because the snapshots follow the evolution of the handling of the CI's defect by the CCB; then, the snapshots should be able to tell when each of the activities that handle the CI's defect are started and when they have been completed.
- Because the status of a CI's defect given in the snapshots provide information about the moment when the CI's defect is being analyzed by the CCB' board, but it is not possible to establish which of the CI's defect are being handled by this board, it will be assumed that all of the CI's defects pass through this evaluation.
- The handling of a CI's defect starts with the activity "Submitted" and it will finish with the activity "Concluded"

5.1.2.2 Assign Roles to the Data Elements Identified

Based on these assumptions below is presented the identification of the data elements that will be introduced in the event-log.

- **The "case" in the CCB process**

Aalst and Weijters [19] define data element "case" in the following way: The "case" or "instance" is the "thing" being handled in the process. Clearly the CI's defects submitted to the CCB are the "thing" which is being handled. The elements that form part of the identification of the CI's defects in the data set are presented in table 5.1 in the section CI's defects attributes.

The "problem number" is meant as the unique identification used in the data sets for the CI's defects; however, after an inspection of the data sets, it was found that there are cases that contain the same value of "problem number" with the difference that they are captured from different subsystems. In order to have a unique identification for the cases used in the event-log, here it is proposed the creation of a new data element that merges the "problem number" and its "subsystem". This element IDP and it will be used as the unique identification for the cases handled in the process within the event-log.

In addition, it was found that the data elements: Product name, Product Subsystem, and Release contain related values among many CI's defects. So it is possible to classify CI's defects using to the following structure:

- Product name or Product Subsystem →CI
  - Release → Release of the CI

  Example: Product name 1
  Product name 2
  ...
  Product name n

This structure can be used in order to carry out analyses of the processes at the level of individual CIs.

The only element that is not bringing any actual use is "Version" because these elements remain constant with the value of 1 in all the instances of all the snapshots.

- **Identification of the "tasks" and "event types" used in the process**

A "task" is some operation on the case [19]. The tasks in the data set are exposed in two parts: (1) the "CrStatus" data element (Current status of the CI's defect within the process), and (2) the activity's dates used by the CI's defect during its handling, these data elements will be called in this project as "Processing data elements". In this first attempt only will be used the "CrStatus" as a descriptor of the activities executed for a case in the CCB process. The advantage of this approach is that it is easier to construct the event-log with it. The different activities that handle the CI's defect will be found one by one in the snapshots. The "CrStatus" data field indicates when a task has either been started or been completed; then besides indicating the task being executed, it is also capable to indicate the event type of the task. For the present analysis two event types are used: (1) "start" when the task is started, and (2) "complete", when the activity is considered finished. Table 5.2 presents the tasks or activities found in the snapshots.

The elements marked with "*" in table 5.1 represent activities related to the assessment and further rejection of a CI's defect's handled in the CCB process. In this project these tasks will be called "CCB evaluation tasks". All of these states are generated after a result of the previous task executed appears in the data set. For the present report the "CCB process analysis" will be considered a task because there is a period of time between completing tasks and starting new tasks where a possible value of the "CrStatus" provides information about the CCB evaluation.

Then, when these elements will be present in the model discovered it is necessary to understand that they represent CI's defects were assessed and later on rejected. In order to assign the type of event to the values of the "CrStatus", it is necessary to use the understanding of the meaning of each value:

- If the value of the "CrStatus" begins with "In_" it means that the task described after the text "In_" is in execution, and the first time that this value appears in the snapshot is when the task starts the execution, so the event is "Start" is created
- If the value of the "CrStatus" finishes with "ed" it means that the task that was in execution has been completed, so the event "Complete" is created.

- If the value of the "CrStatus" does contain values referred to the "CCB evaluation tasks", the event type created is "Complete" and belongs to the tasks that are described in its text.

These assignations are presented in table 5.2.

Table 5.2 List of possible CrStatus and their relationship with the tasks within the CCB process

CrStatus	Task identified	Event type
Submitted	Submit	Complete
In analysis	Analysis	Start
Analyzed		Complete
Analysis failed		
In resolution	Resolution	Start
Resolved		Complete
Resolution failed		
In evaluation	Evaluation (by an analyst)	Start
Evaluated		Complete
Concluded	Close	Complete
*Duplicate_	Work assessment	Complete
*Duplicate analyzed *Duplicate concluded *Duplicate evaluated *Duplicate in analysis *Duplicate in evaluation *Duplicate in resolution *Duplicate later release *Duplicate rejected	"CCB evaluation task"	Complete
* Later release	"CCB evaluation task"	Complete (not considered for further analysis)
* Not reproducible		
* On hold		
* Rejected		

- **Identification of the "time stamp" of the tasks in the process**

Events have a time stamp indicating the time of occurrence [19]. As it was explained before, there is a strong link between the tasks, their event types and time stamps in a process. In order to find the approximate date when a CI's defect has passed from one task to the next one, it will be used the value of the "History date" which is the time of when the snapshot was taken. This data element is directly linked to the task's event found in the "CrStatus" of the CI's defect. The drawback of using this data element as the time stamp for the task's events is the lack of accuracy. The next attempt will deal with this drawback by using other additional data elements that describe better the time of execution of the tasks.

- **Identification of the "resource" that execute the tasks**

The identification of the "resource" is probably the easiest task of this step. The resources are the executors of the tasks that handle the CI's defect. The data field "Team" directly provides us with this information. It was possible to detect that a CI's defect is sometimes handled by more than one team. This will be taken into consideration at the moment of construct the event-log.

Because the first attempt tries to discover the process model using the simplest elements, and avoiding the inclusion of the attributes, here will not be an identification of the attributes and their handling in this attempt. However, these activities are planned for the second attempt.

5.1.3 Design the Structure of the Process Model

Here is the summary of the findings of the previous two steps, providing an idea of how it will be extracted the data elements for the construction of the event-log.

The event-log will contain paths with the following structure:

$$Path_i = \{IDP_i, [event_1, \ldots event_q], Team\}$$

Where,

IDP = ID of the IC' defect = "Problem number"+"Subsystem"

$event_j$ = Task's events that have handled the CI's defect in a number ranging between i= 1.. q

A task's $event_j$ is found in the "CrStatus" as explained in the classification made in table 5.2. The structure of the tasks is the following:

$$event_j = \{CrStatus_k, History\ date\}$$

The way to see the structure of the process model created will be through the task's events, which they are expected to have an adequate sequence in the model discovered. The "CrStatus" data element is found only once in each snapshot, so the idea would be to create the sequence of task's events that have handled a case by creating a sequence of "CrStatus" and their corresponding "History date" data elements. The next step contains an explanation of how the program will extract the data from the data sets.

5.1.4 Apply Rules and Construct the Event-Log

Once it is known where to obtain the data elements for the event log, it is needed to apply the rules identified to all the CI's defects that are in the data set. The total number of CI's defects is 8714 in the last snapshot, so making a manual extraction of the data elements would be an extremely repetitive work. That is why it is propose to use a program capable to do it. A description of the content of this program is in the remaining of this section.

5.1.4.1 Implementation of the Program

The program is divided in the two parts:

1. Organization of the data elements within the snapshot

The program will use a database where it will store all the data elements. It's expected that with the help of the database it will be possible to carry out advanced manipulations of the data. Thus, it contains the following sub-activity:

a. Automatic identification of the data elements

During the process of storing the data the program will be in charge of identifying each data element and allocate it to the adequate data field of a table.

2. Organization of the outcomes of the snapshots in one unique path.

This part of the program will be in charge of the following sub-activities:

a. Automatics identification of valid paths
b. Automatic integration of the structures coming from different snapshots in one unique path
c. Automatic transformation of the paths to the structure of the XML file required.

Three problems were found and need to be overcome in this step of the program:

1. The snapshots contain the cumulative CI's defects submitted to the CCB, because of that the number of CI's submitted increases in each new snapshot of the series.
2. Even if it should not happen, there are CI's defects that have a temporal stay in the data set, it is to say that in certain moment they were submitted but after some time, they are removed from the data set. So they are present in the first snapshots, but not in the further snapshots.

3. There are CI's defects that do not change their status for long periods, so the same "CrStatus" remains the same in many snapshots.

The program will read the snapshots one by one and store the data elements in a unique table created with the identical content of the snapshots, but with the capability to have all the data elements of the snapshots in it. This will solve the problem of the incremental number of cases and temporal cases in the snapshots. The solution for the third problem will be carried out at the moment of constructing the XML file; this part of the program will use a query that will compare values of the "CrStatus" for the same case. The idea used in the program can be seen as an algorithm with the following commands:

a. Read data from snapshot
b. Evaluate IDP

If IDP_i exists in the XML
then
if $CrStatus_i = CrStatus_{i-1}$ and History $date_i$ = History $date_{i-1}$
then
do nothing
else
add the following elements
- $CrStatus_i$
- Event type
- History $date_i$
- Team

else
create IDP_i and add the following elements
- $CrStatus_i$
- Event type
- History $date_i$
- Team

In order to add value to the Event types in the XML file, it is necessary to read the CrStatus values and create the corresponding Event type. It is to say:

c. Read data from snapshot
d. Evaluate CrStatus

If CRstatus starts with "in_",
then
create Even type = Start
else
create Event type = Complete

After applying these instructions, it is obtained an XML file with the following characteristics:

1. The XML file contains all the CI's defects from all the snapshots, this includes CI's defects that were submitted in early time, but later they were deleted from the database.
2. The XML file contains only the tasks that have been performed for a CI's defect, and excludes identical tasks, so the path for the CI's defect is complete and correct.

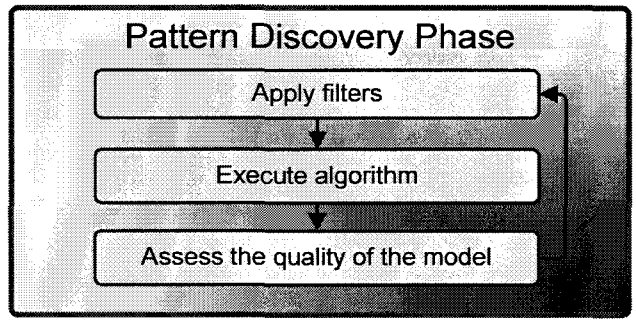
The code of the program can be found in appendix A of this report.

5.2 Pattern Discovery Phase

The pattern discovery phase for this first attempt is carried out using the steps shown in Figure 5.3. As it was mentioned before the objective of the first attempt is to be able to recognize what are the elements

that need to be improved when applying Process Mining techniques in the data sets coming from the CCB process. In the next sub-sections it will be briefly described what was done in this phase; however, the emphasis will be in the last steps where an assessment to the discovered model was carried out.

Figure 5.3. Pattern Discovery and its phases
Based on:[12], [13], [14], [15].



5.2.1 Apply Filters

In the XML file obtained in the previous phase, the following characteristics of its structure were found:

1. The XML file contains all the elements that are required for letting the Software tool ProM to execute the discovery of the process model.
2. The XML file contains the 8832 CI's defects
3. The paths of the CI's defects does not start always with the activity submitted
4. The paths of the CI's defects does not finish always with the activity close

An important observation about the number of paths of the CI's defects is bigger than the number of CI's defects submitted in the last snapshot. It is because the program takes into account also the CI's defects that were initially introduced to the system and captured in the first snapshots, but later they were deleted in the system, and not captured by the last snapshots.

One of the properties of the snapshots is that they contain all the CI's defects handled by the CCB in the project until the moment when the snapshot was taken. The XML generated is influenced by this characteristic. It contains CI's defects that were in the middle of their handling in the first snapshot, showing their current task's event in the first snapshot as the initial task's event of their handling, and it also contains CI's defects that have not been completed in the last snapshot, in this case their end task's event will be the current task's event shown in the last snapshot. These CI's defects can disturb the process discovered showing a process with many starts and many ends, giving the idea that it is possible to have CI's defects with any start and any end. In order to solve this problem, filters that select only CI's defects that really started their handling in the first snapshot and CI's defect that really finished their handling in the last snapshot will be considered. The following filters provided by ProM tool were used:

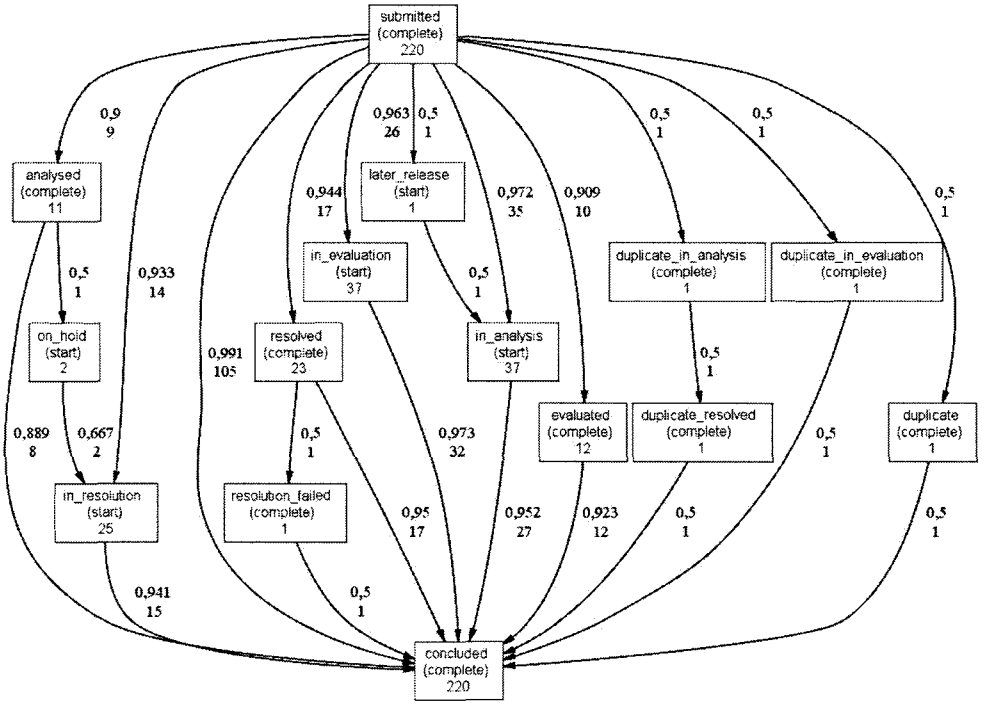
- Start Event Log filter – selecting only cases that start with the task "Submitted (complete)"
- Final Event Log filter – selecting only cases that finish with the task "concluded (complete)"

After applying the filters to the original event-log it was obtained a new event-log containing 220 suitable cases. This represents 2.5% of the total number of CI's submitted to the CCB until the time when the last snapshot was taken. This is an important point that will have to be improved in the next attempt of the experiment.

5.2.2 Execution of the Process Mining Algorithm

After having an adequate event-log for Process Mining, the Process Mining software tool was used. Here the Heuristic algorithm was applied to the event-log. The application of this algorithm is straightforward, because it was used the predetermined values of thresholds required by this algorithm. The result of the mining is shown in figure 5.4.

Figure 5.4. Process model of the CCB process in the first attempt



5.2.3 Assess the Quality of the Process Model Discovered

In order to make an appropriate assessment to the process model discovered, it was carried out an analysis of how well the model describes the event-log and the paths within it (conformance checking), and if there is a logical sequence of the tasks that correctly describe the CCB process.

- **Conformance checking**

In order to analyze how well the model discovered represents the paths within the even-log, it was carried out an analysis of Conformance checking. These techniques execute each path within the model discovered and verify if the model can execute the paths of the event-log. The value of a perfect fit between the process discovered and the event-log is 1 and the value of the worst fit is 0. The model discovered reaches a conformance of 96.7% which means that 212.7 of the 220 cases found in the event-log can be represented in the model. Figure 5.6 shows the results of the Conformance checking test in the process model obtained once it has been converted to a Petri net.

- **Assess the logical structure of the process model discovered with respect to the CCB process**

Making a check of how well the process model explains the CCB process the following characteristics were found:

1. The structure of the process does not show complete executions of tasks. It is to say that there is a direct connection between task's events of different tasks during the handling of a CI's defect. For example there are cases that have the following sequence:

Submitted → analyzed → concluded

It is argued that this sequence is not correct, and if a case passes through the activities "Submit", "Analysis", and "Closing", they should have the following sequence:

Submit (start) → Submit (complete) → Analysis (Start) → Analysis (complete) →Close (complete)

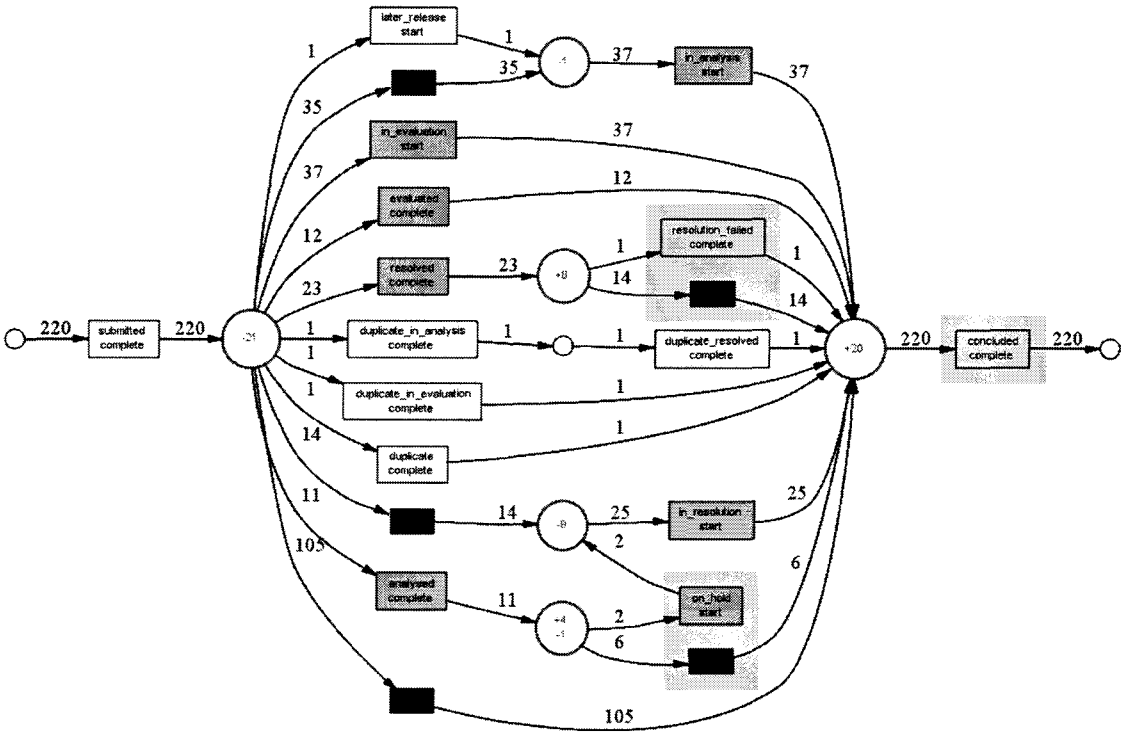
Thus, the process model created does not reflect correctly the structure of the CCB process, even if some sequences in the structure give some insight of the possible combinations of tasks, this information is not rich enough in order to make a worthy analysis of the process model discovered.

