# Customer profiles

3TU. School for Technological Design

STAN ACKERMANS INSTITUTE

# Customer Profiles

## Extracting usage models from log files

Miroslav Janeski

September 2014


TU Delft
Delft University of Technology

TU/e Technische Universiteit Eindhoven University of Technology

UNIVERSITY OF TWENTE.

# Customer Profiles

Extracting usage models from log files

Eindhoven University of Technology
Stan Ackermans Institute / Software Technology

**Partners**



Embedded Systems Innovation by TNO — Eindhoven University of Technology

**Steering Group**

Piërre van de Laar
Jan Schuddemat
Ad Aerts

**Date**

September 2014

Contact Address
Eindhoven University of Technology
Department of Mathematics and Computer Science
MF 7.090, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
+31402474334

Published by
Eindhoven University of Technology
Stan Ackermans Institute

Printed by
Eindhoven University of Technology
*UniversiteitsDrukkerij*

ISBN
**978-90-444-1312-0**

Abstract

This report summarizes the "Customer Profiles" project, which goal was to obtain insight into the actual usage of systems. The approach of the project was to extract and visualize models from log files. The models capture the typical and atypical behavior of different users, both humans and other equipment. The stakeholders need these models to keep in control of the production environment in the high-tech factories that is getting more and more complex. The complexity increases among others due to the automation of the production process and involvement of more systems. The project resulted in a prototype that supports extracting customer profiles, a portable architecture according to which the prototype was implemented, a comprehensive domain analysis that includes solutions to the most common problems in extracting models for the purpose of customer profiles, and a list of suggestions how to achieve better models.

Keywords
log analysis, process mining, resource tracing, pipeline, model-based testing, MBT, TNO, ESI, ASML, Software Technology, PDEng, Tue, Trace

Preferred reference
M. Janeski, Customer Profiles: Extracting usage models from log files. Eindhoven University of Technology, SAI Technical Report, November, 2014. (978-90-444-1312-0)

Partnership
This project was supported by Eindhoven University of Technology and by Embedded Systems Innovation by TNO.

# Foreword

Embedded Systems Innovation by TNO (TNO-ESI) performs applied research in close cooperation with its industrial and academic partners. Our industrial partners observe a number of trends, such as equipment becoming a node in a network, i.e., an ASML TwinScan in a Fab, a Philips MRI scanner in a hospital, and a Thales radar system on a frigate in a battle fleet; and an increase of complexity needed among others to increase performance and to remove the human from the loop. These industrial trends increase the risk that a development organization loses insight in the customer profiles: the actual context and usage by the different customers of the equipment. Our academic partners have recently developed techniques to increase the insight in the actual usage of a system by analyzing the log files. Examples of these techniques are process mining and extracting of execution views. Based on this industrial problem statement and the available academic expert know-how, we defined, in close cooperation with ASML, a project that resulted in the report you are currently reading. The project was executed by a PDEng candidate of the Stan Ackermans Institute of the Eindhoven University of Technology under supervision of TNO-ESI at ASML. The candidate, Miroslav Janeski developed a pipeline to extract graphs and models from log-files that can be easily adopted to fit the context of the different industrial partners of TNO-ESI, such as Philips, Océ, and NXP. These graphs and models increase the understanding of and ability to communicate how the equipment is actually used. Furthermore, the requirements of a logging architecture that includes multiple nodes in a network, to obtain customer profiles have become clearer. This led amongst others to suggestions for improvement of the ASML's current architecture. TNO-ESI is pleased with the project's results and currently is planning the next steps. TNO-ESI hopes that you enjoy reading this report.

August 2014

*Piërre van de Laar*
*Research Fellow*
Embedded Systems Innovation by TNO

# Preface

This report summarizes the “Customer Profiles” project that was executed by Miroslav Janeski under supervision of the Embedded System Innovation by TNO (TNO-ESI) at ASML. The project was a full-time, nine-month graduation assignment in the context of a two-year technological designer program in Software Technology. This post-master program is offered by the Eindhoven University of Technology under the patronage of the Stan Ackermans Institute. The project goal was to obtain insight into the actual usage of systems by analyzing log files.

This report is intended for anyone who is interested in applying process mining techniques to obtain insight into the actual usage of a system by analyzing log files. The project introduction is stated in the first chapter “1. Introduction.” The significance of the project comes from the fact that the project was executed with a clear business goal: A meaningful overview of the system behavior in a production environment. The core of the project is the comprehensive domain analysis stated in the chapter “4. Domain Analysis.” It is recommended for every reader because it emphasizes the main issues and solutions in applying process mining for extracting customer profiles. For further information in applying process mining should focus on the chapters: “3. Problem Analysis”, “10. Verification & Validation”, and “11. Conclusions.” The results that are elaborating the actual capabilities of the “Customer Profiles” are emphasized in the chapter “10. Verification & Validation.” The recommendations for more efficient logging and process mining are stated in the chapter “11. Conclusions.”

The readers interested in the proposed portable architecture and the system design should focus on the chapters: “6. System Requirements”, “7. System Architecture”, “8. System Design”, and “9. Implementation.” In these chapters the stakeholder requirements and the requirements from the domain analysis are incorporated into appropriate system architecture and design that result in an implemented prototype.

One of the challenges in this project was the project management, mostly because of the varieties of stakeholders but also because of the level of the uncertainty in terms of delivering sufficient results. Therefore, the readers interested in the project execution, project management, and project retrospective should focus on the chapters “2. Stakeholder Analysis”, “5. Feasibility Analysis”, “12. Project Management”, and “13. Project Retrospective.”

Miroslav Janeski
September 2014

# Acknowledgements

The “Customer Profiles” project altogether was a lifetime experience for me. Leaving my comfort zone for nine months and joining a team of people that daily solves cutting edge problems to deliver better and better results for their customers was an incredible experience. Delivering a successful project in such an environment could not have been successful without the close collaboration and supervision from several persons.

First of all, I sincerely appreciate the members of my steering group. I thank Piërre van de Laar for constantly guiding me and supporting me during the project time. Thank you Piërre for your critical thinking, for letting me explore, and for the selfless sharing of your knowledge and experience. I appreciate Jan Schuddemat for his advices how to survive in the nowadays business world. Finally, I thank Ad Aerts for his objective and constructive advices. Ad Arts provided a non TNO-ESI and non ASML feedback, which helped in delivering better and more objective project.

During the project I collaborated with the members of the Themis and Magenta projects. Jan Tretmans, Richard Doornbos, and Jack Sleuters thank you for your selfless guidance. Your questions, comments, and feedback helped in defining and delivering a successful project. Our regular lunches and coffee meetings helped me enjoy my internship.

The project was executed at ASML. A lot of people from ASML were involved in this project directly and indirectly. First of all, I want to thank to Roger Lahaye for his guidance in the world of ASML. I learned a lot from your entrepreneurial attitude. Without your support and help my presence at ASML would not have been possible. During the project time I met a lot of people at ASML. I want to thank to Marc Verdiesen, Denie van Kleef, Sebastian Eigenmann, Lily Janssen – Shen, Hank Donker, Roland de Kruijff, and Jan Brekelmans for answering my questions and sharing their experience and domain knowledge.

The “Customer Profiles” project is a result of the knowledge and experience that I have gained during the PDEng programme. Therefore, I want to thank to all the people involved in the programme.

Last, but certainly not least, I want to thank to my family and close friends. A special thank you to my fiancé Brankica. Thank you for your love and your patience. I appreciate my sisters Mimi and Viki, their families, and my father for their unquestionable support during the entire programme. I want to thank you to my dear friends Meggie and Wim Doezé for being true friends and providing me a pleasant stay in the Netherlands.

Miroslav Janeski
September 2014

# Executive Summary

The project “Customer Profiles” is executed under supervision of the Embedded System Innovation by TNO (TNO-ESI) at ASML. The project was a full-time, nine-month graduation assignment in the context of a post-master program in Software Technology offered by the Eindhoven University of Technology. The project goal was to obtain insight into the actual usage of systems by analyzing log files. The project resulted with a prototype, a portable architecture, domain analysis, and suggestions how to improve the process of extracting customer profiles.

The most important project artifact is the prototype that shows the feasibility of applying process mining and resources tracing techniques to obtain insight into the actual usage of a system by analyzing log files. The prototype supports set of different activities such as: data collection, data preprocessing, information extraction, and information aggregation that work together to obtain a customer profile model that express the typical and atypical behavior of the participants in production environment as captured in the log files, which defines the prototype output. The validation phase has shown that the prototype output exceeds the stakeholders’ expectations. ASML profited from the prototype output and TNO-ESI will reuse the approach for different customers.

The success of the prototype output lead to a new requirement: a portable system architecture. Therefore, as a part of the project a portable system architecture that supports extracting customer profiles was designed. The architecture is based on the Pipes and Filters architectural pattern. The system architecture and design are a result of a broad architectural and system analysis, which balances between the stakeholder requirements and the most common practices in the software architecture and software development. As a part of the architecture, components that support different functionalities such as: Data Source, Event Parser, Event Enricher, and Event Combiner were designed.

A lot of domain knowledge was gained during the project. The domain knowledge was transformed into a comprehensive domain analysis. The domain analysis contains the most common aspects of applying process mining for extracting customer profiles such us: mapping issues, missing information, and the minimal log data requirements. As a part of the domain analysis an evaluation of the process mining algorithms was performed. The evaluation showed that the heuristics miner and the genetic miner are the most appropriate process mining algorithms for extracting customer profiles.

In order to improve the process of extracting customer profiles a list of suggestions was created. The suggestions focus on the most common problems in the logging infrastructures and in the process mining techniques. One of the suggestions is conscious manufacturer decision on the log file content. The manufacturer should define the ratio, the context (based on the minimal log data requirements), and the scope of the logging infrastructure. Another important suggestion for the logging infrastructure is having unique identifiers across the entire logging domain. The next suggestion advocates logging infrastructure on use case (end user activity) level. The last, but not the least suggestion is consistent accurate and standardized timestamp in the logging infrastructure. During the project experiments it was detected that the maturity level of the process mining tools is not on an appropriate level for industrial usage.

# Table of Contents

**Foreword** ..... **i**

**Preface** ..... **iii**

**Acknowledgements** ..... **v**

**Executive Summary** ..... **vii**

**Table of Contents** ..... **ix**

**List of Figures** ..... **xiii**

**List of Tables** ..... **xv**

**1. Introduction** ..... **1**

*1.1 Context* ..... *1*

*1.2 TNO-ESI* ..... *1*

*1.3 Customer Profiles* ..... *2*

*1.4 TNO-ESI @ ASML* ..... *3*

*1.5 ASML* ..... *3*

*1.6 Outline* ..... *5*

**2. Stakeholder Analysis** ..... **7**

*2.1 Main Stakeholders* ..... *7*

*2.2 TU/e* ..... *8*

*2.3 TNO-ESI* ..... *8*

*2.4 TNO-ESI Customers* ..... *8*

*2.5 ASML* ..... *8*

*2.1 ASML customers* ..... *9*

**3. Problem Analysis** ..... **11**

*3.1 Context* ..... *11*

*3.2 Testing Trends and Customer Profiles* ..... *11*

*3.3 Customer Profiling* ..... *12*

*3.4 Extracting Customer Profiles* ..... *13*

**4. Domain Analysis** ..... **15**

*4.1 Process Mining* ..... *15*

*4.2 Applying Process Mining* ..... *18*

*4.3 Process Miner Evaluation* ..... *23*

*4.4 Design Opportunities* ..... *26*

**5. Feasibility Analysis ..... 29**
*5.1 Issues and Challenges ..... 29*
*5.2 Risks ..... 29*

**6. System Requirements ..... 31**
*6.1 Requirement Analysis ..... 31*
*6.2 Functional Requirements ..... 31*
*6.3 Non-functional Requirements ..... 33*
*6.4 Main Use Case ..... 34*

**7. System Architecture ..... 35**
*7.1 Introduction ..... 35*
*7.2 Architecturally Significant Requirements ..... 35*
*7.3 Architectural Reasoning ..... 35*
*7.4 Pipes and Filters ..... 36*
*7.5 System Architecture ..... 37*
*7.6 System Component Overview ..... 40*
*7.7 Component Portability Level ..... 41*

**8. System Design ..... 43**
*8.1 Introduction ..... 43*
*8.2 Data Source Design ..... 43*
*8.3 Event Parser Design ..... 44*
*8.4 Event Enricher Design ..... 45*
*8.5 Collecting Missing Information ..... 46*
*8.6 Event Combiner Design ..... 47*
*8.7 Mxml Serializer Design ..... 48*
*8.8 Trace Transformer Design ..... 48*
*8.9 Ready-made Components ..... 48*

**9. Implementation ..... 49**
*9.1 Introduction ..... 49*
*9.2 Prototype ..... 49*
*9.3 Decisions ..... 49*
*9.4 Prototype Healthiness ..... 51*
*9.5 Deployment ..... 52*

**10. Verification & Validation ..... 53**
*10.1 Validation ..... 53*
*10.2 Verification ..... 59*

**11. Conclusions ..... 61**
*11.1 Results ..... 61*
*11.2 Suggestions ..... 62*
*11.3 Lessons Learned ..... 63*
**12. Project Management ..... 65**
*12.1 Introduction ..... 65*
*12.2 Work-Breakdown Structure (WBS) ..... 65*
*12.3 Project Planning ..... 65*
*12.4 Project Management Techniques ..... 66*
**13. Project Retrospective ..... 69**
*13.1 Reflection ..... 69*
*13.2 Design opportunities revisited ..... 69*
**Glossary ..... 71**
**Bibliography ..... 73**
**About the Author ..... 75**

# List of Figures

Figure 1 TNO-ESI Role ..... 1
Figure 2 Envisioned vs. Actual Usage ..... 2
Figure 3 TNO-ESI Customers ..... 3
Figure 4 TNO-ESI @ ASML ..... 3
Figure 5 Schematic control loop for on-product overlay improvement ..... 4
Figure 6 ASML Lithography Evolution ..... 5
Figure 7 Stakeholder Interests ..... 7
Figure 8 Evolution of the Testing Techniques ..... 11
Figure 9 Virtual Fab Concept ..... 12
Figure 10 Process Mining ..... 15
Figure 11 Process Mining Techniques ..... 16
Figure 12 Diagnostic Data Categorization ..... 16
Figure 13 Real Process Model ..... 17
Figure 14 Standard Log Model ..... 17
Figure 15 Mapping-rule Repository ..... 19
Figure 16 Missing Information Solution ..... 20
Figure 17 Complete Event Instances ..... 20
Figure 18 Missing Information Problem ASML ..... 21
Figure 19 Extended Real Process Model ..... 21
Figure 20 Extended Standard Log Model ..... 22
Figure 21 Run-in and run-out effect ..... 23
Figure 22 Heuristics Net Model Evaluation ..... 25
Figure 23 Scattered Infrastructure Risk Evaluation ..... 30
Figure 24 Project Main Use Case ..... 34
Figure 25 System Architecture for Extracting Customer Profiles ..... 37
Figure 26 Multiple Data Source System Architecture ..... 39
Figure 27 System Architecture Trace ..... 40
Figure 28 System Component Classification ..... 41
Figure 29 Component Portability Classification ..... 41
Figure 30 Data Source Variations ..... 44
Figure 31 Event Extractor Variations ..... 44
Figure 32 RegExp Event Parser Design ..... 45
Figure 33 Event Enricher Design ..... 46
Figure 34 Event Enricher Internal Design ..... 46
Figure 35 Multiple Incomplete Data Sources ..... 47
Figure 36 Collecting Missing Information Design ..... 47
Figure 37 Mxml Serializer Design ..... 48
Figure 38 Trace Architecture with Lookup Table ..... 48
Figure 39 Implemented Architecture ..... 50
Figure 40 Prototype Healthiness ..... 51
Figure 41 Dotted Chart - Production Process ..... 54
Figure 42 Dotted Chart - Zoomed In View ..... 54
Figure 43 Dotted Chart - Relative Time View ..... 55
Figure 44 Log File Source A - Process Model ..... 55
Figure 45 Log File Source B - Process Model ..... 55
Figure 46 Log File Source A&B - Process Model ..... 56
Figure 47 Final Data Set Heuristics Net (Sources A,B,&C) ..... 56
Figure 48 Complex Heuristic Net Model (Sources A,B,&C) ..... 57
Figure 49 Item Usage in the Production Process ..... 58
Figure 50 Customer Profiles - Project Roadmap ..... 65
Figure 51 Analytics Use Case Selection ..... 67
Figure 52 Burn down Chart - Infrastructure Design ..... 67

# List of Tables

Table 1 - TU/e Stakeholders ..... 8
Table 2 - TNO-ESI Stakeholders ..... 8
Table 3 - ASML Stakeholders ..... 9
Table 4 - Heuristics Net Model Evaluation ..... 24
Table 5 - Genetic Miner Results ..... 26
Table 6 - Architectural Patterns and Modifiability Tactics ..... 36
Table 7 - Event Execution Paths ..... 58
Table 8 - Planned Activity Distribution ..... 66
Table 9 - Final Activity Distribution ..... 66

# 1.Introduction

A short introduction to the project is given, along with the Embedded Systems Innovation by TNO (TNO-ESI) and ASML in which it took place. The main activities at TNO-ESI and their special role in the industry are elaborated. An introduction to ASML is given, because the project was conducted at ASML, as one of TNO-ESI's clients. This chapter ends with a short overview of the content of this project.