2. The structure of the process model shows cases that just skip complete tasks. In order to verify if this construction were correct the original data set was inspected and compared with the results. For that the following sequence was used:

Submitted (complete) → concluded (complete)

The process model shows that there are 105 cases (47 % of the cases used) that follow this sequence; however, reviewing the original data set, it was found that all these cases had dates in all the "Processing data elements", which means that there were tasks that handled the CI's defects but not captured by the current attempt. The problem is caused because the "CrStatus" only shows the current task's event that was handling the CI's defect when the snapshot was taken, so the tasks that have handled the CI's defect between snapshots are not captured in the event-log and not shown in the process. Using snapshots taken in a more frequent basis could solve easily the problem. For example using snapshots taken in a daily basis would capture all the tasks that have handled the CI's defects. Of course for this solution it is necessary to know if the database is updated in a daily basis as well. Other solution would be the use of additional data elements that give information about the tasks that have been used between snapshots. This information is available in the "Processing data elements". The next attempt will use these data elements in order to solve the problem.

Figure 5.5. Process Checking result for the first attempt



Chapter 6

6 Second Attempt of the Experiment

Given the observations made to the previous attempt, the present section describes a new attempt to discover the real CCB process. This attempt will follow the same method's structure proposed in chapter 4. In addition, because this new attempt is based on what has been learned in the previous attempt, it will reuse part of the information found in the previous attempt, so the phases will be focused on new observations and only if necessary they will brush it on content of the previous attempt. Before starting with the description of the phases a list the objectives of this attempt is presented:

- The method must increase the number of suitable cases in the event-log for the Pattern Discovery phase and Analysis phase.
- The event-log must contain complete tasks, it is to say it has to provide information about start and end events of tasks within the structure of the process model.
- The event-log must contain all the tasks that have handled the CI's defects shown in the data set.

Because the event-log is constructed in the Preparation phase, this is the phase in which the improvements should be carried out in this new attempt.

6.1 Preparation Phase

This phase will consider an increasing the number of data elements that describe the CCB process. Besides the "CrStatus" and the "History date" data elements this attempt will use the "Processing data elements" in order to overcome the problem presented in section 5.2.2.

This attempt will use the same steps described in chapter 4 for this phase, but with a slight different in the organization of them. The understanding of the data and the identification of the process elements will be simultaneously executed and leaving the remaining steps in the same order.

6.1.1 Understanding the CCB Data and Identification of Data Elements

From the four elements that describe the process (case, task, time stamp, and resource), the "tasks" and the way that they are organized within the CCB process is the information that needs a better understanding. The previous attempt used the "CrStatus" and "History date" as the only sources for obtaining information about the CCB process. In this new attempt the remaining data elements with information about the tasks within the process will be used, called "Processing data elements" in this project. However, these data elements store only the successful execution of the tasks (see section 5.1.1), so using them requires a special treatment, they are presented in table 6.1.

Table 6.1. Processing data elements in the CCB process

Data element	Content
Create time	Date when the CI's defect has started the "Submit" task (Not found in the "CrStatus" data element)
Submitted time	Date when the CI's defect has completed the "Submit" task
In analysis	Date when task "Analysis" is started for handling the CI's defect
Analyzed	Date when task "Analysis" has been successfully completed for a CI's defect
In resolution	Date when task "Resolution" is started for handling the CI's defect
Resolved	Date when task "Resolution" has been successfully completed for a CI's defect
In evaluation	Date when task "Evaluation" is started for handling the CI's defect
Evaluated	Date when task "Evaluation" has been successfully completed for a CI's defect

A limitation of only looking at these data elements, especially in only one snapshot, is that they give the wrong impression that the process is totally linear. In order to overcome this limitation it will be used the

"CrStatus" data element. The advantage of the "CrStatus" is that within its list of values it is possible to find if a task was either successfully or unsuccessfully executed, the state of the CI's defects after the handling of a task, and when it was closed. The list of values of the "CrStatus" is presented in table 6.2.

Table 6.2. "CrStatus" list of possible values

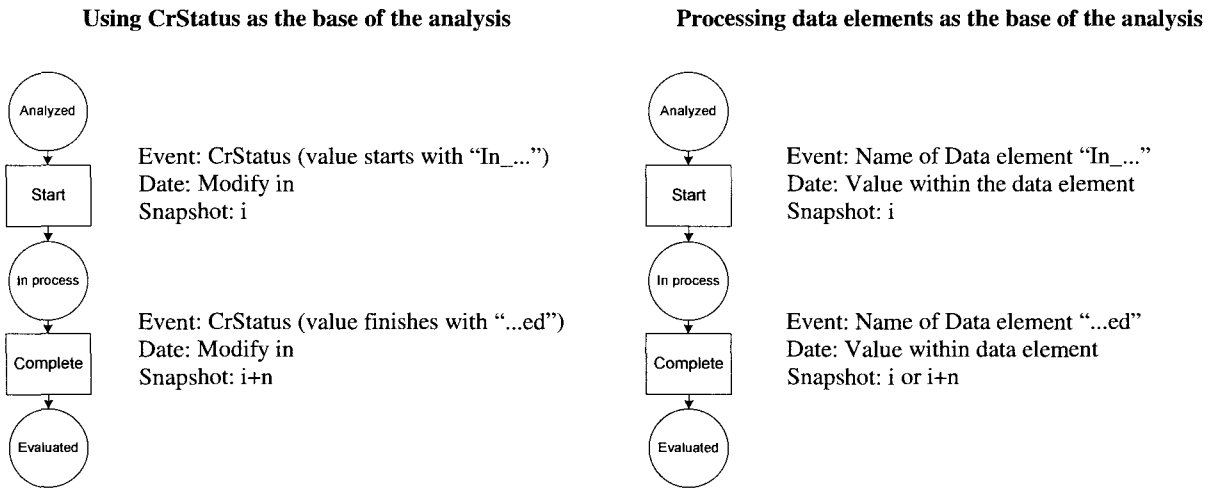
CrStatus value	Meaning of the value
Submitted	CI's defect has completed the "Submit" task
In analysis	The "Analysis" tasks is started for handling the CI's defect
Analyzed	The "Analysis" task has been successfully completed for a CI's defect
Analysis failed	The "Analysis" task has been unsuccessfully completed for a CI's defect
In evaluation	The "Evaluation" task is started for handling the CI's defect
Evaluated	The "Evaluation" task has been successfully completed for a CI's defect
Evaluation failed	The "Evaluation" task has been unsuccessfully completed for a CI's defect
In resolution	The "Resolution" task is started for handling the CI's defect
Resolved	The "Resolution" task has been successfully completed for a CI's defect
Resolution failed	The "Resolution" task has been unsuccessfully completed for a CI's defect
Other values not found in the other data elements that describe the process	
Concluded	The CCB board approves the handling of the CI's defect
Later release	The CI's defect is "rejected" because the changes is planned as part of the next release
Not reproducible	The CI's defect is "rejected" because the changes are not within the scope of the project
On hold	The CI's defect handling in the CCB process is hold on – the same result as "rejected"
Rejected	The CI's defect is "rejected"
Duplicate	The CI's defect was found "duplicated"
Only found in the first six snapshots and later replaced by only "Duplicate"	
Duplicate submitted	The CI's defect is found "duplicated" and task's event where the it was found within the CCB process Note. These values are found only in the first 6 snapshots, since snapshot 7 these values are replaced by "Duplicate"
Duplicate in analysis	
Duplicate analyzed	
Duplicate in resolution	
Duplicate resolved	
Duplicate in evaluation	
Duplicate evaluated	
Duplicate concluded	
Duplicate on hold	
Duplicate not reproducible	
Duplicate later release	
Duplicate rejected	

The difference between these two sources of information is that the "CrStatus" provides the name of the task's events and possible results of the handling of a CI's defect, and it is necessary to use the "History date" data element in order to obtain the time of execution of the task's events in the 'CrStatus". Conversely, the "Processing data elements" provide directly both types of information. The names of the data fields correspond to the task's events (for instance "in analysis" represents the event "start" of the task "Analysis") and their values are the dates of execution of the event. Additionally, while with the "CrStatus" it is not possible to have more than one value per snapshot, the "Processing data elements" can contain the whole handling of the CI's defect in one snapshot. These differences are graphically presented in figure 6.1.

The advantages of integrating both data sources are:

1. Capability to identify a bigger number of the tasks executed during the handling of CI's defect with less dependency on the frequency of taking the snapshots.
2. Capability to identify successful and unsuccessful execution of tasks
3. Capability to include task's events from the Processing data elements that were submitted since the beginning of the project.

Fig. 6.1. Difference of identifying task's events between "CrStatus" and the Processing data elements



Tasks can be either successfully or unsuccessfully executed. To detect those behaviors the values of the "CrStatus" were used. A successful execution of tasks can be detected in two ways: (1) use only the "Processing data elements" related to the execution of task, or (2) use a combination of the "Processing data elements" with the "CrStatus". Table 6.3 shows these two strategies.

Table 6.3. Combination of sources of information for constructing the task "analysis" with successful execution

Task	Events	Data element describing the name	Execution Time of the event
Analysis	Start	In analysis (name of the field)	In analysis (value the belong to the In analysis data element)
		CrStatus (value)	Modify in (value)
	Complete	Analyzed (name of the field)	Analyzed (value the belong to the Analyzed data element)
		CrStatus (value)	Modify in (value)

To identify unsuccessful execution of tasks, it is used a combination of "Processing data elements" that provide the event "Start" and the value of the "CrStatus" equal to "..._failed" for the "Complete" event of the task. Table 6.4 shows the combination for creating a task with unsuccessful execution.

Table 6.4. Combinations of sources of information for constructing the task "analysis" with unsuccessful execution

Task	Events	Data element describing the name	Execution Time of the event
Analysis	Start	In analysis (name of the field)	In analysis (value)
	Complete	CrStatus (value with "..._failed")	Modify in (value)

In addition, there is a period of time between the completion of a task and the start of the next task. The consultant identifies this time as the waiting time before the execution of the next task, where in case of important CI's defects the CCB board evaluates the execution of the task previously executed. In the "CrStatus" data element, it is possible to identify these evaluations, those are: duplicate, rejected, later release, not reproducible, and on hold. It is argued that these values are the outcome of a task called "CCB evaluation". This task is the evaluation made by the CCB board to the handling of a CI's defect by the tasks: Analysis, Resolution and Evaluation. However, because these elements only appear in the "CrStatus", they are present few times in the sequence of a CI's defect handling. Here it is assumed that the "CrStatus" with this values represent the "start" task's event of the "CCB evaluation" task, and because it is necessary to have a "complete" task's event for this task, and artificial task's event will be

created with a timestamp equal to the "start" task's event plus 1 hr. It is argued that this artificial task will not significantly disturb the time of performance of the handling because the data set only provides references of time in days. The advantage of using this artificial task is that the process will contain correct structure of activities. This assumption needs to be confirmed with the practitioners in order to validate it. The combination of data elements for these special tasks is presented in table 6.5.

Table 6.5. Combinations of sources of information for constructing the "CCB evaluation tasks"

Task	Events	Data element describing the name	Execution Time of the event
CCB evaluation	Start	CrStatus (special value)	History date (value)
	Complete	CrStatus (special value)	History date (value)+1 hr

6.1.2 Design the Structure of the Process Model

Like in the previous attempt, here is presented a description of the structure of the process based on the data elements identified in the data set.

The event-log will be composed by a sequence of paths, each path for each CI's defect. This attempt will organize the paths according to the tasks that handled the CI's defect and the attributes and team that carried out the handling of the CI's defect.

$$Path_i = \{IDP_i, [Task_1, \dots Task_o], Resources, Attributes\}$$

Where,

IDP_i = ID of the CI's defect = "Problems number" + "Subsystem"

The tasks are composed by two task's events structured in the following way:

$$Task_i = \{start\ event, complete\ event\}$$

Tasks that appear in the data set can be either successfully or unsuccessfully executed

a. Successfully executed task

$$Start\ event = \{Processing\ data\ element(in\_\dots), Processing\ data\ element(date\ value)\}$$

$$Complete\ event = \{Processing\ data\ element(\dots ed), Processing\ data\ element(date\ value)\}$$

b. Unsuccessfully executed task

$$Start\ event = \{Processing\ data\ element(in\_\dots), Processing\ data\ element(date\ value)\}$$

$$Complete\ event = \{CrStatus(\dots failed), History\ date\}$$

c. The special tasks will have the following structure

$$Starte\ event = \{CrStatus(special\ value), Hisotry\ date\}$$

$$Complete\ event = \{CrStatus(special\ value), Hisotry\ date\}$$

The next section will apply these rules to the data set, and it is expected that based on these definitions a cleaning and adaptation of the data set will be carried out before transposing the data to the event-log.

6.1.3 Apply Rules and Construct the Event-Log

The previous sections gave an understanding of what are the elements that are required to be used from the data set at the moment to create the event-log. Like in the previous attempt a program will be created that will carry out the automatic construction of the event-log by using a small program that will read the

data and manipulate the data elements in order to produce an adequate even-log in XML format. In order to provide a more understandable explanation of the program it will be used a sample of CI's defect that is extracted from the data sets.

6.1.3.1 The CI's Defect 2714 Case

A CI's defect extracted from the snapshot will be used in order to show how the program will create the event-log. For that it was selected the CI's defect number 2714 that has been handled in the given range of time of the snapshots for the present report. Here only the information that is referred to the process structure will be used. The additional data elements (attributes) will be used only if they are required. In addition, confidential information has been removed or changed with more general values. The whole sequence of snapshots for this CI's defect can be found in appendix B. A short description of the first snapshot containing only the elements used in this example can be found in table 6.5.

Table 6.5. Data in the first snapshot of CI's defect 2714

Field	Value
History Date	22-11-06
Subsystem	xxx
Problem nr	2714
Priority	Medium
Severity	B
Request type	PR
CrStatus	Submitted
Create time	04-09-06
Submitted time	04-09-06
In analysis time	
Analyzed time	
In resolution time	
Resolved time	
In evaluation time	
Evaluated time	
Modify time	22-11-06
Team	PSV

6.1.3.2 Implementation of the Program

The new program required for this attempt is different with respect to the program made for the first attempt, because in each snapshot it is possible to find a sequence of tasks that have to be adequately merged with the other task sequences of the other snapshots. The program should do the following:

1. **Organization of the data elements within the snapshot**
   This includes the following sub-activities
   a. **Automatic identification of the data elements**
   b. **Automatic creation of a sequence of events**

The program read each snapshot and allocates each data element in a field of a table in the database. The outcome of this part of the program is a sequence of task's events, which contains all the data related to the CI's defect. The structure of the sequence created by the program is:

$$Sequence_i = \{IDP_j, [Event_1, Event_2, \ldots, Event_n], Resources, Attributes\}$$

The sequence of task's events is composed by the "Processing data elements" and the "CrStatus – History date" data elements. They will be sorted according to their time given in the snapshot. For the CI's defect 2714 the outcome of the program for the sequence of the first snapshot would be similar to the following:

$Sequence_1$ =
{
2714-xxx,
[created(04-09-06),submitted(04-09-06),submitted(22-11-06)],
PSV,
[priority(medium),severity(B),request type(PR)]
}

As it can be seen, only the data elements that contain a value are included in the Sequence. In this example a colored text is applied to the "CrStatus" data element. The "CrStatus" contains the same name value than the Processing data element "Submitted time" with the difference that it contains a later date. Initially these two values are preserved, and only in step 2c they will be reprocesses in order to eliminate the task's events that do not correspond to the correct sequence. The program would be in charge of doing the same for all CI's defects in the snapshots used in the present report.

2. Organization of the outcomes of the snapshots in one unique path.

It contains the following sub-activities

a. Automatic integration of sequences of tasks into

This operation integrates the sequences created in the previous step in one unique path containing all the elements of each sequence, this path has the following structure:

$$Path_{IDP}=\{Sequence_1, Sequence_2, \ldots, Sequence_n\}$$

Below is presented the path for the example of the CI's defect 2714:

$Path_{2714\text{-}xxx}$ =
{
$Sequence_1$ =
{
2714-xxx,
[created(04-09-06),submitted(04-09-06),submitted(22-11-06)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_2$ =
{
2714-xxx,
[created(04-09-06),submitted(04-09-06),submitted(20-12-2006)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_3$ =
{
2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),in analysis(24-01-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_4$ =
{
2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),in analysis(21-02-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_5$ =
{
2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),in analysis(21-03-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_6$ =
{
2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),analyzed(27-03-2007),in resolution(18-04-2007),in resolution(18-04-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_7$ =
2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),analyzed(27-03-2007),in resolution(18-04-2007),in resolution(30-05-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_8$ =

2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),analyzed(27-03-2007),in resolution(18-04-2007),in resolution(25-06-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_9$ =
2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),analyzed(27-03-2007),in resolution(18-04-2007),in resolution(25-07-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_{10}$ =
2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),analyzed(27-03-2007),in resolution(18-04-2007),in resolution(22-08-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_{11}$ =
2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),analyzed(27-03-2007),in resolution(18-04-2007),in resolution(26-09-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_{12}$ =
2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),analyzed(27-03-2007),in resolution(18-04-2007),in resolution(03-10-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
},
$Sequence_{13}$ =
2714-xxx,
[created(04-09-06),submitted(04-09-06),in analysis(18-01-2007),analyzed(27-03-2007),in resolution(18-04-2007),resolved(17-10-2007),in evaluation(19-10-2007),evaluated(19-10-2007),concluded(22-10-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
}
}

b. Organizing the events in one unique path

This step is focused on sorting the task's events by their time stamps. The result is that similar task's events form groups that contain similar information about names and timestamps. This new path will have the following structure:

$Path_{IDP}$={($Event_{1\text{-}Seq1}$, $Event_{1\text{-}Seq2}$, …, $Event_{1\text{-}Seq\,n}$), ($Event_{2\text{-}Seq1}$, $Event_{2\text{-}Seq2}$, …, $Event_{2\text{-}Seq\,n}$), …, ($Event_{m\text{-}Seq1}$, $Event_{m\text{-}Seq2}$, …, $Event_{m\text{-}Seq\,n}$) }

The path created for the CI's defect 2714 is the following:

Path2714-xxx =
{2714-xxx,

[created(04-09-06),created(04-09-06),created(04-09-06),created(04-09-06),created(04-09-06),created(04-09-06),created(04-09-06),created(04-09-06),created(04-09-06),created(04-09-06),created(04-09-06),created(04-09-06),

[submitted(04-09-06),submitted(04-09-06),submitted(04-09-06),submitted(04-09-06),submitted(04-09-06),submitted(04-09-06),submitted(04-09-06),submitted(04-09-06),submitted(04-09-06),submitted(04-09-06),submitted(04-09-06),submitted(04-09-06),submitted(22-11-06),submitted(20-12-2006)],

[in analysis(18-01-2007),in analysis(18-01-2007),in analysis(18-01-2007),in analysis(18-01-2007),in analysis(18-01-2007), in analysis(18-01-2007),in analysis(18-01-2007),in analysis(18-01-2007),in analysis(18-01-2007),in analysis(18-01-2007),in analysis(18-01-2007),in analysis(24-01-2007), in analysis(21-02-2007, in analysis(21-03-2007)],

[analyzed(27-03-2007),analyzed(27-03-2007),analyzed(27-03-2007),analyzed(27-03-2007),analyzed(27-03-2007),analyzed(27-03-2007), analyzed(27-03-2007),analyzed(27-03-2007)],

[in resolution(18-04-2007),in resolution(18-04-2007,in resolution(18-04-2007),in resolution(18-04-2007,in resolution(18-04-2007),in resolution(18-04-2007),in resolution(18-04-2007), in resolution(18-04-2007),in resolution(18-04-2007),in resolution(30-05-2007),in resolution(25-06-2007),in resolution(25-07-2007),in resolution(22-08-2007),in resolution(26-09-2007),in resolution(03-10-2007)],

[resolved(17-10-2007)],

[in evaluation(19-10-2007)],

[evaluated(19-10-2007)],

[concluded(22-10-2007)],

PSV,
[priority(medium),severity(B),request type(PR)]
}

c. Identification of the resources that execute the tasks

When analyzing the data set, it was found that some CI's defects are handled by two or more teams (a total of 73 cases). This is a problem at the moment to construct the event-log because the snapshots contain the description of more than one task at a time, and only one resource (team) as data, so when looking only one snapshot it is possible to have the wrong impression that all the tasks of the snapshot have been executed by this team. If the program does not consider this problem it would produce many paths for one CI's defect, each of them showing a different team.

The solution to this problem is that the program should be able to produce tasks linked to their real executors. Once having integrated all the sequences in step 2a, the program will make a comparison of CI's defects looking in the elements: IDP, Event type name, time stamp, and resource. If the elements IDP, Event type name and time stamp are the same, but the resource is different, the program will replace the resource value of all the group of IDPs by the resource value of the first IDP of the group. The first IDP is taken because the program has introduced the snapshots starting with the earliest snapshot, so the value of the resource (team) in the earliest snapshot is the real executor of the task.

The task's events coming from the "CrStatus" data element have a special treatment, because they will always contain a different time stamp in the sequence. Thus, the program also identifies if the IDP analyzed is coming from the "CrStatus" in order to make the substitution. For that a data field was introduced in the database that indicates the origin of the task's events.

The advantage of this strategy is that during the execution of the next step it will be possible to delete all the event types repeated, and leave only one in the sequence that is linked to its correct executor.

d. Elimination of duplicated events

Here the program is in charge to eliminate the events that are duplicated. This activity eliminates repeated tasks according to the name, and only in the case of the "CrStatus" eliminates them if they are in sequence. The path for the CI's defect 2714 is the following:

Path2714-xxx =
{2714-xxx,
[created(04-09-06)],[submitted(04-09-06)],[in analysis(18-01-2007)],[analyzed(27-03-2007)],[in resolution(18-04-2007)],[resolved(17-10-2007)],[in evaluation(19-10-2007)],[evaluated(19-10-2007)],[concluded(22-10-2007)],
PSV,
[priority(medium),severity(B),request type(PR)]
}

e. Automatic creation of missing elements

Until this moment, what has been obtained is a path of a CI's defects composed by the task's events found in the snapshots. However, due that the filling in of the data is manually carried out, some data are not properly recorded in the database, so they do not appear in the snapshots as well.

A correct description of a task always contains the events "Start" and "Complete"; it was evidenced situations when the CI's defect handling in the process as not been correctly filled in and some of the two events of a task is missing. In the data set five possible situations were found, they are presented in table 6.6 followed by the strategy that will be used in the program. In order to execute this analysis, the program checks pairs of task's events and their times in order to see if it necessary to insert one or two artificial task's event in the middle of the task's events. The timestamps are used in order to sort the sequence of task's events and the names are used in order to distinguish the correct and incorrect sequences of task's events. Because the analysis is carried out pair by pair, only one possible situation can happen, and only one possible strategy for overcoming the problem is applied.

Table 6.6. Possible not correct sequences of events and the strategies for fixing these situations

Sequence	Strategy
Two consecutive "Start" events coming from different tasks Example: [created(04-09-06T00:00:00)],[in analysis(18-01-2007T01:00:00)]	Consider putting artificial "Complete" event in the middle of both "Start" events. The artificial event will belong to the same task of the first "Start" event, and its date and time will be equal to the first event's time of the sequence plus 1/2 hr. The results will be: [created(04-09-06T00:00:00)],[submitted(04-09-06 T00:30:00)],[in analysis(18-01-2007T01:00:00)]
Two consecutive "Start" events coming from equal tasks but different dates Example: [in analysis(18-01-2007T00:00:00)],[in analysis(22-01-2007T00:00:00)]	Take both events as two separate executions of the same task, and apply the same strategy than the previous case. The results will be: [in analysis(18-01-2007T00:00:00)],[analyzed(18-01-2007T00:30:00)], [in analysis(22-01-2007T00:00:00)]
Two consecutive "Complete" events coming from different tasks Example: [analyzed(27-03-2007T00:00:00)], [evaluated(01-04-2007T01:00:00)]	Consider putting an artificial "Start" event in the middle of the two "Complete" events. The artificial event will belong to the same task of the second "Complete" event, and its date and time will be equal to the first event's time of the sequence plus 1/2 hr. The results will be: [analyzed(27-03-2007T00:00:00)],[in evaluation(27-03-2007T00:30:00)] [evaluated(01-04-2007T01:00:00)]
Two consecutive "Complete" events coming from equal tasks but different dates Example: [analyzed(27-03-2007T00:00:00)], [analyzed(01-04-2007T01:00:00)]	Take both events as two separate executions of the same task, and apply the same strategy than the previous case. The results will be: [analyzed(27-03-2007T00:00:00)], [in analysis(27-03-2007T00:30:00)],[analyzed(01-04-2007T01:00:00)]
One "Start" event followed by a "Complete" event but from a different task. Example: [in analysis(27-03-2007T00:00:00)], [evaluated(01-04-2007T01:00:00)]	Produce two artificial events: The first is a "Complete" event that will complement the "Start" event. It will have a date and time equal to its pair event plus 20 min. The second is a "Start" event that will complement the "Complete" event. It will have a date and time equal to the value of the "Start" plus 40 min. The results will be: [in analysis(27-03-2007T00:00:00)],[analyzed(27-03-2007T00:20:00)],[in evaluation(27-03-2007T00:40:00)],[evaluated(01-04-2007T01:00:00)]

In order to apply correctly these strategies, first it is added an artificial time to the task's events of the paths. The time that each of them received is not the same, the first task's event of the path receives a time equal to 00:00.00, and the next will receive a time equal to the previous task's event plus 1 hour. This assignation of times will be made using the same method for the remaining task's events of the path.

It is important to note that the time is not stored in the data set, and it will be added only in order to give a reference of the sequences of the task's events in the path.

These rules are applicable for all the task's events including the special ones mentioned before as the "CCB evaluation tasks". The application of these activities is carried out to all the paths, for that the program works with queries in the database.

f. Generation of tasks based on the task's events found in the paths

So far, the elements handled by the program were the task's events; moreover, as it was mentioned in section 5.2.1 there is not an explicit data element presenting the tasks that handled the CI's defect, so it is necessary to create the tasks that contain the task's events shown in the snapshots. In order to find the tasks used in the paths of the CCB process, the same method used in the previous step will be applied. It is to say that the program will analyze task's events within the paths by pairs. Because the paths contain both "Start" and "Complete" events (either original or artificial) already sorted, whenever the program will find a sequence of a "Start" event followed by a "Complete" event a task will be created. Table 6.7 shows the combinations used in order to create the tasks.

Figure 6.7. Tasks created based on the task's events of the data set

Tasks	Task's events used
Submit	Start → Created Complete →Submitted
Analysis	Start →In analysis Complete →Analyzed Complete →Analysis failed
Resolution	Start →In resolution Complete →Resolved Complete →Resolution failed
Evaluation	Start →In evaluation Complete →Evaluated Complete →Evaluation failed
CCB evaluation	Start →Later release Complete →Later release (artificial)
	Start →Not reproducible Complete →Not reproducible (artificial)
	Start →On hold Complete →On hold (artificial)
	Start →Rejected Complete →Rejected (artificial)
	Start →Duplicate Complete →Duplicate (artificial)

g. Automatic transformation of the paths to the structure of the XML file required

In this step the program will automatically extract the data elements that describe the CCB process from the database, and it will construct the XML file. This step includes the some transformations of the names of the task's event to "Start" and "Complete" instead of the names that have been handled so far. In addition, it will introduce the attributes of each CI's defect, including a special attribute created in order to store the outcome of the handling by the tasks based on the Complete task's events of the event-log, i.e. if the task's event contains a value equal to "..._failed" the value will be "unsuccessful"; conversely if the value of the task's event is other the value of this attribute will be "successful". In this way it is possible to introduce the values of the outcomes of each task within the process model, and understand the paths generated by the process mining algorithm at the moment to create the process model. Appendix D presents the XML for the path created by the CI's defect 2714.

6.2 Pattern Discovery Phase

This phase of the experiment will use the event-log generated during the Preparation phase. This section is dedicated to describe that steps that have been carried out in order to obtain the process models that will help to answer the research questions formulated in chapter 1. In addition, during the work in the first step of this phase (Apply filters) a selection of the attributes will be considered. Like in the previous attempt this phase of the experiment will use the Process Mining tool "ProM". The description of this phase is based on the event-log composed by all the cases found in the snapshots. The procedure is made for all the selected process models that will be used in the Analysis phase.

6.2.1 Apply Filters

As it was mentioned in section 2.8.1 the CCB assigns to each CI's defect a list of attributes. Finding the way that these attributes affect the behavior of the CBB process will answer the three last research questions. In order to analyze the attributes, it is necessary to group CI's defects that contain equal values for each attribute. Thus, two types of filters are proposed:

1. **In order to fix the semantic of the process model**

The CI's defects can be introduced to the CCB process any moment, so it is possible to have cases in the middle of their processing in the first snapshots, but also cases whose handling is not completed in the last snapshot used in the experiment. Thus, in the event-log there are cases whose final task in the sequence is not "Conclude" but any other of the process. A well constructed process model is the one that has a unique ending state, and does not present hanging tasks in the process model. These concepts are used during the design of process models in order to assure the correctness of the execution. Besides these two concepts, the algorithms used in the Pattern Discovery phase, will group cases that follow the same patterns, thus it will be used a filter that inserts an artificial end task in order to allow the algorithm to group finished cases with unfinished cases that contain the same attributes.

2. **In order to group cases according to their CI's defect's attributes**

This type of filters group CI's defects according to their attributes. Table 6.8 presents the selected attributes and process models that will be constructed. The filters used in the present experiment are based on these requirements. Thus the filters will select specific values to the attributes of the CI's defects in order to generate the process models.

Table 6.8. Criteria for selecting the filters

Attribute or combination of attributes	Criteria	Process models created
Priority	Investigate the difference between CI's defects with high priority and low priority	Model containing only CI's defects with high priority. Model containing only CI's defects with low priority.
Severity	Investigate the difference between CI's defects with high severity and low severity	Model containing only CI's defects with high severity. Model containing only CI's defects with low severity.
Request type	Investigate the difference between CI's defects with different types of request changes	Model containing only PR requests Model containing only CR requests Model containing only IR requests
Team	Investigate the difference of productivity between teams	Models containing CI's defects grouped by the team that executed the tasks
Caused during	Investigate the effect of the CI's defects in different phases of the project	Models containing CI's defects submitted during different phases of the project

6.2.2 Execution of the Process Mining Algorithm

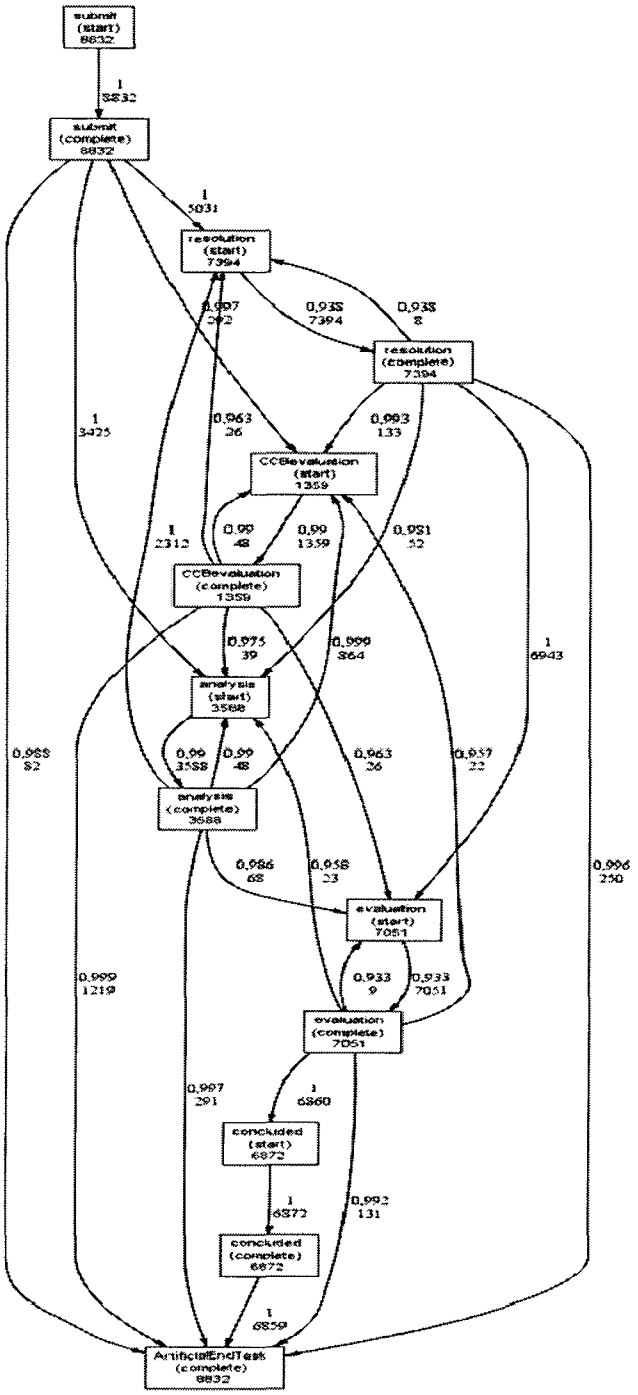
Like in the previous attempt, it will be used the Heuristics algorithm in ProM tool for discovering the process models. To maintain the same rules of identification for all the process models, the same predetermined indices of the algorithm were used. Those are presented in table 6.9.

Table 6.9. Predetermined values of the Heuristic algorithm

Index	Value
Relative-to-best threshold	0.05
Positive observations	10
Dependency	0.9
Length-one-loops threshold	0.9
Length-two-loops threshold	0.9
Long distance threshold	0.9
Dependency divisor	1
AND threshold	0.1

Figure 6.2 shows a process model containing all the cases within the event-log (8832 cases). For the generation of this process model it was necessary to use an "artificial" end task in order to obtain a sound process model. Conversely the previous attempt, it is not required to insert an artificial start task because all the CI's defects contain the correct task that starts the process. Comparing this process model with the ones presented in figures 2.5 and 5.2, it is possible to identify a more complex structure that includes many different sequences and loops.

Fig. 6.2. Process model discovered taking into account all the cases



6.2.3 Assess the Quality of the Process Model Discovered

The process model presented in figure 6.2 shows a different structure than the one shown in figures 2.5 and 5.4. In order to know how well this process model is representing the CCB process model, an assessment of its quality is carried out, considering the same perspectives used in the previous attempt.

- **Conformance checking**

Applying the Conformance checking analysis algorithm in the process model it was found a fitness of 99.96% with respect to the cases within the event-log file. Three additional metrics are provided:

- **Behavioral Appropriateness**

It is the metric that indicates which additional paths can be found in the process model besides the ones that appear in the event-log. The value 1 means that the process model generates allows only the type of paths found in the event-log.

- **Degree of Model Flexibility**

It determines the degree of possible sequences of steps, where a value of 0 means that the model only allow one possible sequence, and the value of 1 means that any kind of possible sequences. The value obtained for the model is equal to 0.54. This means that the process model created has a quasi intermediate flexibility.

- **Structural appropriateness**

It considers how well the structure of the process model discovered is, specially focused on the possibility to have duplicate tasks or invisible tasks that can affect the behavior of the process. The value obtained for this model is equal to 1 that means that the process contains an appropriate structure without duplicate tasks or required invisible tasks. The conformance checking analysis plug-in presents the model inconsistencies found during the analysis, table 6.9 shows them.

Table 6.9. Number of inconsistencies when applying the Conformance checking algorithm

Place	Missing or reaming tokens after the analysis
Case submitted	+3
Case submitted and evaluated by the CCB board	-1
Case evaluated by the CCB board	+3
Case analyzed	+6
Case analyzed and evaluated	-17
Case resolved	+20
Case analyzed, resolved and evaluated by the CCB board	-8
Case evaluated	+3 -12

It is possible to see that the values of missing and remaining tokens are very low, with the lowest value equivalent to the 0.01 % and the highest with a value of 0.23%.

As a conclusion, it is argued that the model discovered can reflect adequately the CCB process model from the point of view of its structure.

- **Assess the logical structure of the process model discovered with respect to the CCB process**

Because during the preparation phase it was carried out a process of cleaning, this included completing task's events and the elimination of repeated ones. All the task elements are adequately correct. It is to say that each task contains two events (start and complete). Thus, comparing the present process model with respect to the one created during the first attempt, the current model represent more the real behavior of the tasks where the tasks are certainly completed. However it is necessary to say that, the correction

(adding artificial task's event) requires refinement in order to capture the real time that consumes the execution of a task.

In conclusion, it is argued that the content of the event-log provides all the elements required for the construction of adequate process models with the Heuristics algorithm. Moreover, the process models generated include information about the attributes of CI's defect. This will allow the analysis of structure and performance of groups of CI's defects containing specific attribute's values. This analysis will be carried out in Analysis phase of the Process Mining framework presented in the next chapter.

Chapter 7

7 Analysis of the Results

The present chapter will make a series of analysis to the CCB process based on the process models that can be discovered with the method described in the present report. The analyses will take into account two perspectives: (1) the structure of the process influenced by the CI's defects and their attributes, and (2) the performance of the process affected by the CI's defects and their attributes.

In this chapter it is expected to come up with enough information in order to answer the four research questions proposed for the present project. Thus, the content of this chapter is the following: first a brief description of the type of analyzes will be presented. Next it will be carried out an analysis of the CCB process with all the CI's defects found in the snapshots. Thereafter, analyses to the CCB process by groups of CI's defects according to their attributes will be placed. At the end of this chapter it will be presented a list of rules that describe the behavior of the CI's defects within the CCB process.

Before carrying out the analyses of the CCB process, it is necessary to make an additional selection of the CI's defects that will be use for the analyses. It is because there is a considerable quantity of CI's defects that have not been concluded in the snapshots. Two possible solutions are suggested to this problem: (1) insert an artificial ending task at the end of the cases, so all the cases analyzed will contain an identical end task, (2) select only the cases that have been completed.

The drawback of the first option is that during the analysis, some activities will have an increased number of occurrences than others, so the results can be biased. The drawback of the second option is that taking into account only the cases that has been completed reduces the number of cases analyzed.