## *1.1 Context*

The "Customer Profiles" project is conducted by Miroslav Janeski as a part of his Professional Degree in Engineering (PDEng) in software technology (ST) education. The PDEng in ST is a two-year post-Master technological designer program, which prepares the candidates for an industrial career as a technological designer, and later on, as a software or system architect. The program consists of two parts: in-house and on-site part. The in-house part consists of several intensive training periods and in-house projects and the on-site part is a final operational project in industry. The final project has to challenge the candidate to deliver a high quality solution based on his designing competences.

The project is initiated by Embedded Systems Innovation by TNO (TNO-ESI), and is part of the current work of TNO-ESI at ASML. One of the project goals is to support the efforts of TNO-ESI at ASML. The high level goal is to provide a general architecture and infrastructure for extracting customer profiles of high-tech equipment running in a production environment.

## *1.2 TNO-ESI*

TNO-ESI collaborates in an open innovation structure with a wide range of industrial and academic partners, helping their partners staying ahead of the innovation curve and lead innovations in embedded systems technology.

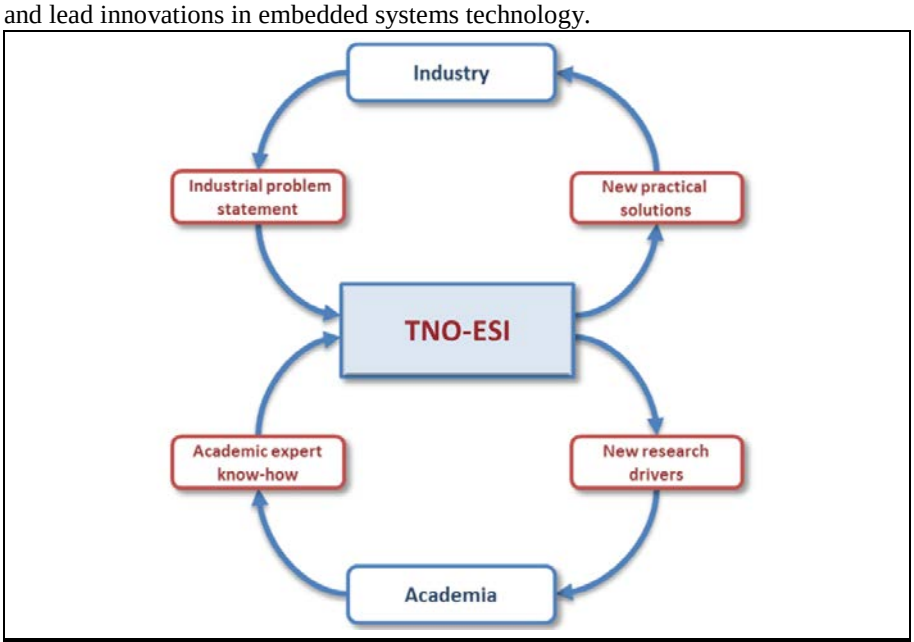


Figure 1 TNO-ESI Role

TNO-ESI plays a significant role in building a bridge (Figure 1) between industry and academia. From one side, it brings new practical solutions from the academia to the industry via academic expertise. On the other side, it brings new research drivers from the industry to the academia via industry problem statements.
To advance the fundamental basis of high-tech product design and improve on its efficiency, effectiveness, quality, and costs, TNO-ESI (Key Competence Areas, 2014) focuses on the following key professional competencies:

- system architecting
- system design
- system integration & testing
- model-based engineering

## *1.3 Customer Profiles*

The customer profiles present models of the behavior of particular equipment in a production environment. With customer profiles one can describe the actual usage of particular equipment. In general, the goal of the customer profiles is to minimize the gap between the envisioned usage of the product in the development process and the actual usage of the same product in a production process. By minimizing the gap between the envisioned and the actual usage (Figure 2), the stakeholders can accomplish a variety of benefits. For instance, with better design and architecture evaluation, better testing and validation, better risk reduction and diagnostics, TNO-ESI's clients can design products that are much closer to the end user needs. At the same time better understanding of the process in the production environment can bring new product and service opportunities.

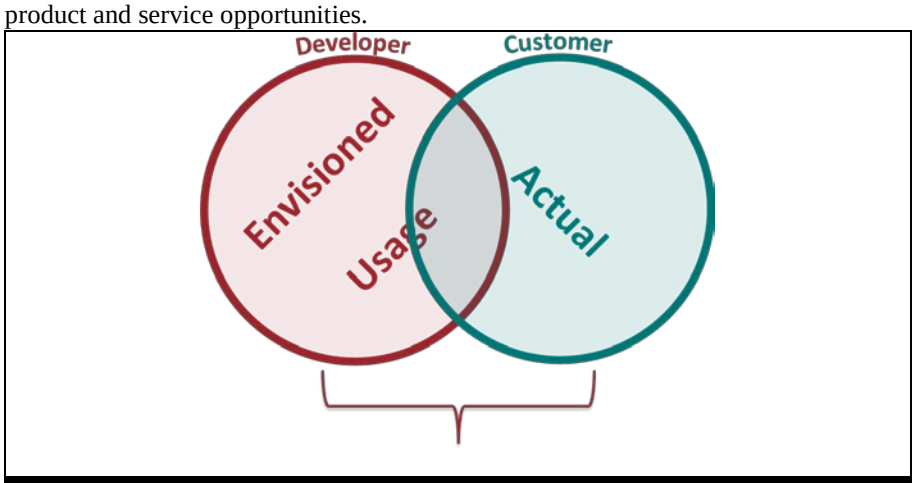


Figure 2 Envisioned vs. Actual Usage

TNO-ESI will use the expertise of generating customer profiles to provide better services to their customers. In general, wherever there is a high-tech product in a production environment, TNO-ESI can support its customers to get better insight into the actual usage of the products. From this point of view, potential TNO-ESI customers (Figure 3) are: ASML, Philips, Océ, and many more.
There are many ways to generate a customer profile. One way is by applying process mining (van der Aalst W., 2011). Process mining is a process management technique that performs analysis of processes based on event logs. The basic idea is to extract knowledge from event logs recorded by an information system. In the context of this project, process mining is used to extract knowledge in terms of behavior models from multiple log data sources. In addition, communicating the process mining requirements can provide suggestions for logging mechanism improvement. Thereby more information can be extracted from the logging files in the future.

Figure 3 TNO-ESI Customers

The complexity of applying process mining techniques can be explained with the fact that the customer profiles have to describe a behavior of systems of systems. TNO-ESI clients manufacture a collection of dedicated systems that work together to create a new, more complex system which offers more functionality and performance than simply the sum of the constituent systems. An ASML high-volume manufacturing factory, a hospital running high scale Philips medical systems, and Océ's complex management systems are examples of complex system of systems where the customer profiles can be applied.

## *1.4 TNO-ESI @ ASML*

One of the partners where TNO-ESI applies their professional competencies including the customer profiles is ASML. Part of the collaboration between TNO-ESI and ASML related with this project are two ongoing projects: Themis and Magenta. Both projects are conducted at the Architecture, Testing, Integration, and Quality Group, which is a part of the Application Development and Engineering Sector.

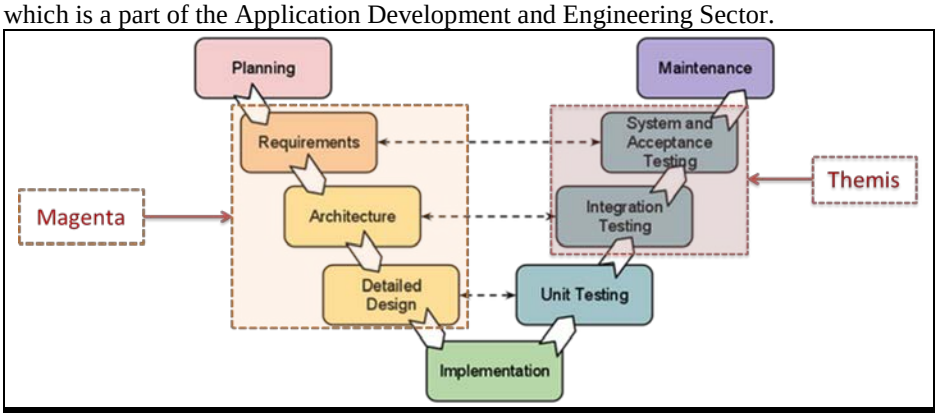

Figure 4 TNO-ESI @ ASML

From the software development point of view these projects support the two directions of the V-model (Figure 4): verification and validation. Magenta is oriented towards system architecting and architecture evaluation, whereas Themis is oriented towards system integration and testing, specifically applying model-based testing.