For the present project it was selected the second option, because having the analysis of only complete cases can provide an more accurate information about the real behavior of the CI's defects in the CCB process. Due this selection of cases, for the present analysis it will considered an event-log with 6871 cases (78.8% of the total of cases in the original event-log), which have complete information of their handling.

7.1 Analysis of the Structure of the CCB Process

It analyses the sequence of the activities in the CCB process. This analysis will be used in order to confirm or reject the preliminary concepts of the structure of the CCB process presented in figures 2.5 and 5.2. The analysis of the structure of the CCB process will consider two aspects:

- **Loops in the process**

It will analyze the existing loops within the process. A loop can be understood as the reprocessing of one or more tasks after the last executed task found a problem in the handling of the case and therefore some rework is required. The importance of detecting when this happens is because it can allow to investigate what type of attributes contain these CI's defects, and infer rules of how the structure of the process changes according to them.

A special case of loop is the Length-One-Loop (L1L) which is found in the process when a task was unsuccessfully executed so it is required to execute the same task again. The L1Ls can be identified as the sequence of the same task but with the difference that they have different time stamps, one after the other.

- **Distinguishable patterns**

An aspect that will be considered in the analysis is the definition of common patterns created by the sequences of executed tasks during the handling of the CI's defects. There are many possible sequences of tasks that a CI's defect can take during its handling; this report will consider the 20/80 rule, which in the context of the patterns means that 20% of the patterns contain the 80% of the cases.

7.2 Analysis of the Performance of the CCB Process

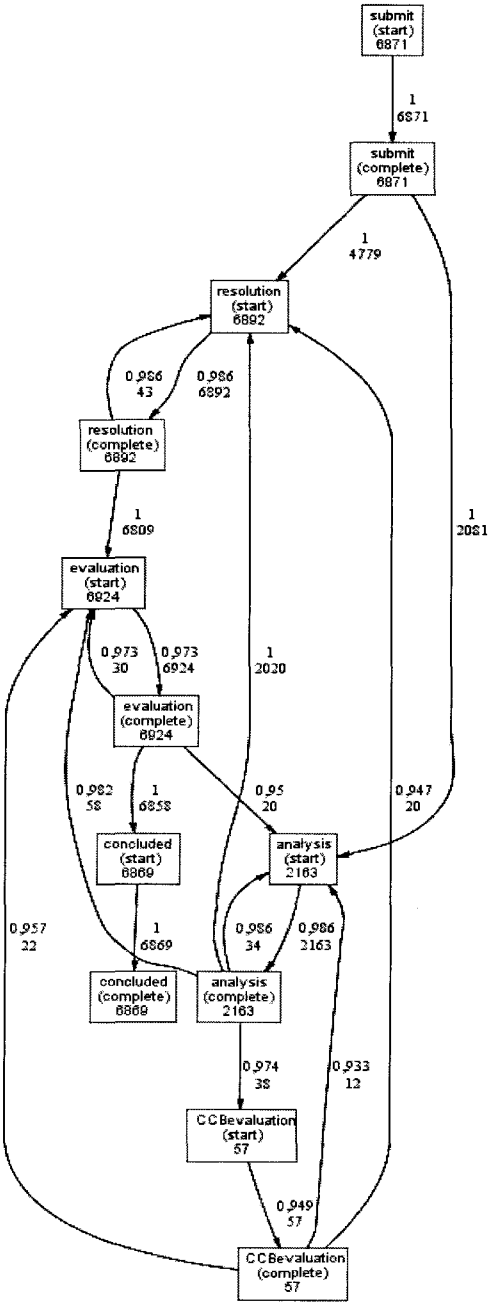
In the CCB process each of the tasks carries out a creative work that depends on skills, experience, intelligence and other factors. Because of that, there is a difference between the CCB process and traditional processes that handle "hardware" cases. In the CCB process the teams receive many CI's defects at the same time, and they pick up one by one in order to look for a solution. In case that they "come up" with the solution the team process the CI's defect immediately; however, if the team does not come up with a solution the CI's defect remains in the task until a solution arises in order to solve the problem. Then in the case of the CCB process there is a difference between the throughput time and the working time of handling a CI's defect.

The present analysis of performance will consider the throughput times of execution of the different tasks of the process. These values are taken from the difference between the dates of "start" and "complete" events of the tasks. However, it is important to highlight, the measurements given here are only references of how much time took the handling of a CI's defect in the process.

7.3 Analysis of the Full CCB Process

In this analysis, all the completed cases submitted to the CCB process are taken into account. The process model discovered with the Heuristics algorithm is shown in figure 7.1. In the following subsections this model is analyzed accordingly to its structure and performance characteristics.

Figure 7.1. CCB process taking into account all the completed cases in the data set



7.3.1 Structure of the Process

Surprisingly, the structure of the process contains some sequences that were unexpected. It was found that 70% cases (4779 of 6871) directly went from the task "submit" to "resolution" without being analyzed. In addition 4756 of these cases have not passed through the "analysis" task during their handling. From the remaining 30% cases (2092 cases of 6871) that went from "submit" to "analysis" in first instance only 61 cases did not pass through the "resolution" task later on. Table 7.1 presents the direct successors of the tasks of the CCB process.

In important observation is that the percentages of rework in the tasks fluctuate between 1 to 2 %. However, these values consider all the CI's defects in the process without making a distinction of attributes.

Table 7.1. Percentage of direct successors for the full CCB process

	Submit	Analysis	Resolution	Evaluation	CCB	Concluded	Total
Submit	0	30	70	0	0	0	100
Analysis	0	2	94	2	2	0	100
Resolution	0	0	1	99	0	0	100
Evaluation	0	0	0	0	0	100	100
CCB	0	20	40	40	0	0	100
Concluded	0	0	0	0	0	0	

In order to distinguish different patterns within the process, the Performance Sequence Diagram plug-in was used. In this diagram 51 patterns were distinguished. The most relevant patterns contain above the 80% of the CI's defects submitted are two:

1. Submit →Resolution →Evaluation →Concluded (4703 cases)
2. Submit →Analysis→Resolution →Evaluation→Concluded (1927 cases)

7.3.2 Performance Characteristics

In the analysis of performance of the full CCB process, two observations are made: the global throughput time and the individual throughput time of the tasks in the process structure. In order to observe the results obtained it is necessary to have in mind that the values are the direct representation of the data set. It is known that the tasks of the CCB process are a creative work; thus, the information about when a task starts or finishes does not provide an accurate information of how much effort was spend. For example the average throughput time of the task "Analysis" is 7.62 [days/case], and if the task would handle all the CI's defects submitted in a sequence (2163 cases analyzed), the accumulated throughput time would be 16482 days or 45 years. Thus, it is important to understand that many CI's defects are handled simultaneously in not a linear manner and their real handling takes less time than the one that it is nominally found in the data sets. Table 7.2 and 7.3 summarize the values found in the full CCB process. The calculations presented are made taking into account complete days (days of 24 hours).

Table 7.2. Performance characteristics of the full CCB process

Task	Average [days]	Min [days]	Max [days]	Std Dev [days]	Arrival time [case/day]
Submit	0.00	0.00	0.00	0.00	10.72
Analysis	7.62	0.00	346.00	21.11	3.39
Resolution	7.91	0.00	242.00	20.13	10.72
Evaluation	5.65	0.00	377.00	12.74	10.82
CCB	1.00	0.00	0.00	0.00	0.12

Table 7.3. Waiting time for the cases before a new task pick them up and process them

	Average [days]	Min [days]	Max [days]	Std Dev [days]
Case Submitted	4.67	0.00	277.00	15.36
Case Analyzed	8.63	0.00	311.00	25.00
Case Resolved	6.62	0.00	236.00	15.73
Case Evaluated	65.96	0.00	506.00	81.63
Case assessed by CCB	62.31	1.00	257.00	53.49

7.4 Analysis of the CCB Process According to their Attributes

In this section a short description is made about the behaviors of the CCB process according to the groups of attributes of the CI's defects.

7.4.1 CCB Process Model Based on CI's Attributes: Priority

The levels of priority assigned to the CI's defects are submitted at the beginning of their handling. There are three levels of priority assigned to the CI's defects in the data set: High, Medium, and Low. In this section it will be carried out an analysis of the two extremes in order to find the differences of handling in these two groups.

Figure 7.2a. CCB process of cases with high priority

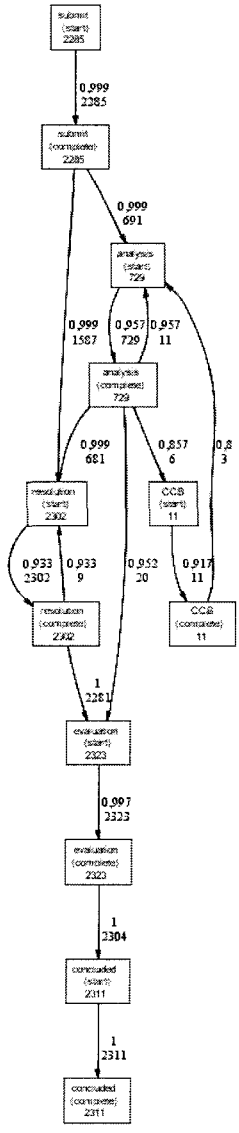
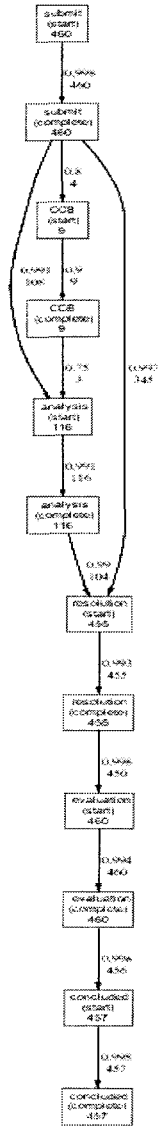


Figure 7.2b. CCB process of cases with low priority



- **Structure of the process**

The process models containing only CI's defects with "high" priority contain a much more complex structure, in which it is possible to observe that it exist reprocessing in tasks "analysis" and "resolution". The incoming and outgoing arcs of the task "CCB evaluation" are only from the task "analysis". In addition, it was found that 2% of the CI's defects in this group directly pass from "analysis" to "evaluation". On the contrary, the process model that contains only the CI's defects with "low" priority present a structure much simpler than the one with "high" priorities. The process does not contain the task "CCB evaluation", which means that there are not too critical observations to the CI's defects contained

in this process. Like in the general analysis of the CCB process, the majority of the CI's defects submitted do not execute the tasks "analysis" instead of that they directly go from "submit" to "resolution". Figures 7.2a and 7.2b show both CCB processes. In table 7.4 are presented the direct successors of the tasks of the CCB processes in this section.

Table 7.4. Percentage of direct successors for CCB processes with "high" and "low" priority [H=high/L= low]

	Submit		Analysis		Resolution		Evaluation		CCB		Concluded		Total	
	H	L	H	L	H	L	H	L	H	L	H	L	H	L
Submit	0	0	30	24	70	75	0	0	0	1	0	0	100	100
Analysis	0	0	1	1	96	98	2	0	1	1	0	0	100	100
Resolution	0	0	0	0	0	0	100	100	0	0	0	0	100	100
Evaluation	0	0	0	0	0	0	0	0	0	0	100	100	100	100
CCB	0	0	100	100	0	0	0	0	0	0	0	0	100	100
Concluded	0	0	0	0	0	0	0	0	0	0	0	0	100	100

- **Performance analysis of the process**

When comparing the arrival times of both process models, it was found that the process model containing CI's defect with "high" priority have an arrival rate equal to 10.72 cases per day; on the contrary the arrival time for the process with CI's defects with "low" priority is 0.73 cases per day. This is also in accordance with the volume of CI's defects with "high" priority (2285 cases) in comparison to the number of cases submitted with "low" priority (460 cases). Looking at the differences of global performance between both processes it was found that the CI's defects with "high" priority" are handled in an average throughput time of 94.84 days in comparison to the 131.36 days for CI's defect with "low" priority. The activities that consume more time are: "evaluation" for CI's defects with "high" priority, and "analysis" for CI's defects with "low" priority. Other interesting observation is that the resolution of a CI's defect with "high" priority takes almost three times less than the resolution of a CI's defect with "low" priority. Additionally, for CI's defects evaluated the waiting time before these cases are closed take in average 74.80 and 92.32 days respectively for "high" and "low" priority cases. These values are higher than the other average waiting times of the tasks for both types of CI's defects. Tables 7.5 and 7.6 present the performance measurements of both CCB processes.

Table 7.5 Performance characteristics [H=high/L= low]

Task	Average [days]		Min [days]		Max [days]		Std Dev [days]		Arrival time [case/day]	
	H	L	H	L	H	L	H	L	H	L
Submit	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.7	0.73
Analysis	4.4	15.1	0.0	0.0	138.0	198.0	12.9	35.4	1.1	0.9
Resolution	4.0	11.1	0.0	0.0	225.0	242.0	13.5	28.7	3.5	0.7
Evaluation	5.8	5.6	0.0	0.0	273.0	252.0	17.2	20.7	3.5	0.7
CCB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.02
Complete process	94.8	131.4	0.0	0.0	355.0	502.0	92.5	91.8	3.5	0.7

Table 7.6 Waiting time for cases before a new task pick them up [H=high/L= low]

	Average [days]		Min [days]		Max [days]		Std Dev [days]	
	H	L	H	L	H	L	H	L
Case Submitted	3.3	7.0	0.0	0.0	251.0	168.0	13.9	19.2
Case Analyzed	5.0	4.8	0.0	0.0	175.0	286.0	14.2	26.8
Case Resolved	4.7	7.5	0. 0	0.0	182.0	129.0	3.6	17.0
Case Evaluated	74.8	92.3	0.0	0.0	303.0	294.0	89.6	86.8
Case assessed by CCB	0.1	131.4	0.0	0.0	50.0	502.0	1.9	91.9

7.4.2 CCB Process Model Based on CI's Attributes: Severity

The attribute "Severity" contains five possible values which range from S (showstopper / blocker) to D (not important at all). Only the CI's defects with the high of severity A (major function affected) will be analyzed here.

- **Structure of the process**

The structure shaped by the CI's defects with a value of severity "A" shows that after the case was submitted, 51% of the cases go directly for the "resolution" tasks without being previously analyzed. In addition, it was found that only the cases that are analyzed can be rejected later on in the task "CCB evaluation". The unique task that carries out rework is "analysis" where 2% of all the cases are unsuccessfully analyzed. The remaining structure of the processing is linear it is to say that all the cases that will be "resolved" are successfully handled in the reaming tasks of the process. Figure 7.3 presents the CCB process shaped by the CI's defects with "A" level of severity, and table 7.7 presents the percentages of direct successors of tasks in the structure for this process.

Figure 7.3. CCB process of cases with "A" Severity

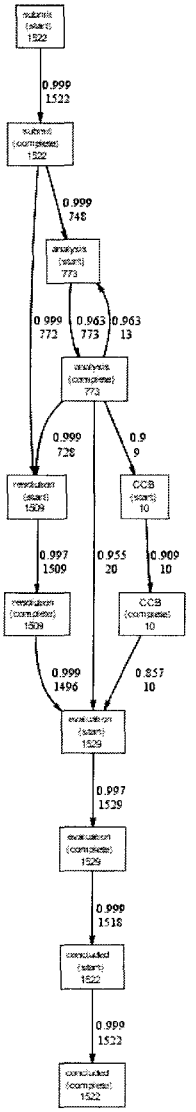


Table 7.7 Percentage of Direct successors for the CCB process with "A" level of severity

	Submit	Analysis	Resolution	Evaluation	CCB	Concluded	Total
Submit	0	49	51	0	0	0	100
Analysis	0	2	95	2	1	0	100
Resolution	0	0	0	100	0	0	100
Evaluation	0	0	0	0	0	100	100
CCB	0	0	0	0	100	0	100
Concluded	0	0	0	0	0	0	0

- **Performance analysis of the process**

The total number of CI's defects with severity level "A" is 1522 with an arrival time of 2.3 cases per day. The average time of processing a CI's defect is 73 days, with a maximum time of 310 days. The task that consumes the most time during the processing in average is the "evaluation". Like in the general processing of the CI's defects the time that takes close a case in the process is the one that takes more time, for this type of CI's defects the waiting time for the "concluded" task is 51.5 days. Tables 7.8 and 7.9 show the performance characteristics of this CCB process.

Table 7.8 Performance characteristics

Task	Average [days]	Min [days]	Max [days]	Std Dev [days]	Arrival time [case/day]
Submit	0.0	0.0	0.0	0.0	2.4
Analysis	4.7	0.0	197.0	13.1	1.2
Resolution	3.4	0.0	203.0	12.0	2.3
Evaluation	5.5	0.0	221.0	13.4	2.4
CCB	0.0	0.0	0.0	0.0	0.04
Complete process	73.0	1.0	310.0	79.9	2.3

Table 7.9 Waiting time for cases before a new task pick them up

	Average [days]	Min [days]	Max [days]	Std Dev [days]
Case Submitted	3.4	0.0	145.0	10.1
Case Analyzed	5.2	0.0	176.0	15.5
Case Resolved	4.3	0.0	142.0	10.5
Case Evaluated	51.5	0.0	300.0	76.4
Case assessed by CCB	4.3	0.0	142.0	10.5

7.4.3 CCB Process Model Based on CI's Attributes: Request type

For the analysis of the CI's defects in the process according to their type of request, three process models were generated, each one for the specific type of request that can be found in CCB. It was found that 57% of the CI's defects are submitted because they need to be fixed (PR), 35.5% are submitted because an implementation is required (IR), and 7.5% of them require a change (CR).

- **Structure of the process**

Each of the types of requests has a different process model. The main difference among them is the sequence that involves the tasks "CCB evaluation". For the request type PR all the cases that have been evaluated by the CCB board come from the "analysis" task and later are redirected to the "evaluation" task. In the case of the CI's defects requesting an implementation, when the "CCB evaluation" is executed, they come directly from the "submit" task and are redirected to "resolution" task later on. In case of CI's defects requesting for a change the "CCB evaluation" task the incoming and outgoing arcs are linked to the "analysis task. The structures also present evidence of rework, the "analysis" task has a rework of 2% in PR cases and 3% in CR cases. Other task like "resolution" and "evaluation" required also rework but in fewer cases. Figures 7.4a, 7.4b, and 7.4c show the three different CCB process shaped by the requests types; in addition, table 7.10 presents percentages of direct successors between tasks in these three processes.

Figure 7.4a CCB process of only cases with Problem Request (PR)

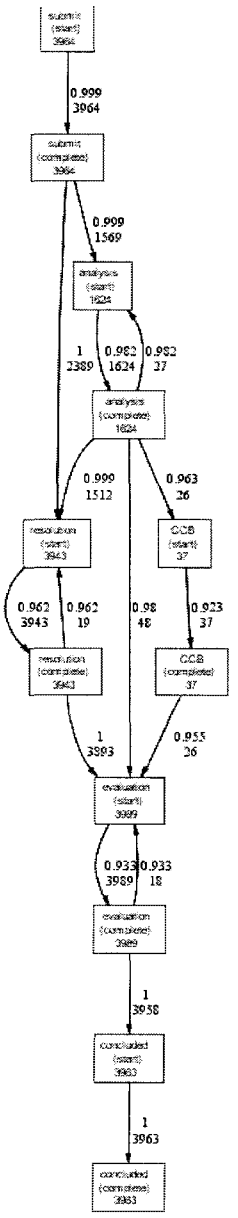


Figure 7.4b CCB process of only cases with Change Request (CR)

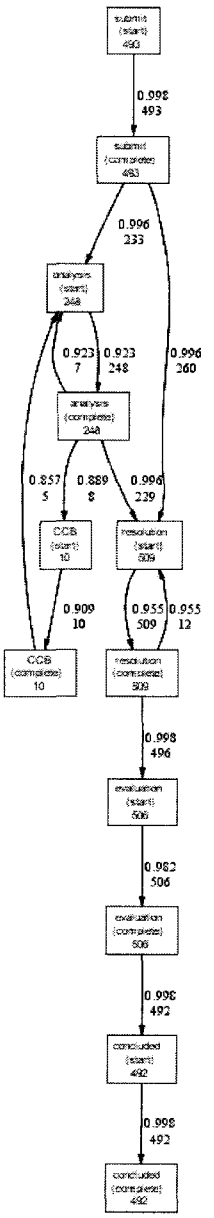


Table 7.4c CCB process of only cases with Implementation Requests (IR)

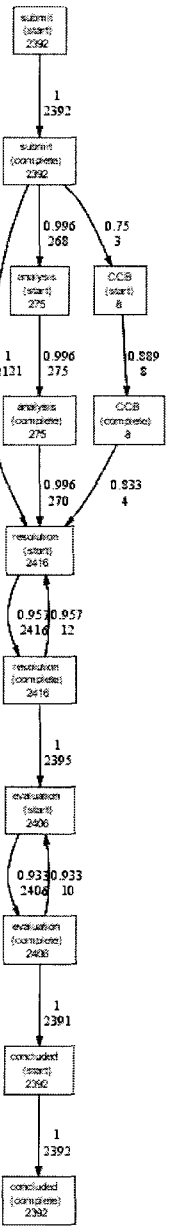


Table 7.10. Percentage of Direct successors for CCB processes with different request types [Problem request = PR/Change request = CR/Implementation Request = IR]

	Submit			**Analysis**			**Resolution**			**Evaluation**			**CCB**			**Concluded**			**Total**		
	PR	CR	IR	PR	CR	IR	PR	CR	IR	PR	CR	IR	PR	CR	IR	PR	CR	IR	PR	CR	IR
Submit	0	0	0	40	47	10	60	53	89	0	0	0	0	0	1	0	0	0	100	100	100
Analysis	0	0	0	2	3	0	93	94	100	3	0	0	2	3	0	0	0	0	100	100	100
Resolution	0	0	0	0	0	0	0.05	2	0.5	99.5	98	99.5	0	0	0	0	0	0	100	100	100
Evaluation	0	0	0	0	0	0	0	0	0	0.5	0	0.5	0	0	0	99.5	100	99.5	100	100	100
CCB	0	0	0	0	100	0	0	0	100	100	0	0	0	0	0	0	0	0	100	100	100
Concluded	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **Performance analysis of the process**

The arrival times for the different type of CI's defects are 6 cases per day for the PR request, 0.8 cases per day for the CR request, and 3.8 cases per day for the IR request. From the three types of requests, the Implementation request has the longer processing time with an of average 118.7 days with respect to 82.2 days for Problem request and 86.6 days for Change request. However, because of the high variability of processing the cases the maximum time that took completing the handling of a process was found in a PR request type with 502 days. This value does not mean that it was necessary 502 days of work in the CI's defect, but it is the cumulative time that took a case since it was submitted until its closing. Like in all the cases observed the waiting time before the execution of the "conclude" task is considerably big that values also blow the final results. Tables 7.11 and 7.12 show the performance characteristics of these three different CCB processes.

Table 7.11. Performance characteristics [Problem request = PR/Change request = CR/Implementation Request = IR]

Task	Average [days]			Min [days]			Max [days]			Std Dev [days]			Arrival time [case/day]		
	PR	CR	IR	PR	CR	IR	PR	CR	IR	PR	CR	IR	PR	CR	IR
Submit	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.2	0.8	3.8
Analysis	7.4	8.7	7.1	0.0	0.0	0.0	346.0	182.0	148.0	20.6	26.4	17.4	2.5	0.4	0.4
Resolution	4.7	10.8	10.1	0.0	0.0	0.0	203.0	150.0	242.0	14.7	20.6	25.2	6.1	0.8	3.8
Evaluation	6.0	9.5	4.2	0.0	0.0	0.0	377.0	252.0	200.0	17.7	24.7	14.9	6.1	0.8	3.8
CCB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.02	0.0
Complete process	82.2	86.6	118.7	0.0	5.0	0.0	502.0	398.0	447.0	80.9	77.4	93.6	6.0	0.8	3.8

Table 7.12. Waiting time for cases before a new task pick them up [Problem request = PR/Change request = CR/Implementation Request = IR]

	Average [days]			Min [days]			Max [days]			Std Dev [days]		
	PR	CR	IR	PR	CR	IR	PR	CR	IR	PR	CR	IR
Case Submitted	3.7	2.9	6.2	0.0	0.0	0.0	188.0	181.0	236.0	11.1	15.0	18.0
Case Analyzed	5.9	19.4	1.4	0.0	0.0	0.0	176.0	286.0	196.0	16.5	43.7	10.3
Case Resolved	4.4	8.3	9.5	0.0	0.0	0.0	183.0	197.0	236.0	10.5	18.6	20.0
Case Evaluated	58.1	41.5	86.3	0.0	0.0	0.0	302.0	294.0	303.0	76.1	59.8	91.1
Case assessed by CCB	0.2	0.8	1.4	0.0	0.0	0.0	135.0	69.0	186.0	3.3	6.2	10.3

7.4.4 CCB Process Model Based on the Teams

As it was argued in section 2.8.3, it is possible that different teams can execute in different manner the CI's defect under their responsibility. Here three different teams were selected according to the amount of CI's defects that they are in charge on. There is a total of 24 teams. The first team is in charge of handling 2459 CI's defects, the second is in charge of 1417 CI's defects, and the third one is in charge of 1122 CI's defects.

- **Structure of the process**

After generating the process models for the three teams, it is possible to see that the three structures are different. The main difference among these three process models is the execution of the task after "submit", and before reaching the task "evaluation". What is interesting to see in these three processes is that none of them have unsuccessfully executed tasks. This means that teams with less CI's defects in their charge are responsible for unsuccessful executions of tasks within the process. Figures 7.5a, 7.5b, and 7.5c present the different process models created by the three teams. In addition, table 7.13 presents the direct successors between tasks in the processes created.

Figure 7.5a CCB process of cases handled by team 1

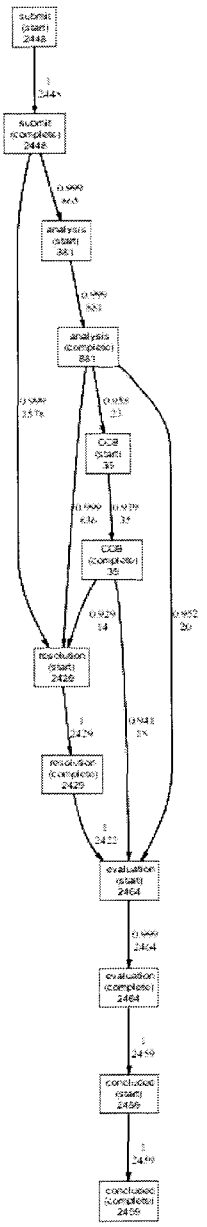


Figure 7.5b CCB process of cases handled by team 2

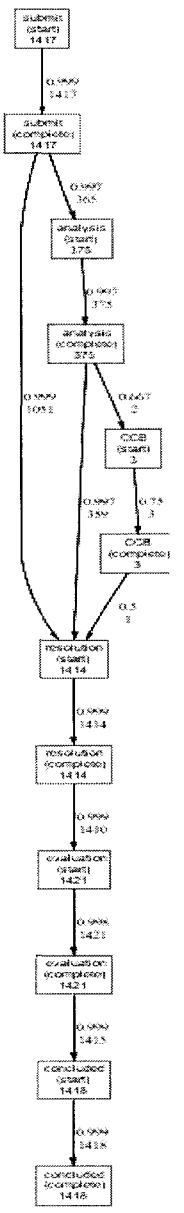


Figure 7.5c. CCB process of cases handled by team 3

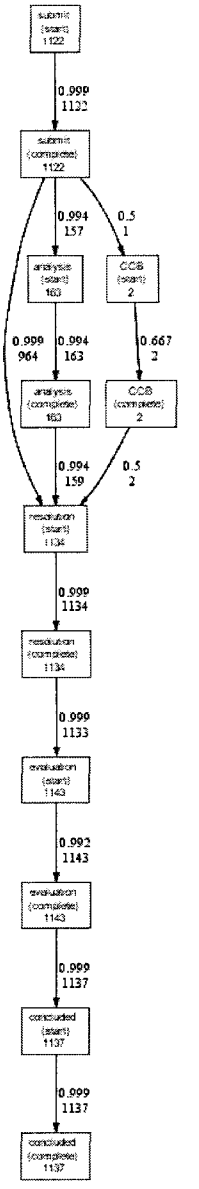


Table 7.13. Percentage of Direct successors for CCB processes by executor teams [Team 1 = T1/Team 2 = T2/ Team 3= T3]

	Submit			**Analysis**			**Resolution**			**Evaluation**			**CCB**			**Concluded**			**Total**		
	T1	T2	T3	T1	T2	T3	T1	T2	T3	T1	T2	T3	T1	T2	T3	T1	T2	T3	T1	T2	T3
Submit	0	0	0	35	26	14	65	74	85.5	0	0	0	0	0	0.5	0	0	0	100	100	100
Analysis	0	0	0	0	0	0	95	99.5	100	2	0	0	3	0.5	0	0	0	0	100	100	100
Resolution	0	0	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	100	100	100
Evaluation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	100	100	100	100	100
CCB	0	0	0	0	0	0	48	100	100	52	0	0	0	0	0	0	0	0	100	100	100
Concluded	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **Performance analysis of the process**

The arrival rate for the first team is 4.2 cases per day, for the second 2.4 cases per day, and for the third is 2 cases per day. The average throughput times for the three teams are 98.7, 112.6, and 102.4 days per case respectively. This means that even if the first team receives more cases per day it solves them slightly faster than the other two teams. However, it is important to make a more in-deep evaluations of the type of cases handled by each team in order to understand this behavior. Tables 7.14 and 7.15 show the performance characteristics of these CCB process models.

Table 7.14. Performance characteristics [Team 1 = T1/Team 2 = T2/ Team 3= T3]

Task	Average [days]			Min [days]			Max [days]			Std Dev [days]			Arrival time [case/day]		
	T1	T2	T3	T1	T2	T3	T1	T2	T3	T1	T2	T3	T1	T2	T3
Submit	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.2	2.4	2.0
Analysis	5.1	7.7	6.1	0.0	0.0	0.0	198.0	138.0	112.0	14.2	16.4	14.8	1.5	0.7	0.3
Resolution	6.6	4.4	7.2	0.0	0.0	0.0	157.0	242.0	225.0	15.8	17.0	22.5	4.2	2.5	2.0
Evaluation	4.7	4.3	7.3	0.0	0.0	0.0	134.0	273.0	194.0	12.3	13.9	21.3	4.2	2.5	2.0
CCB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.05	0.05
Complete process	98.7	112.6	102.4	1.0	0.0	2.0	468.0	432.0	346.0	89.2	91.9	84.3	4.2	2.4	2.0

Table 7.15. Waiting time for cases before a new task pick them up [Team 1 = T1/Team 2 = T2/ Team 3= T3]

	Average [days]			Min [days]			Max [days]			Std Dev [days]		
	T1	T2	T3	T1	T2	T3	T1	T2	T3	T1	T2	T3
Case Submitted	3.5	6.5	2.7	0.0	0.0	0.0	190.0	168.0	108.0	11.9	14.7	12.0
Case Analyzed	6.4	6.6	1.2	0.0	0.0	0.0	228.0	228.0	228.0	17.4	20.5	10.7
Case Resolved	4.6	10.2	4.8	0.0	0.0	0.0	127.0	236.0	98.0	8.8	21.8	6.2
Case Evaluated	74.9	83.6	78.6	0.0	0.0	0.0	303.0	300.0	288.0	85.7	90.0	84.6
Case assessed by CCB	57.8	0.02	78.6	1.0	0.0	0.0	257.0	26.0	288.0	56.3	0.7	84.6

7.4.5 CCB Process Model Based on CI's Defects and their Phase

By using the attribute "Caused during" it is possible to allocate the CI's defects to the phases of the V-shaped structure of the SDP of the MTR-A project. The phases and sub-phases found in the data set are seven: Requirements (511 cases), Design (903 cases), Scenarios (96 cases), Architecture (180 cases), Implementation (2176 cases), Integration testing (680 cases), and Testing (302 cases). Unfortunately, 1638 CI's defects do not contain information about the phases, so the total number or cases for the analysis is 5233 cases. Below is the description of the Design, Implementation, and Integration phases.

- **Structure of the processes**

Each of the phases and sub-phases present different shapes in the process models generated. Here a description of these structures is presented.

During the Design phase, once the CI's defect is submitted 25% of the cases are redirected to the "analysis" task, from there, 1% of the cases are evaluated by the CCB. All the cases evaluated by the CCB are redirected to the task "evaluation" and later "concluded".

The process shaped by the CI's defects from the Implementation, as it is expected, contains the biggest amount of CI's defects, in this phase the task "submit" to "resolution" happens in 67% of the cases. The remaining 33% go to "analysis" and 1% of the analyzed cases are redirected to a "CCB evaluation". The task "resolution" receives the 100% of the cases coming from "submit", "analysis" or "CCB evaluation". In addition, there is rework in 0.5% of the cases submitted to the process during this phase in the "resolution" task.

The Integration phase is similar to the shape of the Implementation phase; however, the percentages of how the CI's defects are distributed are different. The sequence "submit" to "analysis" happens 24% of

the cases. From "analysis" to "CCB evaluation" happens 2% of the cases. The rework of the "resolution" task is 0.5%. Figures 7.6a, 7.6b, and 7.6c show the three processes analyzed in this section. In table 7.16 are presented the percentages of directs successors between tasks in each of the CCB processes of this section.

Figure 7.6a. CCB Process with CI's defects submitted during the Design phase

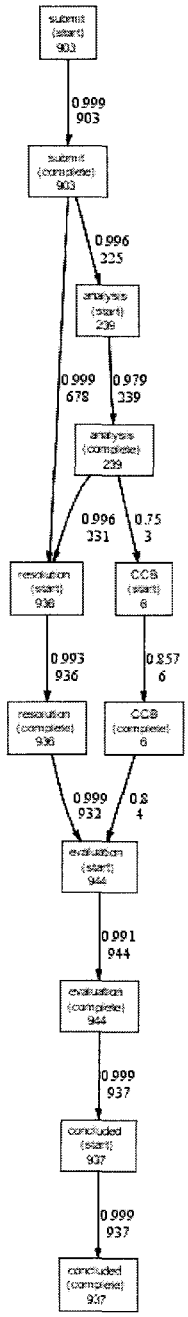


Figure 7.6b. CCB Process with CI's defects submitted during the Implementation phase

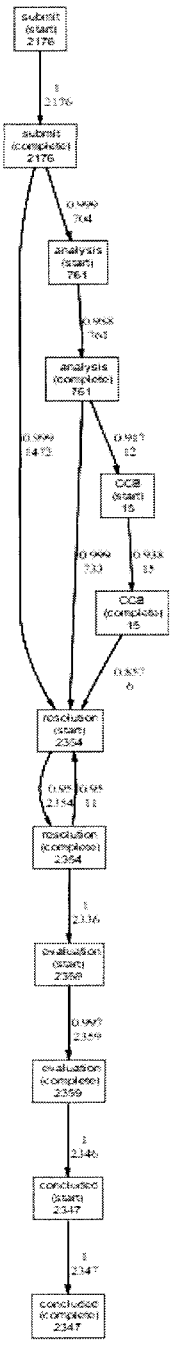


Figure 7.6c. CCB Process with CI's defects submitted during the Integration testing phase

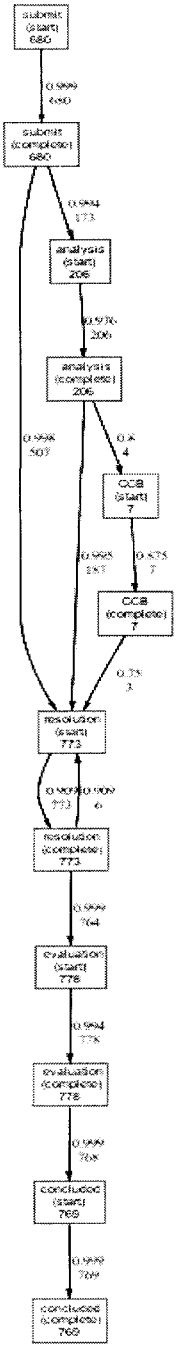


Table 7.16. Percentage of Direct successors for CCB processes by phases [Design = D/Implementation = IM/Integration = IN]

	Submit			Analysis			Resolution			Evaluation			CCB			Concluded			Total		
	D	IM	IN	D	IM	IN	D	IM	IN	D	IM	IN	D	IM	IN	D	IM	IN	D	IM	IN
Submit	0	0	0	25	33	26	75	67	74	0	0	0	0	0	0	0	0	0	100	100	100
Analysis	0	0	0	0	0	0	99	99	0	0		98	1	1	2	0	0	0	100	100	100
Resolution	0	0	0	0	0	0	0	0.05	0.05	100	99.5	99.5	0	0	0	0	0	0	100	100	100
Evaluation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	100	100	100	100	100
CCB	0	0	0	0	0	0	0	100	100	100	0	0	0	0	0	0	0	0	100	100	100
Concluded	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	100	100

- **Performance analysis of the process**

The arrival times are 1.39 cases per day during the Design phase, 3.35 cases per day during the Implementation phase, and 1.06 cases during the Integration phase. With respect to the throughput times they are also different depending on the phases. The total average throughput time for cases in the Design phase is 85.15 days, for the Implementation phase about 94.38 days and for the Integration phase 64.29 days. Tables 7.17 and 7.18 present the performance characteristics of these three CCB process models.