## *1.5 ASML*

### 1.5.1. Introduction

ASML is the world's leading provider of lithography systems for the semiconductor industry, manufacturing complex machines that are critical to the production of integrated circuits. ASML designs, develops, integrates, markets, and services advanced systems used by customers – the major global semiconductor manufacturers – to cre-

ate chips that power a wide array of electronic, communications, and information technology products. ASML advanced systems integrate multiple disciplines in the semiconductor manufacturing process such as

- Photolithography
- Metrology
- Computational & Holistic Lithography

The integration of ASML's vast knowledge of the photolithography and the computational lithography process, and the expertise in metrology and process control with YieldStar, enables comprehensive lithography system set-up for high volume manufacturing.

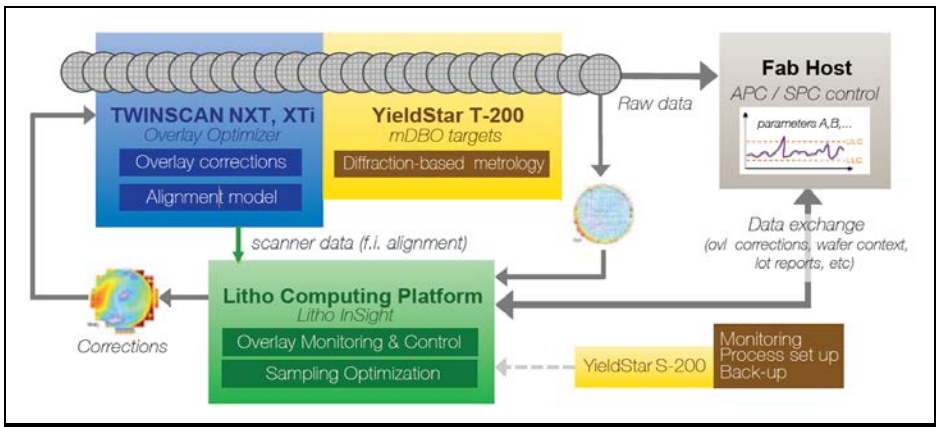


Figure 5 Schematic control loop for on-product overlay improvement

## 1.5.2. ASML Holistic Lithography

The semiconductor industry is driven by "shrink" – the ability to make the features that make up chips ever smaller. Shrink improves chip performance and increases manufacturers' profitability. However, as chip features get smaller, so do the tolerances that manufacturers must work to. The smaller the tolerances, the harder it is to manufacture chips that work properly.

Moving to 20-nm chip wafers and beyond brings unique challenges for the semiconductor industry. From the production point of view, this means it will no longer be enough to consider Integrated Circuit (IC) design, mask creation, lithography and metrology in isolation. Instead it needs a new integrated approach to IC manufacture. The ability to optimize multiple IC production steps simultaneously is fundamental to ASML's Holistic Lithography vision. A schematic overview of ASML's Holistic Lithography approach (Images - ASML's customer magazine, 2013) is shown in Figure 5. Figure 6 depicts the evolution of the ASML lithography process. The conclusion is that the lithography process is getting more and more complex from pioneering to holistic approach. Based on Figure 6, one can conclude that now ASML is in the phase of implementing a holistic approach. Holistic Lithography is an intelligent integration of ASML's vast knowledge of the scanner itself, ASML's expertise in metrology and process control with YieldStar and the Litho Computing Platform, and computational lithography. This enables further shrink by optimizing process windows and lithography system set-up for high volume manufacturing.

ASML's Holistic Lithography approach includes different ASML equipment that supports the holistic approach and that increases the ability to control lithography scanners. Nowadays, the high volume manufacturing process is performed in a fully automated fab environment. A fully automated fab environment consists of different machines from different manufacturers that work together to achieve a high level goal. Putting the end user in full control of the process, but also putting ASML in a full support of the process takes a lot of knowledge and understanding. The understanding of the production process is essential because every customer has its own production process, although they use the same or similar ASML equipment.

Understanding the actual process in the customer production environment and modeling the actual customer usage of the ASML equipment is one example where customer profiles can be applied. The new approach in this project is that the extraction

of the customer profiles is based on analysis of log files created from different equipment (participants) in a production environment.

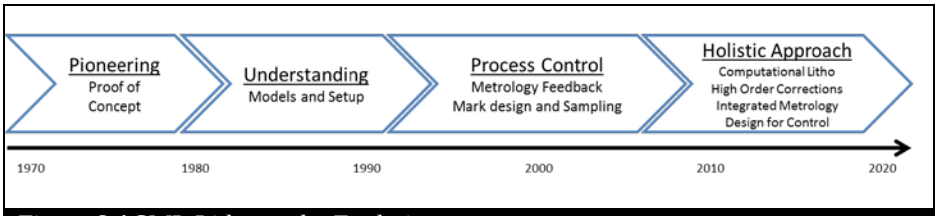

Figure 6 ASML Lithography Evolution

## *1.6 Outline*

The report starts by introducing the reader to the different stakeholders in the project, together with a brief overview of all their motivations and needs. Following this, a problem analysis is elaborated in a top-down approach, from the question "Why do we need customer profiles?" to the question "How to accomplish them?"

After the problem analysis, a comprehensive domain analysis is given. The domain analysis is focused on main challenges in applying process mining to a custom domain. This is meant to allow an easier understanding of the content that follows, which focuses on the problem to be solved.

Next, the feasibility analysis is placed. The feasibility analysis focuses on the main issues and challenges expected in the project. This chapter also includes a risk list, along with mitigating measures.

Following the feasibility analysis, there are chapters describing the architecture and design of the prototype, including a summary of the system requirements that drive the architecture and the design. The most important design decisions and the architectural reasoning are elaborated in these chapters. After this, more practical issues that arose during development are addressed.

Next, the verification and validation analysis is elaborated. It focuses on the definition of the customer profiles and how the prototype was validated and verified. Finally, there are chapters dealing with the conclusions, where the results and suggestions are elaborated; project retrospective focusing on the personal experience of the author of the project; and an overview of how the project management was handled.■

# 2.Stakeholder Analysis

A brief overview of all stakeholders in the project is given, and their motivations and needs are explored. An overview of the projects and departments interested in the "Customer Profiles" is presented.

## *2.1 Main Stakeholders*

Understanding the stakeholder relations and their concerns is an essential step in managing and implementing a successful project. What makes this project special is the variety of stakeholders, their relations, concerns, and expectations.
Based on their main interests in the project, there are three groups of stakeholders:

- Stakeholders interested in the process of delivering the final artifact
- Stakeholders interested in the quality attributes of the final artifact
- Stakeholders interested in the final artifact

Eindhoven University of Technology (TU/e) belongs to the first group of stakeholders, TNO-ESI belongs to the second group, and TNO-ESI clients such as ASML, but also their customers belong to the third group of stakeholders.
In order to better understand the stakeholder stakes a distribution of their interests is shown in Figure 7. It can be concluded that TU/e is mostly interested in the project implementation process and in the quality attributes of the project artifact. TNO-ESI is mostly interested in the coverage of the functional requirements and the quality attributes, whereas ASML and the ASML customers main interest is in the implementation of the functional requirements, which is the project artifact.