Table 7.17. Performance characteristics [Design = D/Implementation = IM/Integration = IN]

Task	Average [days]			Min [days]			Max [days]			Std Dev [days]			Arrival time [case/day]		
	D	IM	IN	D	IM	IN	D	IM	IN	D	IM	IN	D	IM	IN
Submit	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4	3.4	1.1
Analysis	2.9	3.5	2.7	0.0	0.0	0.0	66.0	197.0	56.0	8.6	13.3	6.7	0.4	1.1	0.3
Resolution	8.0	4.3	5.5	0.0	0.0	0.0	150.0	144.0	242.0	17.8	13.0	19.4	1.4	3.4	1.08
Evaluation	7.4	5.0	5.7	0.0	0.0	0.0	194.0	181.0	221.0	20.84	13.8	14.3	1.4	3.4	1.1
CCB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.02	0.01
Complete process	85.2	94.4	64.3	1.0	0.0	1.0	307.0	468.0	294.0	82.83	88.9	66.37	1.4	3.4	1.1

Table 7.18. Waiting time for cases before a new task pick them up [Design = D/Implementation = IM/Integration = IN]

	Average [days]			Min [days]			Max [days]			Std Dev [days]		
	D	IM	IN	D	IM	IN	D	IM	IN	D	IM	IN
Case Submitted	3.2	2.7	2.5	0.0	0.0	0.0	137.0	190.0	69.0	10.5	10.5	6.9
Case Analyzed	7.3	8.4	6.2	0.0	0.0	0.0	110.0	286.0	175.0	17.5	27.7	19.2
Case Resolved	5.3	4.9	4.7	0.0	0.0	0.0	229.0	230.0	183.0	13.38	12.3	12.5
Case Evaluated	58.8	73.8	44.1	0.0	0.0	0.0	302.0	303.0	294.0	78.9	86.21	61.85
Case assessed by CCB	5.3	0.2	0.1	0.0	0.0	0.0	229.0	257.0	69.0	13.38	6.0	2.7

7.5 Patterns Discovered

Each of the attributes used during the generation of the models shape the CCB structure in different manner. However, it is possible to make a list of common patterns and behaviors they presented below:

- The most used sequence is: Submit → Resolution → Evaluation →Concluded
- All the cases after being submitted can be redirected to Analysis, Resolution, or CCB evaluation, but they never go directly to evaluation or concluded.
- The rework of the tasks is not common, ranging from 0.5 to 2% of the cases.
- In general all the throughput times presented contain high variability of execution with high standard deviations.
- There is always a long waiting time before the cases are concluded

7.6 Post-Experiments Evaluation

Taking into account the description of the different process models generated and the general patterns described in the previous section. It is important to make the following evaluation.

The generation of the event-log is based on a set of snapshots coming from the middle time of execution of the MTR-A project. This means that there are already finished cases since the first snapshot. They are not being handled anymore within the CCB process, so they present a linear structure of the process in all the snapshots used for the experiment. Because the program used for creating the event-log is based on the changes of dates in the task's events of the snapshots, all the concluded CI's defects will remain untouched in the event-log showing only a linear structure of their handling. Thus, the results can be biased by the influence of these CI's defects presenting a more linear structure of the CCB process than how it in reality is. The solution for solving this problem is to analyze CI's defects that have started their handling in the first snapshot, so it is possible to see its real handling by the CCB process.

Other observation is to the results of the analysis is that there is not a parallelism between tasks. There are two factors for that: (1) the creation of artificial task's events that complete missing values in the data set, (2) the manual filling in of the information in the snapshots force to have a linear sequence of tasks avoiding the likelihood of carrying out tasks in parallel. The solution for the first factor is to improve the program in order to carry out the analysis not only taking into account individual task's events but also the interdependence of these events in the task that they belong. The solution of the second factor requires that the personnel in charge of handling the CCB process understands the structure of the process and the possible constructs that can happen during the handling of CI's defects.

Chapter 8

8 Conclusions and Further Directions

The present project has been focused on provide more insight to the understanding of the CCB process within the MTR-A software development project. In order to achieve that, Process Mining techniques have been used through an experiment that applies what has been called in this project "Process Mining framework", which is a list of sequential steps that help to carry out the Process Mining experiment. These steps have been organized in three phases within the framework: Preparation phase, Pattern Discovery phase, and Pattern Analysis phase.

During the application of the Preparation phase, it was found that characteristics of the raw data material are of supreme importance in order to appropriately extract information that accurately describes the process model analyzed. This phase is the most difficult and time consuming of the three phases. This phase of the Process Mining framework lasted until the last weeks of work for the present project. Looking at the results obtained from this phase, it is argued that the outcome of the Preparation phase is the translation of the raw data material in adequate process elements by applying the understanding of the CCB process. This work has to be applied to all the cases in the data set, which means that it is required to have an understanding of all the possible routes or sequences that a CI's defect can follow. This work would be time consuming if it is manually carried out in the 8714 cases. Thus, the Preparation phase involves the construction of a small program that transformed automatically the raw data material in a suitable event-log with all the required information for carrying out the next phase of the Process Mining framework. After the experience given by the present project it is argued that three factors influence a good creation of an event-log: (1) the understanding of the process,(2) the type of information that can be found in the data set used during the experiment, and (3) the capability to create an algorithm that correctly translates the raw data set in a suitable event-log that really describes the process.

In order to execute the Pattern Discovery phase and Pattern Analysis phases, it was employed a software tool called ProM (Process Mining tool) that was in charge to automatically read the event-log, generate a process model for the CCB, and provide performance calculations about the tasks found in the process. The software tool provides all the functionalities required for carrying out these two phases. However, the work of linking the Preparation phase and the next two phases needs some coordination. It is to say that the event-log produced needs to fulfill some specific requirements in its structure and content. This fact validates the two-way connection between phases in the Process Mining framework created. It's important to remark that ProM is an experimental software tool developed by the Eindhoven University of Technology, and not a commercial software tool which provides an easy-to-use interface.

The data set used for the present project contained specific characteristics, and no previous developed programs or plug-ins existed in order to deal with these characteristics. The content of the data set was designed for an analysis with classical Software Engineering techniques. The main tasks of the project were to understand and adapt the data elements to a process-meaning structure of data elements. Here the challenges were to identify the useful data elements, assign them a specific role and deal with their behavior in the data set. Two main problems were overcome in the present project: (1) the deletion of unsuccessful executions of tasks during the handling of a CI's defect, and (2) the use of information about states and task's events which come from a unique data element in the data set.

The analysis of the CCB process not only provided information about the possible sequence of the tasks, but also identified the rework carried out in some of the tasks of the CCB process. The discovered process

models presented a complex structure even though they just contain six tasks. Surprisingly, it was found that a majority of the CI's defects after being submitted were redirected to the "resolution" task and not to the "analysis". This fact completely changes the initial conception that the process is composed by a stiff linear sequence of tasks.

The Pattern Discovery phase studied isolated groups of CI's defects according to their attributes. Each of the groups presented different sequences of pattern and performance activities values. Thus, with the results obtained after the execution of the three phases of the Process Mining framework, it is possible to validate the assumption that the structure and performance of the CCB process change according to the type of attributes of the CI's defects.

Besides the positive results obtained during the experiment carried out in the project. Some observations were found at the moment to analyze the results. With the present approach it was not possible to find a parallel execution of activities. Additionally, the standard deviations of the execution of the tasks present big values. Two reasons can produce these results: (1) the attributes do not clearly represent the properties of the CI's defects submitted and the groups of CI's defects are still too heterogeneous, or (2) the information of the data set used for delimiting the tasks and the calculation of the throughput times is not accurately transformed.

It is argued that an accurate study of the process is possible only when there is a correct measurement of the execution of the tasks for each CI's defect. The further research has to be focused on improving the observations made in each of the phases of the experiment. Here it is proposed a list of activities that should be taken into consideration in order to improve the results:

- An increase on the understanding of the CCB process
    - Understanding about the tasks that evaluate the work within the CCB process
    - Understanding the tasks in detail and identify the factors that affect their performance.
        - Include technical measurements for individual tasks
        - Identification of the substructure of the tasks. It is say to take the tasks as sub-processes with their own process structure.
    - Understanding of the CI's attributes and relate them to the CI's defects
- An improvement of the content of the data sets
    - Add information of the status of the cases and the status of their attributes as well.
    - Include information that relates cases, for instance in case to find duplicated request, it would be useful to know the other case that requested the same change.
    - Standardization of the terms used in order to describe the elements that describe the components of the process and attributes that describe the CI's defects
    - Standardization of the data elements with respect to other data sets coming from other entities within the organization.
    - Add technical attributes related to tasks that handle the CIs as it was requested previously
    - Add CI's attributes as part of the CI's defects attributes for the analysis of the process.

In chapter 2 it was argued that an analysis of the CCB should not only take into account the CI's defects, but also the processes that handle them. Moreover, an integral understanding of the processes of each phase of the Life cycle model would be preferable before any further analysis of the individual software elements. With a clearer understanding of the processes, the measurements that for long time have been neglected are possible, for instance the measurement of the effort.

Finally, Process Mining techniques are positively applied in processes like the CCB process, even if they still have some limitations. These techniques have helped to overcome the initial paradigm that it is not possible to carry out an analysis of process in the software production.

9 References

[1] N. Whitten. Managing Software Development Projects: Formula for Succes. Chichester. Second Edition: Wiley, 1995

[2] P.N. Robillard. The Role of Knowledge on Software Development. *Comunications of the ACM*. Vol. 42. - 87-92. 1999.

[3] S.L. Fenton and N.E. Pfleeger Software Metrics - A Rigorous & Practical Approach. Second Edition. Cambridge : Thompson Computer Press 1997.

[4] F.P. Brooks. The Mythical man-month. *Information Cahners Business*.- pp. 129-136. Datamation.1974.

[5] M.C.Paulk, B. Curtis, M.B. Chrissis, C.V.Weber. The capability maturity model. Software Engineering Institute. 1991

[6] B. Meyer. Dependable Software. *Lecture Notes in Computer Science*. Vol. 4028. - pp. 1-33. 2006.

[7] B. Westfechtel and R. Conradi. Software Architecture and Software Configuration Management . *Lecture notes in computer science*. Vol. 2649/2003. - pp. 24-39. 2003

[8] Institute of Electrical and Electronics Engineers. IEEE Standard Glossary of Software Engineering Terminology . 1990

[9] D. Leffinwell and D. Widrig. Managing Software Requirements - A Unified Approach. New Jersey : Addison Wesley Longman, 2000

[10] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters, Workflow mining: A survey of issues and approaches. *Data & Knowledge Engineering*. Volume 47, 237 – 267. 2003

[11] A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros. Process Mining with the Heuristics Miner-algorithm. *BETA Working Paper Series*, Eindhoven University of Technology, Eindhoven, 2006.

[12] W.M.P. van der Aalst and A J M M Weijters Process Mining: A Research agenda . *Computers in Industry*. Vol. 53. - pp. 231 - 244. Elsevier, 2004

[13] G. Kassem, and C. Rautenstrauch. Problem of Tracing Workflow Instances in ERP-Systems Information Management in Modern Enterprise*: Issues & Solutions, Proceedings of The 2005 International Business Information Management Conference*, 123–131. 2005

[14] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M. Shan. Business Process Intelligence. *Computers in Industry*. Volume 53, 321 – 343, 2004

[15] J.J.E. Ingvaldsen and J.A. Gulla, Model Based Business Process Mining. *Information Systems Management*. Volume 23, Issue 1, 2006

[16] I.H. Witten and E. Frank. Data Mining: Practical Machine Learing Tools and Techniques. San Fransisco CA. Second Edition. Morgan Kaufmann Publishers, 2005

[17] W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters et al. Business process mining: An industrial application . *Information Systems*, Vol. 22. - pp. 713-732. 2006

[18] M. Bianchi, M. Maggini and L. Sarti. From Flat to Structural Pattern Recognition. *Advances in Imaging and Electron Physics*. Auth. Hawkes P W.- San Diego CA : Elsevier, - Vol. 140. 2006

[19] W.M.P. van der Aalst and A J M M Weijters. Process Mining .*Process Aware Information System*.Auth. Dumas M, Aalst W P M van der and Hoefstede : John Wiley & Sons, 2005

A. First attempt program

```
<?PHP
        include "bdd.inc";
        include "var.php";

                $tableName  = 'data';

if($bsend=="Execute"){
mysql_query('TRUNCATE TABLE `data`');
                //$csvfile = '../../www/edy/files/';

//READ FILES'S DIRECTORY-----------------------------------
        $dir = "c:/wamp/www/edy/files/";
        // Open a known directory, and proceed to read its contents
        if (is_dir($dir)) {
        if ($dh = opendir($dir)) {
                while (($file = readdir($dh)) !== false) {
                //echo "filename: $file : filetype: " . filetype($dir . $file) . "\n";
                                if($file!="." and $file!=".."){
                                        $csvfile=$dir.$file;
                                        echo "<br>".$file;
                                        $loadsql  =  'LOAD  DATA  INFILE  "'.$csvfile.'"  INTO  TABLE
'.$tableName.' FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY """ LINES TERMINATED BY "\r"
IGNORE 1 LINES';
                                        mysql_query($loadsql) or die(mysql_error());
                                        print mysql_error();

                                }

                }
                        //mysql_query("delete    from    data    where    CONVERT('DataSet'    USING
utf8)='\n'");
                        //print mysql_error();

                closedir($dh);
        }
        }
        else{echo "The directory does not exist";}
//--------------------------------------------
//CREATE XML FILE------------------------------
//--------------------------------------------

$head="<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>
<WorkflowLog                                    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"http://is.tm.tue.nl/research/processmining/WorkflowLog.xsd\">
        <Data>
                <Attribute name=\"app.name\">Name of the Program</Attribute>
  </Data>
        <Source program=\"Thesis\"/>
        <Process id=\"MTRA\" >";

$foot="</Process>
</WorkflowLog>";
        $ridBefore='';
        //BODY---------------
        $fp = fopen("ccb.xml", "w+");
                $sql=mysql_query("select  *  from  data  group  by  Problem_number,Subsystem,Crstatus
order by Problem_number,Subsystem");
                $numrrd=mysql_num_rows($sql);
                if($numrrd>0){
                        $numlrrd=0;
                        while($numlrrd<$numrrd){
                                $rid=mysql_result($sql,$numlrrd,"Problem_number")."-
".mysql_result($sql,$numlrrd,"Subsystem");
                                $rDataSet=mysql_result($sql,$numlrrd,"DataSet");
                                $rHistoryDate=mysql_result($sql,$numlrrd,"HistoryDate");
```

```
                        $rSystem=mysql_result($sql,$numlrrd,"System");

                        $rProduct_name=mysql_result($sql,$numlrrd,"Product_name");
                        $rProduct_subsys=mysql_result($sql,$numlrrd,"Product_subsys");
                        $rVersion=mysql_result($sql,$numlrrd,"Version");
                        $rRelease=mysql_result($sql,$numlrrd,"Release");
                        $rPriority=mysql_result($sql,$numlrrd,"Priority");
                        $rSeverity=mysql_result($sql,$numlrrd,"Severity");
                        $rDefect_type=mysql_result($sql,$numlrrd,"Defect_type");
                        $rProblem_type=mysql_result($sql,$numlrrd,"Problem_type");
                        $rRequest_type=mysql_result($sql,$numlrrd,"Request_type");
                        $rCrstatus=mysql_result($sql,$numlrrd,"Crstatus");

                        $rin=substr($rCrstatus,0,2);
                        $red=substr($rCrstatus,-2);
                        if($rin=='in'){$rEventType="start";}
                        if($red=="ed"){$rEventType="complete";}

                        $rCaused_during=mysql_result($sql,$numlrrd,"Caused_during");

        $rDiscovered_during=mysql_result($sql,$numlrrd,"Discovered_during");
                        $rAct_total_eff=mysql_result($sql,$numlrrd,"Act_total_eff");
                        $rCreate_time=mysql_result($sql,$numlrrd,"Create_time");
                        $rSubmitted_time=mysql_result($sql,$numlrrd,"Submitted_time");
                        $rIn_analysis_time=mysql_result($sql,$numlrrd,"In_analysis_time");
                        $rAnalysed_time=mysql_result($sql,$numlrrd,"Analysed_time");

        $rIn_resolution_time=mysql_result($sql,$numlrrd,"In_resolution_time");
                        $rResolved_time=mysql_result($sql,$numlrrd,"Resolved_time");

        $rIn_evaluation_time=mysql_result($sql,$numlrrd,"In_evaluation_time");
                        $rEvaluated_time=mysql_result($sql,$numlrrd,"Evaluated_time");
                        $rModify_time=mysql_result($sql,$numlrrd,"Modify_time");
                        $rModifiable_in=mysql_result($sql,$numlrrd,"Modifiable_in");
                        $rDiscovered_on=mysql_result($sql,$numlrrd,"Discovered_on");
                        $rProgram=mysql_result($sql,$numlrrd,"Program");
                        $rTeam=mysql_result($sql,$numlrrd,"Team");
                        $rScope=mysql_result($sql,$numlrrd,"Scope");

                        //echo "<br>".$rDataSet."";

                $body=$body;

                if($rid!=$ridBefore){
                if($ridBefore!=''){$body .="\n</ProcessInstance>";}
                $body .= "\n<ProcessInstance id=\"$rid\" description=\"\">
                <Data>
                        <Attribute name=\"Product_name\">$rProduct_name</Attribute>
                        <Attribute name=\"Release\">$rRelease</Attribute>
                        <Attribute name=\"Priority\">$rPriority</Attribute>
                        <Attribute name=\"Severity\">$rSeverity</Attribute>
                        <Attribute name=\"Request_type\">$rRequest_type</Attribute>
                        <Attribute name=\"Discovered_during\">$rDiscovered_during</Attribute>
                </Data>";
                }
                        $body .= "
                        <AuditTrailEntry>
                        <WorkflowModelElement>$rCrstatus</WorkflowModelElement>
                        <EventType >$rEventType</EventType>
                        <Timestamp>$rHistoryDate</Timestamp>
                        <Originator>$rTeam</Originator>
                        </AuditTrailEntry>";



                        $ridBefore=$rid;
                        $numlrrd++;
                }
        }
$body .="\n</ProcessInstance>\n";
        $cxml=$head.$body.$foot;
```

```
        fputs($fp,$cxml);
}
?>
<form name="form1" method="post" action="load.php">
<table width="90%"  border="1">
  <tr>
    <td >
      <input type="submit" name="bsend" value="Execute">
    </td>
  </tr>
</table>
</form>
```

B. Sequence of the 2714 case in the snapshots

History Date	Subsystem	Problem number	Priority	Severity	Request type	Crstatus	Caused during	Discovered during	Act total eff	Create time	Submitted time	In analysis time	Analyzed time	In resolution time	Resolved time	In evaluation time	Evaluated time	Modify time	Modifiable in	Discovered on	Project	Team
22-Nov-06	XXX	2714	Medium	B	PR	submitted		Design		04-Sep-06	04-Sep-06							06-Sep-06	XXX	DEV		PSV
20-Dec-06	XXX	2714	Medium	B	PR	submitted		Design		04-Sep-06	04-Sep-06							06-Sep-06	XXX	DEV		PSV
24-Jan-07	XXX	2714	Medium	B	PR	in analysis		Design		04-Sep-06	04-Sep-06	18-Jan-07						18-Jan-07	XXX	DEV		PSV
21-Feb-07	XXX	2714	Medium	B	PR	in analysis		Design		04-Sep-06	04-Sep-06	18-Jan-07						18-Jan-07	XXX	DEV		PSV
21-Mar-07	XXX	2714	Medium	B	PR	in analysis		Design		04-Sep-06	04-Sep-06	18-Jan-07						01-Mar-07	XXX	DEV		PSV
18-Apr-07	XXX	2714	Medium	B	PR	in resolution	Requirements	Design		04-Sep-06	04-Sep-06	18-Jan-07	27-Mar-07	02-Apr-07				02-Apr-07	XXX	DEV		PSV
30-May-07	XXX	2714	Medium	B	PR	in resolution	Requirements	Design		04-Sep-06	04-Sep-06	18-Jan-07	27-Mar-07	02-Apr-07				02-May-07	XXX	DEV		PSV
25-Jun-07	XXX	2714	Medium	B	PR	in resolution	Requirements	Design		04-Sep-06	04-Sep-06	18-Jan-07	27-Mar-07	02-Apr-07				02-May-07	XXX	DEV		PSV
25-Jul-07	XXX	2714	Medium	B	PR	in resolution	Requirements	Design		04-Sep-06	04-Sep-06	18-Jan-07	27-Mar-07	02-Apr-07				02-May-07	XXX	DEV-FP		PSV
22-Aug-07	XXX	2714	Medium	B	PR	in resolution	Requirements	Design		04-Sep-06	04-Sep-06	18-Jan-07	27-Mar-07	02-Apr-07				02-May-07	XXX	DEV-FP		PSV
26-Sep-07	XXX	2714	Medium	B	PR	in resolution	Requirements	Design		04-Sep-06	04-Sep-06	18-Jan-07	27-Mar-07	02-Apr-07				05-Sep-07	XXX	DEV-FP	DEV	PSV
03-Oct-07	XXX	2714	Medium	B	PR	in resolution	Requirements	Design		04-Sep-06	04-Sep-06	18-Jan-07	27-Mar-07	02-Apr-07				05-Sep-07	XXX	DEV-FP	DEV	PSV
22-Oct-07	XXX	2714	Medium	B	PR	concluded	Requirements	Design	1	04-Sep-06	04-Sep-06	18-Jan-07	27-Mar-07	02-Apr-07	17-Oct-07	19-Oct-07	19-Oct-07	19-Oct-07	XXX	DEV-FP	DEV	PSV

C. Second Attempt program

First file

```
<?PHP
include "bdd.inc";
include "var.php";

function name2time($name) {
        if ($name<0 or $name>=16*60) {print "Time value '$name' out of range!<br>\n";}
        return sprintf("%02d:%02d:%02d", 8+floor($name/60), $name % 60, 0);
}

$tableName  = 'data';
$batch_size = 1000;
$reload_data = TRUE;

if($bsend=="Execute"){

        $lasttime = microtime(TRUE);

        mysql_query('USE `edy`');

        mysql_query('DROP TABLE IF EXISTS `states`');
        mysql_query('CREATE TABLE `states` (
                        `id_state` int(10) NOT NULL auto_increment,
                        `id` char(255) NOT NULL,
                        `team` char(255) default NULL,
                        `name` char(200) NOT NULL,
                        `date` date NOT NULL,
                        `time` time NOT NULL,
                        `state` int(11) default NULL,
                        `file` int(10) default NULL,
                        `id_data` int(11) NOT NULL,
                        PRIMARY KEY  (`id_state`)
                ) ENGINE=InnoDB DEFAULT CHARSET=latin1');
        print mysql_error();

        mysql_query('DROP TABLE IF EXISTS `tasks`');
        mysql_query('CREATE TABLE `tasks` (
                        `id` int(10) NOT NULL,
                        `id_task` int(10) NOT NULL auto_increment,
                        `name_task` char(100) NOT NULL,
                        `id_state1` int(10) NOT NULL,
                        `name_state1` char(100) NOT NULL,
                        `date_state1` date NOT NULL,
                        `time_state1` time NOT NULL,
                        `id_state2` int(10) NOT NULL,
                        `name_state2` char(100) NOT NULL,
                        `date_state2` date NOT NULL,
                        `time_state2` time NOT NULL,
                        `state_task` int(11) default NULL,
                        PRIMARY KEY  (`id_task`)
                ) ENGINE=InnoDB DEFAULT CHARSET=latin1');
        print mysql_error();

        if ($reload_data) {
                mysql_query('DROP TABLE IF EXISTS `data`');
                mysql_query('CREATE TABLE `data` (
                                `DataSet` char(255) default NULL,
                                `HistoryDate` date default NULL,
                                `System` char(255) default NULL,
                                `Subsystem` char(255) default NULL,
                                `Problem_number` int(11) NOT NULL,
                                `Product_name` char(255) default NULL,
                                `Product_subsys` char(255) default NULL,
                                `Version` double(10,5) default NULL,
                                `Release` char(255) default NULL,
                                `Priority` char(255) default NULL,
                                `Severity` char(1) default NULL,
```

```
                        `Defect_type` char(255) default NULL,
                        `Problem_type` char(255) default NULL,
                        `Request_type` char(255) default NULL,
                        `Crstatus` char(255) default NULL,
                        `Caused_during` char(255) default NULL,
                        `Discovered_during` char(255) default NULL,
                        `Act_total_eff` double(10,5) default NULL,
                        `Create_time` date default NULL,
                        `Submitted_time` date default NULL,
                        `In_analysis_time` date default NULL,
                        `Analysed_time` date default NULL,
                        `In_resolution_time` date default NULL,
                        `Resolved_time` date default NULL,
                        `In_evaluation_time` date default NULL,
                        `Evaluated_time` date default NULL,
                        `Modify_time` date default NULL,
                        `Modifiable_in` char(255) default NULL,
                        `Discovered_on` char(255) default NULL,
                        `Team` char(255) default NULL,
                        `Scope` char(255) default NULL,
                        `file` int(10) default NULL,
                        `id` float NOT NULL auto_increment,
                        PRIMARY KEY  (`id`)
                      ) ENGINE=InnoDB DEFAULT CHARSET=latin1');
            print mysql_error();

            $csvfile = '../../www/edy/files/';

//READ FILES'S DIRECTORY-----------------------------------

            $dir = "c:/wamp/www/edy/files/";
            $filen=1;
            // Open a known directory, and proceed to read its contents
            if (is_dir($dir)) {
                    if ($dh = opendir($dir)) {
                            while (($file = readdir($dh)) !== false) {
                                    //echo "filename: $file : filetype: " . filetype($dir .
                            $file) . "\n";
                                    if($file!="." and $file!=".."){
                                            $csvfile=$dir.$file;
                                            echo "<br>".$file;
                            $loadsql = 'LOAD DATA INFILE "'.$csvfile.'" INTO TABLE
                            '.$tableName.' FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY """"
                            LINES TERMINATED BY "\r" IGNORE 1 LINES';
                                            mysql_query($loadsql) or die(mysql_error());
                                            mysql_query("update data set file=$filen where file
IS NULL");
                                            print mysql_error();
                                            $filen++;
                                    }

                            }
                            //mysql_query("delete from data where CONVERT('DataSet' USING
utf8)='\n'");
                            //print mysql_error();

                            closedir($dh);
                    }
            }
            else{echo "The directory does not exist";}

            print "<br>\nmake data took: ".(microtime(TRUE) - $lasttime)."<br>\n";
            $lasttime = microtime(TRUE);
    }

    //CREATE TABLES
    $raf=0;
    $rrf=0;
    $ref=0;
    $sql=mysql_query("select * from data where Problem_number>0");
    print mysql_error();
```

```
print "<br>\nquery on data took: ".(microtime(TRUE) - $lasttime)."<br>\n";
$lasttime = microtime(TRUE);

if(mysql_num_rows($sql)>0){
        $statePr="0000-00-00";
        $lastid = -1;
        $query = array();
        while ($row = mysql_fetch_assoc($sql)) {
                $rid=$row["id"];
                $rProblem_number=$row["Problem_number"];
                $rSubsystem=$row["Subsystem"];
                $rTeam=$row["Team"];
                $rHistoryDate=$row["HistoryDate"];
                $rCrstatus=$row["Crstatus"];
                $rCreate_time=$row["Create_time"];
                $rSubmitted_time=$row["Submitted_time"];
                $rIn_analysis_time=$row["In_analysis_time"];
                $rAnalysed_time=$row["Analysed_time"];
                $rIn_resolution_time=$row["In_resolution_time"];
                $rResolved_time=$row["Resolved_time"];
                $rIn_evaluation_time=$row["In_evaluation_time"];
                $rEvaluated_time=$row["Evaluated_time"];
                $rfile=$row["file"];

                $print3 = TRUE;

                if ($lastid != $rid) {
                        $SeqCounter = 0;
                        $lastid = $rid;
                }
                $mixid = '"'.$rSubsystem."/".$rProblem_number.'"';

                if (substr($rCrstatus, 0, 9) == "duplicate") {$rCrstatus = "duplicate";}

                if ($rCreate_time!="0000-00-00"){
                        array_push($query,
                        "($rid,$mixid,\"$rTeam\",\"create\",\"$rCreate_time\",\"".name2time
                        ($SeqCounter++)."\",1,$rfile)");
                        $statePr=$rCreate_time;
                        $print3 &= $rCrstatus != "create";
                }
                if ($rSubmitted_time!="0000-00-00" and $rSubmitted_time>=$statePr){
                        array_push($query,
                        "($rid,$mixid,\"$rTeam\",\"submitted\",\"$rSubmitted_time\",\"".nam
                        e2time($SeqCounter++)."\",1,$rfile)");
                        $statePr=$rSubmitted_time;
                        $print3 &= $rCrstatus != "submitted";
                }
                //-----------
                if ($rIn_analysis_time!="0000-00-00" and $rIn_analysis_time>=$statePr){
                        array_push($query,
                        "($rid,$mixid,\"$rTeam\",\"in_analysis\",\"$rIn_analysis_time\",\""
                        .name2time($SeqCounter++)."\",1,$rfile)");
                        $statePr=$rIn_analysis_time;
                        $print3 &= $rCrstatus != "in_analysis";
                }
                if ($rAnalysed_time!="0000-00-00" and $rAnalysed_time>=$statePr){
                        array_push($query,
                        "($rid,$mixid,\"$rTeam\",\"analysed\",\"$rAnalysed_time\",\"".name2
                        time($SeqCounter++)."\",1,$rfile)");
                        $statePr=$rAnalysed_time;
                        $print3 &= $rCrstatus != "analysed";
                }
                //-----------
                if ($rIn_resolution_time!="0000-00-00" and $rIn_resolution_time>=$statePr){
                        array_push($query,
                "($rid,$mixid,\"$rTeam\",\"in_resolution\",\"$rIn_resolution_time\",\"".nam
                e2time($SeqCounter++)."\",1,$rfile)");
                        $statePr=$rIn_resolution_time;
                        $print3 &= $rCrstatus != "in_resolution";
```

```
                }
                if ($rResolved_time!="0000-00-00" and $rResolved_time>=$statePr){
                        array_push($query,
                "($rid,$mixid,\"$rTeam\",\"resolved\",\"$rResolved_time\",\"".name2time($S
                eqCounter++)."\",1,$rfile)");
                        $statePr=$rResolved_time;
                        $print3 &= $rCrstatus != "resolved";
                }
                //----------
                                if ($rIn_evaluation_time!="0000-00-00" and
                $rIn_evaluation_time>=$statePr){
                        array_push($query,
        "($rid,$mixid,\"$rTeam\",\"in_evaluation\",\"$rIn_evaluation_time\",\"".name2time(
        $SeqCounter++)."\",1,$rfile)");
                        $statePr=$rIn_evaluation_time;
                        $print3 &= $rCrstatus != "in_evaluation";
                }
                if ($rEvaluated_time!="0000-00-00" and $rEvaluated_time>=$statePr){
                        array_push($query,
        "($rid,$mixid,\"$rTeam\",\"evaluated\",\"$rEvaluated_time\",\"".name2time($SeqCoun
        ter++)."\",1,$rfile)");
                        $statePr=$rEvaluated_time;
                        $print3 &= $rCrstatus != "evaluated";
                }
                if ($print3) {
                        array_push($query,
        "($rid,$mixid,\"$rTeam\",\"$rCrstatus\",\"$rHistoryDate\",\"".name2time($SeqCounte
        r++)."\",3,$rfile)");
                }
                if (count($query) >= $batch_size) {
                        mysql_query("insert into states
                (id_data,id,team,name,date,time,state,file) values".join(",", $query));
                        print mysql_error();
                        $query = array();
                }
        }
        if (count($query) > 0) {
                mysql_query("insert into states
        (id_data,id,team,name,date,time,state,file) values".join(",", $query));
                print mysql_error();
        }
}

print "<br>\nmake states took: ".(microtime(TRUE) - $lasttime)."<br>\n";
$lasttime = microtime(TRUE);

//CREATE ARTIFICAL EVENTS-------------------------------------------------

$sql=mysql_query("SELECT id_state, id_data, id, team, name, MIN(date) as date, time,
state
                FROM `states`
                GROUP BY id, team, name, state
                ORDER BY id, team, date, time, id_state");
print mysql_error();

print "<br>\nquery on states took: ".(microtime(TRUE) - $lasttime)."<br>\n";
$lasttime = microtime(TRUE);

if(mysql_num_rows($sql)>0){
        $start2complete = array('concluded'          => 'concluded',
                                'create'             => 'submitted',
                                'duplicate'          => 'duplicate',
                                'in_analysis'        => 'analysed',//, analysis_failed
                                'in_evaluation'              => 'evaluated',//,
                                evaluation_failed
                                'in_resolution'              => 'resolved',//,
                                resolution_failed
                                'later_release'              => 'later_release',
                                'not_reproducible'   => 'not_reproducible',
                                'on_hold'            => 'on_hold',
                                'rejected'           => 'rejected'
```

```
                        );

$complete2start = array('analysis_failed'    => 'in_analysis',
                        'evaluation_failed'   => 'in_evaluation',
                        'resolution_failed'   => 'in_resolution'
                        );
foreach ($start2complete as $key => $value) {
        if ($key != $value) {$complete2start[$value] = $key;}
}

$lastrow = array();
$query = array();
while ($row = mysql_fetch_assoc($sql)) {
        $rid_state=$lastrow["id_state"];
        $rid_data=$lastrow["id_data"];
        $rid=$lastrow["id"];
        $rteam=$lastrow["team"];
        $rname=$lastrow["name"];
        $rdate=$lastrow["date"];
        $rtime=$lastrow["time"];
        $rstate=$lastrow["state"];

        $r1id_state=$row["id_state"];
        $r1id_data=$row["id_data"];
        $r1id=$row["id"];
        $r1team=$row["team"];
        $r1name=$row["name"];
        $r1date=$row["date"];
        $r1time=$row["time"];
        $r1state=$row["state"];

        //i. Two consecutive "Start" events coming from different tasks
        //ii. Two consecutive "Start" events coming from equal tasks but different
dates
        if (isset($start2complete[$rname]) and isset($start2complete[$r1name]) and
        (($name!=$r1name) or (($rname==$r1name) and ($rdate!=$r1date)))) {
                //insert complete; name/id/team = 1st's, d/t = 1st's + 20 sec
                $time = strtotime("$rdate $rtime") + 20;
        if (($rid==$r1id) and ($rteam==$r1team) and ($time >= strtotime("$r1date
        $r1time"))) {
                        print "i/ii: Time crash! $rdate $rtime + 20 >= $r1date
                        $r1time<br>\n";
                        print "$rid_state, $rid, $rteam, $rname, $rdate,
                        $rtime<br>\n";
                        print "$r1id_state, $r1id, $r1team, $r1name, $r1date,
                        $r1time<br>\n";
                }
                array_push($query,     "($rid_data,
                                       \"$rid\",
                                       \"$rteam\",
                                       \"$start2complete[$rname]\",
                                       \"".date("Y-m-d", $time)."\",
                                       \"".date("H:i:s", $time)."\",
                                       ".(5+$rstate).")");
                //print "$rid_state/$r1id_state: inserted i/ii<br>\n";
        }

        //iii. Two consecutive "Complete" events coming from different tasks
        //iv. Two consecutive "Complete" events coming from equal tasks but
        different dates
        if (isset($complete2start[$rname]) and isset($complete2start[$r1name]) and
        (($complete2start[$rname]!=$complete2start[$r1name]) or
        (($complete2start[$rname]==$complete2start[$r1name]) and
        ($rdate!=$r1date)))) {
                //insert start; name/id/team = 2nd's, d/t = 2nd's - 20 sec
                $time = strtotime("$r1date $r1time") - 20;
                if (($rid == $r1id) and ($rteam==$r1team) and ($time <=
                strtotime("$rdate $rtime"))) {
                        print "iii/iv: Time crash! $r1date $r1time - 20 <= $rdate
                        $rtime<br>\n";
```

```
                        print "$rid_state, $rid, $rteam, $rname, $rdate,
                        $rtime<br>\n";
                        print "$r1id_state, $r1id, $r1team, $r1name, $r1date,
                        $r1time<br>\n";
                }
                array_push($query,      "($r1id_data,
                                        \"$r1id\",
                                        \"$r1team\",
                                        \"$complete2start[$r1name]\",
                                        \"".date("Y-m-d", $time)."\",
                                        \"".date("H:i:s", $time)."\",
                                        ".(5+$r1state).")");
                //print "$rid_state/$r1id_state: inserted iii/iv<br>\n";
        }

        //v. One "Start" event followed by a "Complete" event but from a different
        task
        if (isset($start2complete[$rname]) and isset($complete2start[$r1name]) and
        ($rname!=$complete2start[$r1name])) {
                //insert complete; name/id/team = 1st's, d/t = 1st's + 20 sec
                $time1 = strtotime("$rdate $rtime") + 20;
                array_push($query,      "($rid_data,
                                        \"$rid\",
                                        \"$rteam\",
                                        \"$start2complete[$rname]\",
                                        \"".date("Y-m-d", $time1)."\",
                                        \"".date("H:i:s", $time1)."\",
                                        ".(5+$rstate).")");
                //insert start; name/id/team = 2nd's, d/t = 2nd's - 20 sec
                $time2 = strtotime("$r1date $r1time") - 20;
                if (($rid == $r1id) and ($rteam==$r1team) and ($time1 >= $time2)) {
                        print "v: Time crash! $rdate $rtime + 20 >= $r1date $r1time
                        - 20<br>\n";
                        print "$rid_state, $rid, $rteam, $rname, $rdate,
                        $rtime<br>\n";
                        print "$r1id_state, $r1id, $r1team, $r1name, $r1date,
                        $r1time<br>\n";
                }
                array_push($query,      "($r1id_data,
                                        \"$r1id\",
                                        \"$r1team\",
                                        \"$complete2start[$r1name]\",
                                        \"".date("Y-m-d", $time2)."\",
                                        \"".date("H:i:s", $time2)."\",
                                        ".(5+$r1state).")");
                //print "$rid_state/$r1id_state: inserted V<br>\n";
        }

        if (count($query) >= $batch_size) {
                mysql_query("insert into states
                (id_data,id,team,name,date,time,state) values".join(",", $query));
                print mysql_error();
                $query = array();
        }

        $lastrow = $row;
    }
    if (count($query) > 0) {
            mysql_query("insert into states (id_data,id,team,name,date,time,state)
            values".join(",", $query));
            print mysql_error();
    }
}

print "<br>\nadd artificial events took: ".(microtime(TRUE) - $lasttime)."<br>\n";
$lasttime = microtime(TRUE);
```

```
        $sql=mysql_query("DROP TABLE IF EXISTS `temp2`");
        print mysql_error();

        $sql=mysql_query("CREATE TABLE `temp2` (
                                `id` char(255) NOT NULL,
                                `name` char(200) NOT NULL,
                                `state` int(10) default NULL,
                                `date` date NOT NULL,
                                `id_data` int(10) NOT NULL,
                                `team` char(255) default NULL,
                                `id_state` int(10) NOT NULL auto_increment,
                                `time` time NOT NULL,
                                PRIMARY KEY  (`id_state`)
                        ) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1");
                        print mysql_error();

        $sql=mysql_query("INSERT INTO temp2 SELECT id, name, state, `date`, MIN(id_data),
MIN(team), min(id_state), min(`time`)
                                FROM `states`
                                GROUP BY id, name, state, `date`
                                HAVING state!=3 AND state!=8
                                ORDER BY id, name, state, `date`");
        print mysql_error();

        print "<br>\ninsert into temp2 A took: ".(microtime(TRUE) - $lasttime)."<br>\n";
        $lasttime = microtime(TRUE);

        $sql=mysql_query("INSERT INTO temp2 SELECT id, name, state, MIN(`date`), MIN(id_data),
MIN(team), min(id_state), min(`time`)
                                FROM `states`
                                GROUP BY id, name, state
                                HAVING state=3 OR state=8
                                ORDER BY id, name, state");
        print mysql_error();

        print "<br>\ninsert into temp2 B took: ".(microtime(TRUE) - $lasttime)."<br>\n";
        $lasttime = microtime(TRUE);



//MATCH PAIRS-------------------------------------------------------------
        $sql=mysql_query("SELECT id_state, id_data, id, team, name, MIN(date) as date, time,
state
                                FROM `temp2`
                                GROUP BY id, team, name, state
                                ORDER BY id, team, date, time, id_state
                               ");
        print mysql_error();

        print "<br>\nquery on states took: ".(microtime(TRUE) - $lasttime)."<br>\n";
        $lasttime = microtime(TRUE);

        if(mysql_num_rows($sql)>0){
                $lastrow = array();
                $query = array();
                while ($row = mysql_fetch_assoc($sql)) {
                        $rid_state=$lastrow["id_state"];
                        $rid_data=$lastrow["id_data"];
                        $rid=$lastrow["id"];
                        $rteam=$lastrow["team"];
                        $rname=$lastrow["name"];
                        $rdate=$lastrow["date"];
                        $rtime=$lastrow["time"];
                        $rstate=$lastrow["state"];

                        $r1id_state=$row["id_state"];
                        $r1id_data=$row["id_data"];
                        $r1id=$row["id"];
                        $r1team=$row["team"];
                        $r1name=$row["name"];
```

```
                        $r1date=$row["date"];
                        $r1time=$row["time"];
                        $r1state=$row["state"];

                        $rname_task="undef";

                        if($rname=="in_analysis" and $r1name=="analysed")
        {$rname_task="analysis";}
                        if($rname=="in_analysis" and $r1name=="analysis_failed")
        {$rname_task="analysis";}
                        if($rname=="in_evaluation" and $r1name=="evaluated")
        {$rname_task="evaluation";}
                        if($rname=="in_evaluation" and $r1name=="evaluation_failed")
        {$rname_task="evaluation";}
                        if($rname=="in_resolution" and $r1name=="resolved")
        {$rname_task="resolution";}
                        if($rname=="in_resolution" and $r1name=="resolution_failed")
        {$rname_task="resolution";}
                        if($rname=="create" and $r1name=="submitted")
        {$rname_task="submit";}

                        if($rname=="concluded" and $r1name==$rname)
        {$rname_task=$rname;}
                        if($rname=="duplicate" and $r1name==$rname)
        {$rname_task="CCBevaluation";}
                        if($rname=="later_release" and $r1name==$rname)
        {$rname_task="CCBevaluation";}
                        if($rname=="not_reproducible" and $r1name==$rname)
        {$rname_task="CCBevaluation";}
                        if($rname=="on_hold" and $r1name==$rname)
        {$rname_task="CCBevaluation";}
                        if($rname=="rejected" and $r1name==$rname)
        {$rname_task="CCBevaluation";}

//INSERT-----------------------------------------------

                        if($rid==$r1id and $rname_task!="undef"){
                                if ($rname_task == "undef") {
                                        print "Warning: task name undefined on:<br>\n";
                                        print "$r1id_state, $r1id, $r1name, $r1date,
$r1time<br>\n";
                                        print "$rid_state, $rid, $rname, $rdate, $rtime<br>\n";
                                        print "<br>\n";
                                }
                                array_push($query,      "(\"$rid_data\",\"$rname_task\",
                                                        \"$rid_state\" ,\"$rname\" ,\"$rdate\"
                                                        ,\"$rtime\" ,

        \"$r1id_state\",\"$r1name\",\"$r1date\",\"$r1time\",
                                                        ".($rstate*100 + $r1state).")");
                        }

                        if (count($query) >= $batch_size) {
                                mysql_query("insert into tasks (id,name_task,
id_state1,name_state1,date_state1,time_state1, id_state2,name_state2,date_state2,time_state2,
state_task) values".join(",", $query));
                                print mysql_error();
                                $query = array();
                        }

                        $lastrow = $row;
                }
                if (count($query) > 0) {
                        mysql_query("insert into tasks (id,name_task,
id_state1,name_state1,date_state1,time_state1, id_state2,name_state2,date_state2,time_state2,
state_task) values".join(",", $query));
                        print mysql_error();
                }
        }

        print "<br>\nmake tasks took: ".(microtime(TRUE) - $lasttime)."<br>\n";
```

```
        $lasttime = microtime(TRUE);

}
?>

<form name="form1" method="post" action="loadA.php">
<table width="90%"  border="1">
<tr>
<td >
<input type="submit" name="bsend" value="Execute">
</td>
</tr>
</table>
</form>
```