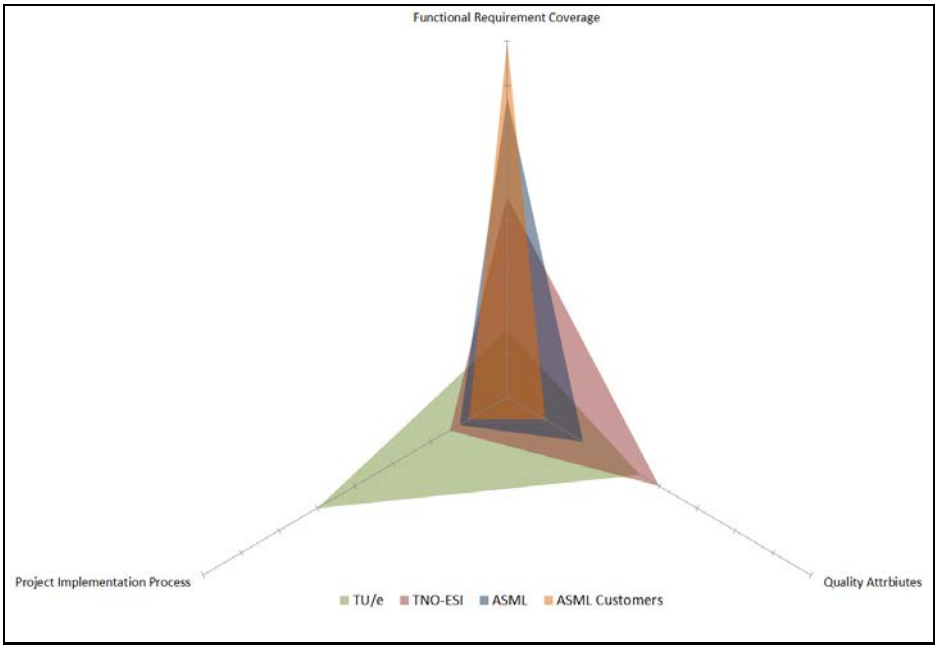


Figure 7 Stakeholder Interests

## 2.2 TU/e

TU/e has an interest in the design and in the process of the project execution. Therefore, the TU/e stakes include project management, analysis, design, implementation, verification, and validation. Table 1 shows the TU/e stakeholders. Ad Aerts is a general director of the PDEng programme in ST and he is a university supervisor in this project. Miroslav Janeski is the PDEng candidate responsible to implement this project.

Table 1 - TU/e Stakeholders

| Stakeholder | Role |
|---|---|
| Ad Aerts | University Supervisor |
| Miroslav Janeski | PDEng Candidate |

## 2.3 TNO-ESI

TNO-ESI has the role of project owner and therefore is directly interested in the project artifacts. Driven by the core idea of open innovation and delivering domain-independent solutions, TNO-ESI expects that the final artifact can be reused in various domains.

The relation between TNO-ESI and ASML in the context of this project is elaborated in the previous chapter. The motivation of TNO-ESI for this project is to support their two ongoing projects at ASML, Themis and Magenta. The final artifact of this project, a prototype that extracts customer profiles, will be used as input for the model-based testing infrastructure provided with Themis and the architectural evaluation provided with Magenta. The main stakeholders from TNO-ESI are listed in Table 2.

Table 2 - TNO-ESI Stakeholders

| Stakeholder | Position | Project |
|---|---|---|
| Jan Schuddemat | Senior Project Manager | Themis; Magenta |
| Jan Tretmans | Research Fellow | Themis |
| Piërre van de Laar | Research Fellow | Themis |
| Richard Doornbos | Research Fellow | Magenta |
| Jack Sleuters | Senior Software Architect | Magenta |

## 2.4 TNO-ESI Customers

The largest group of stakeholders that benefits from the project artifact are the TNO-ESI customers (Figure 3). TNO-ESI customers can use the customer profiles for more efficient testing (business wise testing), better design (design optimization), faster diagnostics, and new product opportunities.

## 2.5 ASML

ASML is a member of the TNO-ESI Customers stakeholder group. At the same time, ASML is the domain owner and a lot of design stakeholders come from ASML.

The main stakeholders from ASML are listed in Table 3. Roger Lahaye, an Interoperability Test Architect, is the main stakeholder for the “Themis” project. Denie van Kleef is a Usability and Interoperability Architect and is involved in the general customer profiling. Lily Janssen Shen is a project leader of the current “Customer profiling” project at ASML. The “Customer profiling” project has the same goal as this project, but uses a different approach based on interviews, machine configuration, and factory specification. Lily Janssen Shen is interested in combining the results from both projects (“Customer Profiles” and “Customer Profiling”.) The Software Testing Group at ASML, which is led by Ronnie van‘t Westeinde, is also interested in the results of this project, mostly for the same reasons, providing business-wise selection of the test cases and test case discovery. Pieter Knelissen is a System Architect and is concerned with the project vision.

| Table 3 - ASML Stakeholders | |
|---|---|
| Stakeholder | Position |
| Roger Lahaye | Interoperability Test Architect |
| Denie van Kleef | Usability and Interoperability Architect |
| Lily Janssen Shen | Project Leader - CSI |
| Pieter Knelissen | System Architect |
| Roland de Kruijff | Cross-sector Structural Improvement |
| Hank Donker | Cross-sector Structural Improvement |
| Ronnie van‘t Westeinde | Software Architecture and Integration Test Group Leader |

## *2.1 ASML customers*

Figure 7 shows one more group of stakeholders. This group consists of the ASML customers. Some of the ASML customers provide the log data for this project have the role of data supplier. They are not directly involved in the project, but they will benefit from the project results in many ways such as factory optimization and more efficient diagnostics.■

# 3.Problem Analysis

The problem analysis starts with the question why do we need customer profiles in generic fashion and ends with the problem analysis that answers the question “How to extract customer profiles?”

## *3.1 Context*

Along with the technology growth, the complexity of the systems that high-tech companies produce grows too. Building large software-intense systems includes high level integration skills and significant development effort. Sometimes, the development of such machines includes a lot of legacy design and backward compatibility which makes the end design very complex.

The main reasons for profiling the customer usage are to understand the actual usage and to analyze the behavior of the equipment in a factory environment. With the understanding of the ongoing process in the factory environment, the companies can benefit in many ways such as: improved design of its equipment, more efficient diagnostics, factory optimization, and opportunities for new products.

If we take the ASML products as an example, a factory that uses ASML equipment represents a system of complex systems. Therefore, all the ongoing processes in the factory are complex by default. They are performed by different subsystems and supervised by teams of highly qualified engineers. Based on this, the whole process of profiling the customer usage of the high-tech equipment and particular ASML equipment is not trivial at all.

## *3.2* Testing Trends and Customer Profiles

To better understand the need of customer profiles, the trends in the testing models and the Virtual Fab concept are explained.

### 3.2.1. Model-based Testing

In Figure 8, the evolution of the testing techniques is given. The latest trend in the testing techniques is model-based testing. Model-based testing is application of model-based design for designing and optionally also executing artifacts to perform software testing or system testing.

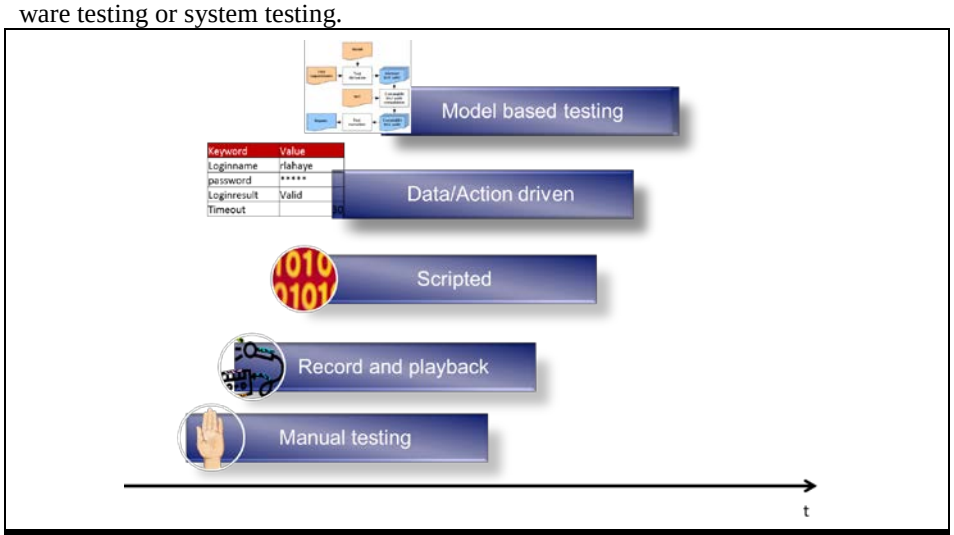


Figure 8 Evolution of the Testing Techniques

The model-based testing produces test cases as a variation of the model and the data. Not every variation of the test model is relevant and not every variation has the same business value in terms of money and time. The number of test cases can be unlimited. The customer profiles can help in selecting the most useful test cases in a production environment. The reason for including the customer profiles in model-based testing is to support the selection of the most appropriate sequences for testing. This means that the customer profiles will offer more optimized model-based testing. A useful test case is a test case that describes the most frequently used scenarios in the system. At the same time the customer profiles can add a new test case, which describes a usage scenario in a production environment that is not a part of the testing framework. By selecting the most used test cases and by adding new test cases, the customer profiles provide more efficient testing.
Applying model-based testing in a combination with customer profiles to optimize the model-based testing is one of the best practices in the modern testing frameworks. ASML is always using the best practices to develop high-tech equipment and TNO-ESI supports ASML's application of model-based testing to achieve best practices.

### 3.2.2. Virtual Fab Concept

Another trend in the modern testing approaches is virtualization. With virtualization companies can have much cheaper testing of their equipment. For instance, if ASML wants to simulate a typical customer environment, it will cost an enormous amount of money. With the virtualization, a complete system can be simulated at low cost. At the same time the virtualization offers flexibility and more possibilities. Different equipment, virtual or real, can be plugged in and different test cases can be executed in a different test case.
The combined effort of TNO-ESI and ASML to apply virtualization led to the Virtual Fab concept. Figure 9 shows a part of Virtual Fab concept. The role of "Customer profiles" in the Virtual Fab concept is to extract models of the typical and atypical behavior of different participants in a production environment. With other words, the customer profiles are an abstraction of the fab usage. In addition, the customer profiles can be transformed into virtual fab scenarios. At the same time, virtual fab scenarios can be generated based on the design models of the system. In that case, the customer profiles should provide selection of the most used virtual fab scenarios. The role of the customer profiles in the virtual fab concept is the same as in the model-based testing approach: selecting of the most used virtual fab scenarios, but also adding new virtual fab scenarios that cannot be extracted from the system design models.

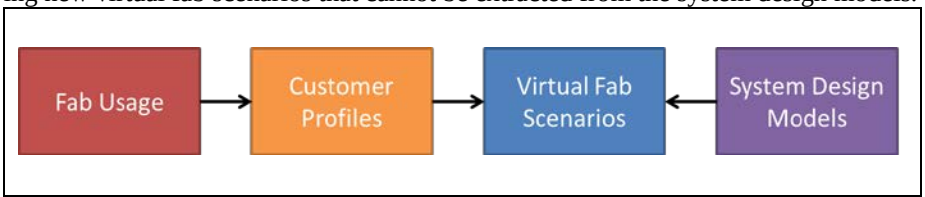

Figure 9 Virtual Fab Concept[1]

## *3.3 Customer Profiling*

Creating customer profiles is not a new activity at ASML. Currently there are a few ongoing projects trying to obtain customer profiles. These projects differ from this project in the input data. In order to obtain customer profiles, they consider different inputs such as: diagnostics, interviews, logging, and machine configurations.
The diagnostics approach is only focused on tracing the root cause for the problems. It gives low level (machine level) insight into the problem and is not enough to create the full picture of the customer usage. The interviews give better insight into the customer usage, but they are subjective and incomplete. A lot of engineers are involved in the production environment and they are responsible for different tasks. Their perspective of the production environment often brings inconsistent and incomplete results. The logging approach so far was used to generate a static model of the produc-

[1] The diagram shows part of the Virtual Fab concept that is relevant for "Customer Profiles"

tion environment, which does not describe the behavior of the participants in the production environment.
The results from the these projects are exceeding the expectations ,but the ongoing process in the production environment is getting more and more complex and looking at only one aspect does not give the full picture. At the same time the ASML expectations are increasing based on their customer needs. In this situation, where other approaches have difficulties in obtaining a full description of the process, process mining techniques can help. First, because process mining techniques can extract the main ongoing process, which comprises a process structure and a process behavior model, and second, because the process mining techniques give a more high level (use case) view of the problem.

## *3.4 Extracting Customer Profiles*

The stakeholders acknowledged that data is available and that gives opportunities to apply different data analysis techniques to reach a specific business goal. Extracting customer profiles is such a business goal based on data analysis with process mining. Extracting customer profiles based on process mining consists of several subproblems that make the process challenging. The main challenge is mapping the entities from the customer specific domain (such as the ASML logging infrastructure) to a process mining domain. For the purpose of this project the process mining domain is presented with the Standard Log Model (elaborated in the following section) and an appropriate mapping from the customer specific domain to the Standard Log Model. From more abstract point of view, an opportunistic problem solving approach is applied in this project.
The process mining challenges and solutions are presented in the following chapter. Extracting customer profiles consists of intermediate data analysis. It involves analysis of the implemented logging infrastructure and analysis of the stored historical data. In case of missing data, changes in the data collection strategies have to be proposed. The data analysis is a process that is driven by the stakeholders and their interests. In this project is the phase where the process mining techniques and resource tracing techniques are applied in the ASML domain. The high uncertainty in terms of different business goals, which require different analytical skills, is the main characteristic of the data analysis. The results of this step are used as an input for the software development phase elaborated in the chapters “6. System Requirements”, ”7. System Architecture”, “8. System Design”, and “9. Implementation”.■

# 4.Domain Analysis

The process mining domain is elaborated and an overview of the main challenges in applying process mining is given. In addition, the high level design opportunities in the project are emphasized.

## *4.1 Process Mining*

Process mining (van der Aalst W., 2011) is a process management technique (Figure 10) that allows for the analysis of processes based on event logs. The basic idea is to extract knowledge from event logs recorded by a set of systems. Process mining provides techniques and tools for discovering process, control, data, organizational, and social structures from event logs. This approach is often used when no formal description of the process can be obtained by other approaches, or when the quality of existing documentation is questionable.

In the context of customer profiles, we have the same assumptions as process mining, because of the similarity of the problem that we have to solve. The assumptions are that the TNO-ESI clients manufacture high-tech equipment (Figure 5) that is used in a production environment i.e. the real world shown in Figure 10. There is a system that controls the process and records events in log files. In extracting customer profiles, we extract relevant information from the log files and apply process mining techniques to generate process models that abstract the real world.

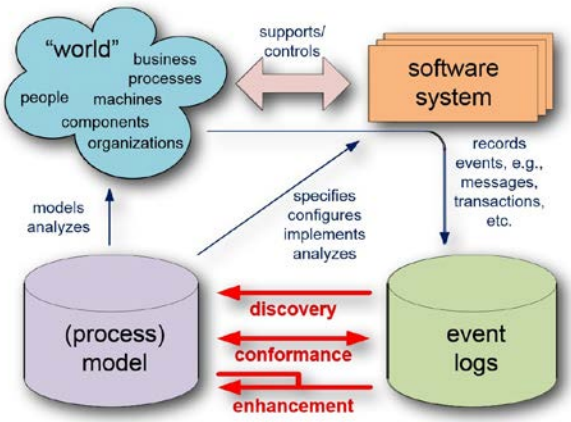


Figure 10 Process Mining

### 4.1.1. Process Mining Techniques

There are three classes (Figure 11) of process mining techniques (van der Aalst W., 2011). This classification is based on whether there is a prior model and, if so, how it is used.

a. Discovery: There is no a priori model, i.e., based on an event log, a process model can be discovered.
b. Conformance analysis: There is an a priori model. This model is compared with the event log and discrepancies between the log and the model are analyzed.
c. Enhancement: There is an a priori model. This model is extended with a new aspect or perspective, i.e., the goal is not to check conformance but to enrich the model.

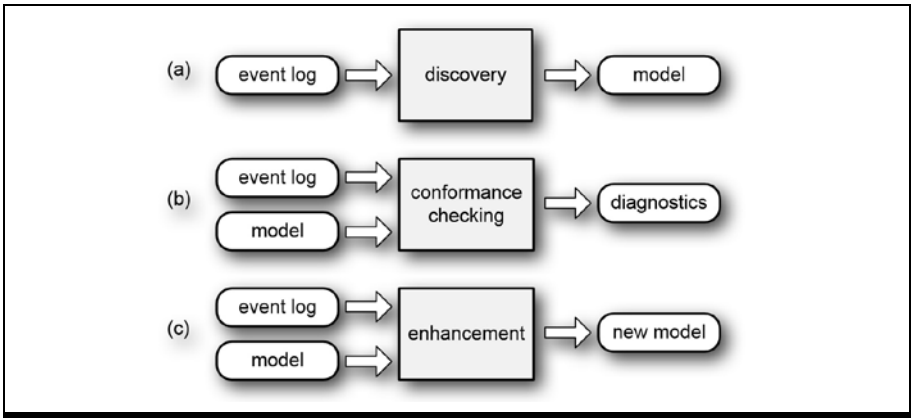


Figure 11 Process Mining Techniques

In the context of this project, a combination of conformance checking and enhancement is applied to model the customer process and to provide an insight into the customer environment. Therefore, we assume that there is an a priori model that we want:

- to confirm that is in the log files
- to extend with additional information

### 4.1.2. Process Mining Concepts

The process mining technique is based on several assumptions (van der Aalst & van Dongen, 2005) about the log files. It assumes that it is possible to record event instances such that the event instance refers to

- an event and
- a process instance

Most large software systems use logging mechanisms to record and store information of their specific activities into log files. In practice, logging messages are recorded with mechanisms that are designed to collect data for purposes different from process mining. Regardless of the logging mechanisms, it is common that developers, testers, and other specialized users use logging information for understanding, debugging, testing, and corrective maintenance. This immediately shows one of the biggest challenges (van der Aalst & van Dongen, 2005) faced during applying process mining. When trying to use event logs from a different system to do process mining, we need to be able to present the logs in a standardized way, i.e., there is a need for a good description of such a log. Furthermore, for each information system, a mapping to that description has to be provided.