Second file

```
<?PHP
include "bdd.inc";
include "var.php";

if($bsend=="Execute"){


//CREATE XML FILE---------------------------------------------
//-----------------------------------------------------------

        $fp = fopen("ccb_".date("Y-m-d_H.i.s", time()).".xml", "w+");

        fputs($fp,"<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>
        <WorkflowLog xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"http://is.tm.tue.nl/research/processmining/WorkflowLog.xsd\">
        <Data>
        <Attribute name=\"app.name\">Name of the Program</Attribute>
        </Data>
        <Source program=\"Thesis\"/>
        <Process id=\"MTRA\" >");

        $ridBefore='';

        $lasttime = microtime(TRUE);

//BODY---------------

        $sql=mysql_query("SELECT Problem_number, Subsystem, name_task, date_state2, date_state1,
min(id_task) AS id_task, `Release`, Priority, Severity, Request_type, Caused_during,
Discovered_on, Team, name_state1, name_state2, time_state1, time_state2
                        FROM `data`, tasks
                        WHERE data.id = tasks.id
                        GROUP BY Problem_number, Subsystem, name_task, date_state2
                        ORDER BY Problem_number, Subsystem, date_state1");
        print mysql_error();

        print "<br>\n<br>\nquery took: ".(microtime(TRUE) - $lasttime)."<br>\n";
        $lasttime = microtime(TRUE);

        if(mysql_num_rows($sql)>0){
                $query = array();
                while ($row = mysql_fetch_assoc($sql)) {
                        $rid                    = $row["Problem_number"]."-".$row["Subsystem"];
                        $rRelease               = $row["Release"];
                        $rPriority              = $row["Priority"];
                        $rSeverity              = $row["Severity"];
                        $rRequest_type = $row["Request_type"];
                        $rDiscovered_on         = $row["Discovered_on"];
                        $rCaused_during         = $row["Caused_during"];
                        $rTeam                  = $row["Team"];

                        /*
                        $rDataSet               = $row["DataSet"];
                        $rHistoryDate           = $row["HistoryDate"];
                        $rSystem                = $row["System"];
                        $rProduct_name          = $row["Product_name"];
                        $rProduct_subsys        = $row["Product_subsys"];
                        $rVersion               = $row["Version"];
                        $rDefect_type           = $row["Defect_type"];
                        $rProblem_type          = $row["Problem_type"];
                        $rCrstatus              = $row["Crstatus"];
                        $rCaused_during                 = $row["Caused_during"];
                        $rAct_total_eff                 = $row["Act_total_eff"];
                        $rCreate_time           = $row["Create_time"];
                        $rSubmitted_time        = $row["Submitted_time"];
                        $rIn_analysis_time      = $row["In_analysis_time"];
                        $rAnalysed_time                 = $row["Analysed_time"];
                        $rIn_resolution_time    = $row["In_resolution_time"];
                        $rResolved_time                 = $row["Resolved_time"];
```

```
                        $rIn_evaluation_time   = $row["In_evaluation_time"];
                        $rEvaluated_time       = $row["Evaluated_time"];
                        $rModify_time          = $row["Modify_time"];
                        $rModifiable_in               = $row["Modifiable_in"];
                        $rDiscovered_on               = $row["Discovered_on"];
                        $rScope                = $row["Scope"];
                        */
// TASK TABLE-----------------------------------------------

                        $rname_task            = $row["name_task"];
                        $rdate_state1          =
                        $row["date_state1"]."T".$row["time_state1"].".000+00:00";
                        $rdate_state2          =
                        $row["date_state2"]."T".$row["time_state2"].".000+00:00";
                        $rname_state1          = $row["name_state1"];
                        $rname_state2          = $row["name_state2"];

                        /*
                        $rid_task              = $row["id_task"];
                        */

                        if($rid != $ridBefore){
                                if($ridBefore != ''){
                                        fputs($fp,"\n</ProcessInstance>");
                                }
                                fputs($fp,"\n<ProcessInstance id=\"$rid\" description=\"\">");
                        }

                                        if ($rname_state2=="analysis_failed" or
                $rname_state2=="evaluation_failed" or $rname_state2=="resolution_failed") {
                                $Execution = "unsuccessful";
                        } else if ($rname_task=="concluded" or $rname_task=="CCBevaluation") {
                                $Execution = $rname_state1;
                        } else {
                                $Execution = "successful";
                        }

                        foreach(array("start", "complete") as $rEventType) {
                                $rdate_state = $rEventType == "start" ? $rdate_state1 :
$rdate_state2;
                                fputs($fp,"
                                <AuditTrailEntry>
                                        <Data>
                                                <Attribute name=\"Product_name\">$rTeam</Attribute>
                                                <Attribute name=\"Release\">$rRelease</Attribute>
                                                <Attribute name=\"Priority\">$rPriority</Attribute>
                                                <Attribute name=\"Severity\">$rSeverity</Attribute>
                                                <Attribute
                                                Name=\"Request_type\">$rRequest_type</Attribute>
                                                <Attribute
                                                name=\"Discovered_on\">$rDiscovered_on</Attribute>
                                                <Attribute name=\"Execution\">$Execution</Attribute>
                                                <Attribute
                                                name=\"Caused_during\">$rCaused_during</Attribute>
                                        </Data>
                                        <WorkflowModelElement>$rname_task</WorkflowModelElement>
                                        <EventType >$rEventType</EventType>
                                        <Timestamp>$rdate_state</Timestamp>
                                        <Originator>$rTeam</Originator>
                                </AuditTrailEntry>");
                        }

                        $ridBefore=$rid;
                }
                fputs($fp,"</ProcessInstance>\n");
        }

        fputs($fp,"</Process>
        </WorkflowLog>");

        print "<br>\n<br>\ngenerating xml took: ".(microtime(TRUE) - $lasttime)."<br>\n";
```

```
}
?>

<form name="form1" method="post" action=<?PHP print '"'.basename(__FILE__).'"' ?>>
<table width="90%"  border="1">
<tr>
<td >
<input type="submit" name="bsend" value="Execute">
</td>
</tr>
</table>
</form>
```

D. Event-log of 2714 CI's defect in XML format

```
<ProcessInstance id="2714-XXX" description="">
                    <AuditTrailEntry>
                            <Data>
                                    <Attribute name="Product_name">PSV</Attribute>
                                    <Attribute name="Release"></Attribute>
                                    <Attribute name="Priority">Medium</Attribute>
                                    <Attribute name="Severity">B</Attribute>
                                    <Attribute name="Request_type">PR</Attribute>
                                    <Attribute name="Discovered_on">DEV-FP</Attribute>
                                    <Attribute name="Execution">successful</Attribute>
                                    <Attribute name="Caused_during"></Attribute>
                            </Data>
                            <WorkflowModelElement>submit</WorkflowModelElement>
                            <EventType >start</EventType>
                            <Timestamp>2006-09-04T08:00:00.000+00:00</Timestamp>
                            <Originator>PSV</Originator>
                    </AuditTrailEntry>
                    <AuditTrailEntry>
                            <Data>
                                    <Attribute name="Product_name">PSV</Attribute>
                                    <Attribute name="Release"></Attribute>
                                    <Attribute name="Priority">Medium</Attribute>
                                    <Attribute name="Severity">B</Attribute>
                                    <Attribute name="Request_type">PR</Attribute>
                                    <Attribute name="Discovered_on">DEV-FP</Attribute>
                                    <Attribute name="Execution">successful</Attribute>
                                    <Attribute name="Caused_during"></Attribute>
                            </Data>
                            <WorkflowModelElement>submit</WorkflowModelElement>
                            <EventType >complete</EventType>
                            <Timestamp>2006-09-04T08:01:00.000+00:00</Timestamp>
                            <Originator>PSV</Originator>
                    </AuditTrailEntry>
                    <AuditTrailEntry>
                            <Data>
                                    <Attribute name="Product_name">PSV</Attribute>
                                    <Attribute name="Release">REL-6</Attribute>
                                    <Attribute name="Priority">Medium</Attribute>
                                    <Attribute name="Severity">B</Attribute>
                                    <Attribute name="Request_type">PR</Attribute>
                                    <Attribute name="Discovered_on">DEV-FP</Attribute>
                                    <Attribute name="Execution">successful</Attribute>
                                    <Attribute name="Caused_during"></Attribute>
                            </Data>
                            <WorkflowModelElement>analysis</WorkflowModelElement>
                            <EventType >start</EventType>
                            <Timestamp>2007-01-18T08:02:00.000+00:00</Timestamp>
                            <Originator>PSV</Originator>
                    </AuditTrailEntry>
                    <AuditTrailEntry>
                            <Data>
                                    <Attribute name="Product_name">PSV</Attribute>
                                    <Attribute name="Release">REL-6</Attribute>
                                    <Attribute name="Priority">Medium</Attribute>
                                    <Attribute name="Severity">B</Attribute>
                                    <Attribute name="Request_type">PR</Attribute>
                                    <Attribute name="Discovered_on">DEV-FP</Attribute>
                                    <Attribute name="Execution">successful</Attribute>
                                    <Attribute name="Caused_during"></Attribute>
                            </Data>
                            <WorkflowModelElement>analysis</WorkflowModelElement>
                            <EventType >complete</EventType>
                            <Timestamp>2007-03-27T08:03:00.000+00:00</Timestamp>
                            <Originator>PSV</Originator>
                    </AuditTrailEntry>
                    <AuditTrailEntry>
                            <Data>
```

```
                <Attribute name="Product_name">PSV</Attribute>
                <Attribute name="Release">REL-7</Attribute>
                <Attribute name="Priority">Medium</Attribute>
                <Attribute name="Severity">B</Attribute>
                <Attribute name="Request_type">PR</Attribute>
                <Attribute name="Discovered_on">DEV-FP</Attribute>
                <Attribute name="Execution">successful</Attribute>
                <Attribute name="Caused_during">Requirements</Attribute>
        </Data>
        <WorkflowModelElement>resolution</WorkflowModelElement>
        <EventType >start</EventType>
        <Timestamp>2007-04-02T08:04:00.000+00:00</Timestamp>
        <Originator>PSV</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
        <Data>
                <Attribute name="Product_name">PSV</Attribute>
                <Attribute name="Release">REL-7</Attribute>
                <Attribute name="Priority">Medium</Attribute>
                <Attribute name="Severity">B</Attribute>
                <Attribute name="Request_type">PR</Attribute>
                <Attribute name="Discovered_on">DEV-FP</Attribute>
                <Attribute name="Execution">successful</Attribute>
                <Attribute name="Caused_during">Requirements</Attribute>
        </Data>
        <WorkflowModelElement>resolution</WorkflowModelElement>
        <EventType >complete</EventType>
        <Timestamp>2007-10-17T08:05:00.000+00:00</Timestamp>
        <Originator>PSV</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
        <Data>
                <Attribute name="Product_name">BLURAY</Attribute>
                <Attribute name="Release">PBP_SUN_SIGMA-1</Attribute>
                <Attribute name="Priority">High</Attribute>
                <Attribute name="Severity">B</Attribute>
                <Attribute name="Request_type">PR</Attribute>
                <Attribute name="Discovered_on">DEV-FP</Attribute>
                <Attribute name="Execution">successful</Attribute>
                <Attribute name="Caused_during">Requirements</Attribute>
        </Data>
        <WorkflowModelElement>evaluation</WorkflowModelElement>
        <EventType >start</EventType>
        <Timestamp>2007-10-19T08:06:00.000+00:00</Timestamp>
        <Originator>BLURAY</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
        <Data>
                <Attribute name="Product_name">BLURAY</Attribute>
                <Attribute name="Release">PBP_SUN_SIGMA-1</Attribute>
                <Attribute name="Priority">High</Attribute>
                <Attribute name="Severity">B</Attribute>
                <Attribute name="Request_type">PR</Attribute>
                <Attribute name="Discovered_on">DEV-FP</Attribute>
                <Attribute name="Execution">successful</Attribute>
                <Attribute name="Caused_during">Requirements</Attribute>
        </Data>
        <WorkflowModelElement>evaluation</WorkflowModelElement>
        <EventType >complete</EventType>
        <Timestamp>2007-10-19T08:07:00.000+00:00</Timestamp>
        <Originator>BLURAY</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
        <Data>
                <Attribute name="Product_name">BLURAY</Attribute>
                <Attribute name="Release">PBP_SUN_SIGMA1</Attribute>
                <Attribute name="Priority">High</Attribute>
                <Attribute name="Severity">B</Attribute>
                <Attribute name="Request_type">PR</Attribute>
                <Attribute name="Discovered_on">DEV-FP</Attribute>
                <Attribute name="Execution">concluded</Attribute>
```

```
                <Attribute name="Caused_during">Requirements</Attribute>
        </Data>
        <WorkflowModelElement>concluded</WorkflowModelElement>
        <EventType >start</EventType>
        <Timestamp>2007-10-22T08:08:00.000+00:00</Timestamp>
        <Originator>BLURAY</Originator>
</AuditTrailEntry>
<AuditTrailEntry>
        <Data>
                <Attribute name="Product_name">BLURAY</Attribute>
                <Attribute name="Release">PBP_SUN_SIGMA-1</Attribute>
                <Attribute name="Priority">High</Attribute>
                <Attribute name="Severity">B</Attribute>
                <Attribute name="Request_type">PR</Attribute>
                <Attribute name="Discovered_on">DEV-FP</Attribute>
                <Attribute name="Execution">concluded</Attribute>
                <Attribute name="Caused_during">Requirements</Attribute>
        </Data>
        <WorkflowModelElement>concluded</WorkflowModelElement>
        <EventType >complete</EventType>
        <Timestamp>2007-10-22T08:08:20.000+00:00</Timestamp>
        <Originator>BLURAY</Originator>
</AuditTrailEntry>
```