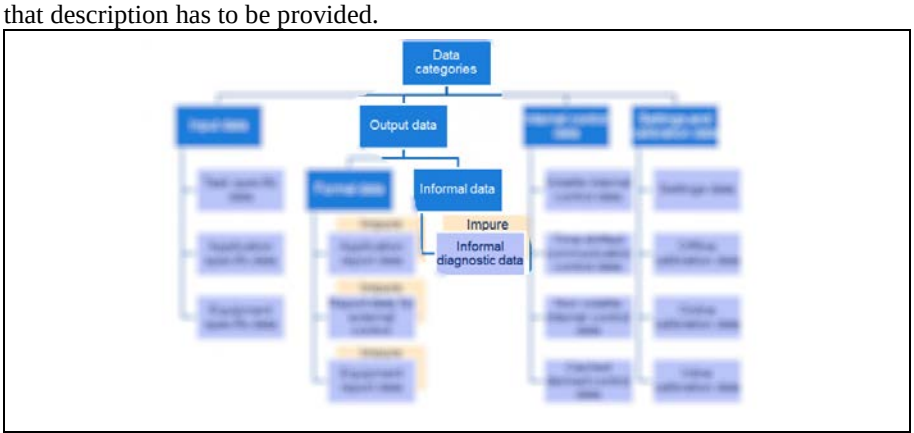


Figure 12 Diagnostic Data Categorization[2]

Therefore, the mapping of event logs from one system to the standard format is not a trivial task. The mapping issue can be even more difficult when the development teams adopt different logging formats, naming conventions, and even different log-

[2] The rest of the figure is blurred for confidentiality

ging mechanisms. Another reason that makes the mapping issue a real problem is that the logging data is considered as informal data. Figure 12 shows the classification of the diagnostic data (logging data) at ASML as informal data. The fact that the diagnostic data is informal and unstructured means that is designed for human analysis, which usually means is semi-structured or structureless data that complicates the automated analysis.

Mapping any informal data model to a standardized model is a challenging process. In order to understand the challenge, we have to define two models: a Real Process Model and a Standard Log Model. The Real Process Model is shown in Figure 13. It consists of two entities: a process and an event.

- Process – A collection of related, structured activities or tasks that produce a specific service or product (serve a particular goal) for a particular customer.
- Event - Activity or a task that was executed in a process.

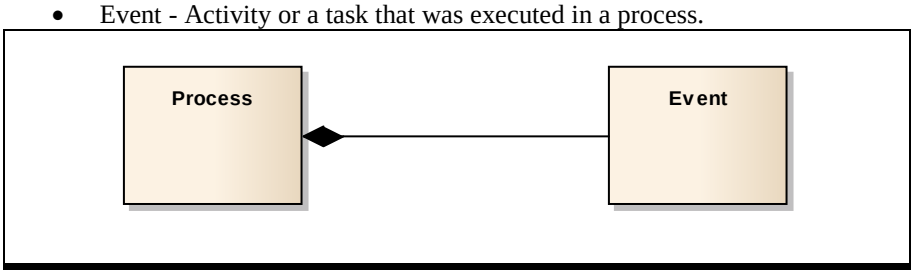


Figure 13 Real Process Model

The Standard Log Model, shown in Figure 14, models the data in the log files. There are several relevant models that are used as reference models (Hage, Malaisé, Segers, Hollink, & Schreiber, 2011) (van der Aalst & van Dongen, 2005). The Standard Log Model entities are instances of the Real Process Model entities:

- ***Event Instance*** is an instance of an *Event* entity in the real process model.
- ***Process Instance*** is an instance of the *Process* in the real process model. A process instance is an execution of a real process.

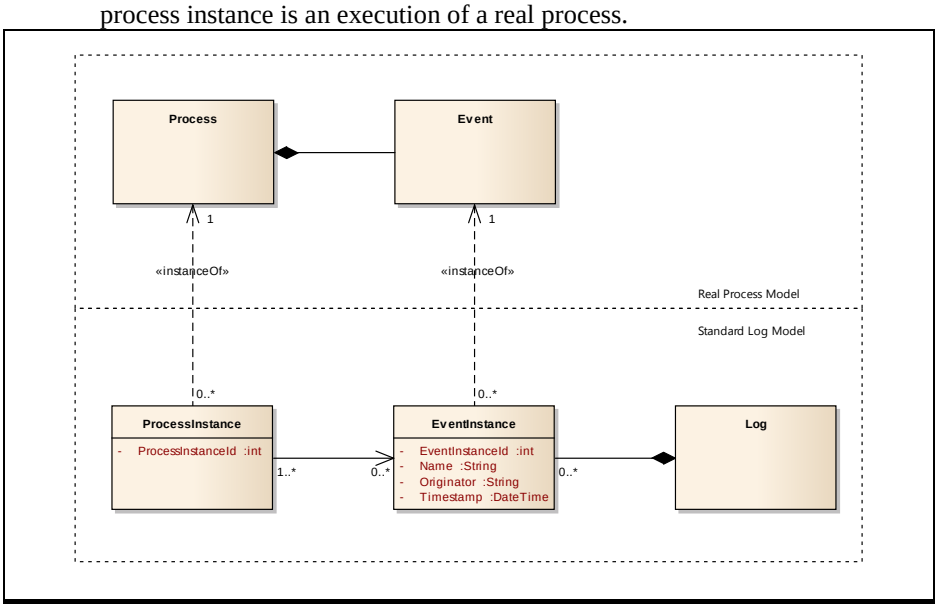


Figure 14 Standard Log Model

At the same time the Standard Log Model defines the minimal information that the log data have to have in order to apply process mining. In (Rozinat, Data Requirements for Process Mining, 2012) and (van der Aalst & van Dongen, 2005) different data requirements are discussed. For the purpose of this project and based on the communication with the stakeholders, we had to extend their requirements with additional information required for customer profile extraction. The data requirements for extracting customer profiles with process mining are the following:

- ***Event Instance Unique ID:*** Each event instance should have a Unique Event Instance Identifier in order to distinguish different event instances.

- ***Process Instance Unique ID:*** Is necessary to distinguish different executions of the same process.
- ***Event Instance Class:*** Each event instance should have information about the Event Class to which belongs.
- ***Event Instance Originator:*** Each event instance should have an originator. An Originator is a participant (user or equipment) in the production environment (a factory). One of the goals of customer profiles is to model the behavior of different participants in the production environment. One way to distinguish events executed by different participants is by ***Event Instance Originator.***
- ***Event Instance Timestamp:*** At least one timestamp is needed to bring the event instances in the right order. Another way to understand the order of the event instances is by sequence Id.
- Each ***Event Instance*** belongs to at least one process instance. This information has to be explicit in the log files in form of ***Process Instance Unique ID***.

## *4.2 Applying Process Mining*

Process mining techniques in this project were applied in several steps

1. Identifying a process (and events)
2. Solving the mapping challenge
3. Solving the missing information challenge
4. Solving the Non-Unique IDs problem
5. Dealing with the run-in and run-out effects

The first two steps are already recognized as challenges in the literature. In the context of extracting customer profiles, we realized that there are three more important challenges: missing information, having non-unique IDs, and dealing with the run-in and run-out effects. In the following subsections each of these steps is explained.

### 4.2.1. Identifying Process and Events

As we already mentioned, feeding process mining algorithms with raw log data usually does not provide sufficient results for the stakeholders. That is only possible when the logging infrastructure design satisfies the process mining requirements, which is almost never the case.

The first step in applying process mining for customer profiles was selection of a major representative process in the production environment. More precisely, we need to map the real process and events with the Real Process Model (Figure 13). If we take a more abstract view, all of the TNO-ESI clients have a product that has a main use case. These products are used in an automated production environment. In a sense, the products are nodes in a system of systems. These systems pool their resources and capabilities together to create a new, more complex system which offers more functionality and performance. Whether it is the ASML TwinScan, Océ's printers, or Philips MRI scanners, they all have main use case scenario, they are used in a production environment (respectively: an automated Fab, a document management system, or a hospital), and they are connected with other participants. Therefore, for the purpose of this project, the process that we try to conform to and to enhance with process mining is the main production process. Together with the process that we have to model, we have to identify the events that compose the process. In general, there might be an enormous amount of events, and in that case only the most important for the stakeholders are selected.

There might be different ongoing processes in the production environment. They might be completely independent or they might be sub-processes of the main production process. For different processes, different data requirements for extracting customer profiles might be needed. For instance, if there is only one selected participant in the production environment the stakeholders might not be interested in the ***Event Instance Originator***.

Selecting the main production process as a process that we have to extract or conform to is a design decision made in this project. The main production process is selected because of two reasons. The first reason is the genericity, each TNO-ESI client has a

product that is used in a production environment. The second reason is that is a good starting point for the customer profiling and it confirms to the stakeholder interests. The stakeholders can quickly get an overview of the real usage and they easy detect deviations from the expected usage.

### 4.2.2. Mapping Rule Repository

After identifying the real process and the events that comprise the process, we had to map event instances and their properties from the log files to the entities in the Standard Log Model. The mapping challenge in process mining is already recognized (van der Aalst & van Dongen, 2005) and there are recommendations (Callo Arias, America, & Avgeriou, 2013) on how to solve it. One way to implement this is by using a mapping-rule repository (Callo Arias, America, & Avgeriou, 2013). A Mapping-rule Repository (Figure 15) is a set of formal and informal specifications that describe how event instances from the log files can be mapped to the events in the real process model. The specifications are derived from the domain where the customer profiles are applied. Therefore, in the solution proposed within this project, there has to be an engine to bind domain specific mapping rules. This leads to one of the system design requirements. The proposed architecture must have an Extractor component. The role of the Extractor is to extract event instances that are compliant with the Standard Log Model.

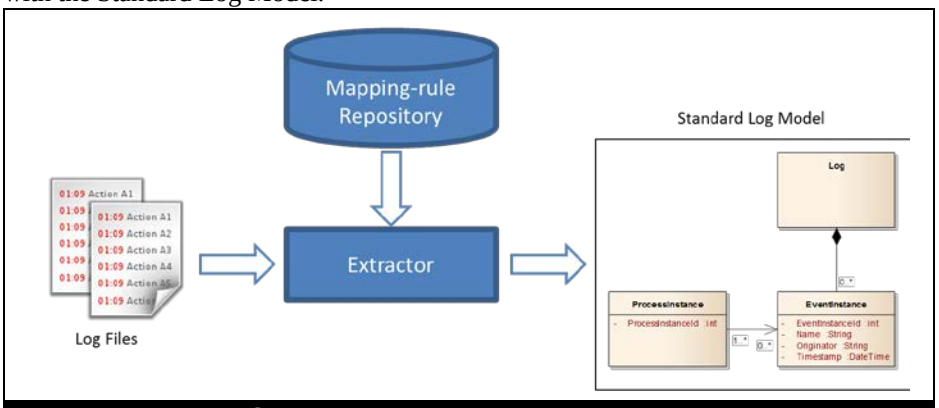


Figure 15 Mapping-rule Repository

### 4.2.3. Missing Information

In this project we faced an important challenge that goes beyond the complexity of the mapping challenge. The mapping-rule approach assumes that in the log files we have the minimal information necessary for process mining techniques. That means that for each event instance in the log files there is information that can be mapped to the Standard Log Model requirements. By using the mapping-rule approach we use different mapping rules to enable 1:1 mapping between the event instances in the log files and the Event Instance class in the Standard Log Model.

During the project, we realized that in the context of ASML there are log files that miss information. There are event instances in the log files but they do not fully satisfy the minimal requirements that we have defined in the subsection “4.1.2. Process Mining Concepts.” Usually, the event instances that do not have 1:1 mapping with the Standard Log Model are considered as incomplete event instances and they are filtered out. In the context of ASML, the incomplete event instances represent significant events in the production process. Filtering out incomplete but significant event instances will affect the quality of the customer profiles. Moreover, filtering out incomplete but significant event instances results in a process model that does not meet the stakeholder expectations. This is explained in the section “10.1. Validation.”

Therefore, we searched for a mechanism that enriches these incomplete event instances with additional information and transforms them into event instances that are compliant with the Standard Log Model. The solution we choose is to use an enricher to make this incomplete event instance compliant with the Standard Log Model. The enricher will use the data from a Complementary Information Repository. The repository includes a set of formal and informal specifications (as in the Mapping-rule

Repository). The procedure is illustrated in Figure 16. Again, this leads to additional system design requirement. In the proposed architecture there must be an Enricher component that by using complementary information will enrich the extracted incomplete event instances. The incomplete event enrichment depends on the missing information. Therefore, the design and implementation of the enricher is rather domain specific than generic.

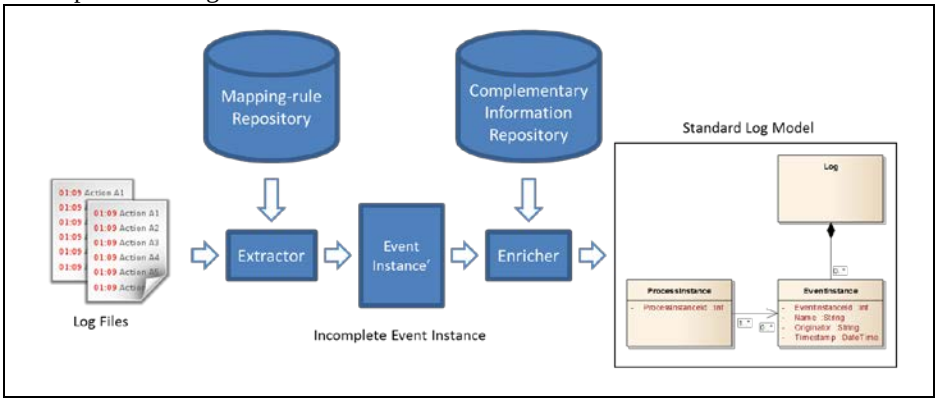


Figure 16 Missing Information Solution

In the case of ASML, we faced event instances without ***Process Instance ID***. That means that there are (incomplete) event instances that we do not know to which process instance they belong. From this perspective there are two different log file sources:

- Log files with event instance with minimal information (*Event Instance*)
- Log files with event instances with missing information (*Event Instance'*)

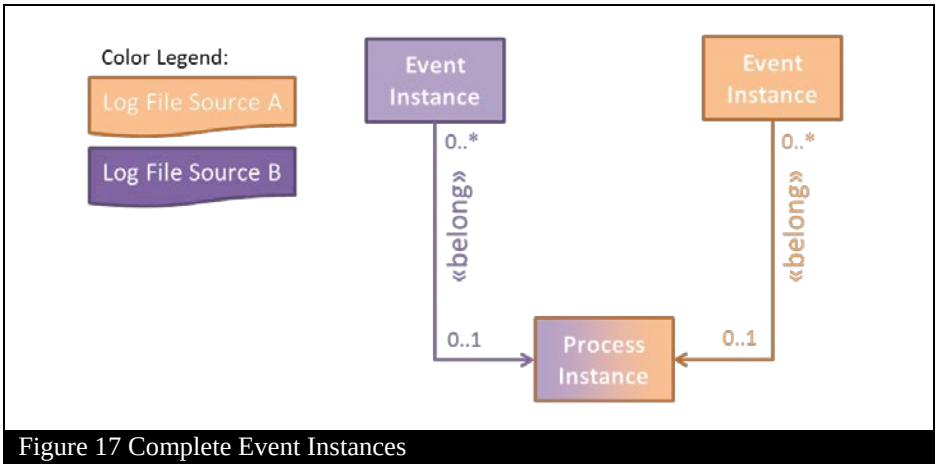


Figure 17 Complete Event Instances

Figure 17 shows an example of two data sources at ASML. Log File Source A and Log File Source B have complete event instances that belong to same process instances. This makes the event instances compliant with the Standard Log Model.
In the example of ASML, there is one more log file source: Log File Source C. Unlike to Log File Source A & B, Log File Source C does not have complete event instances. The problem is illustrated in Figure 18.

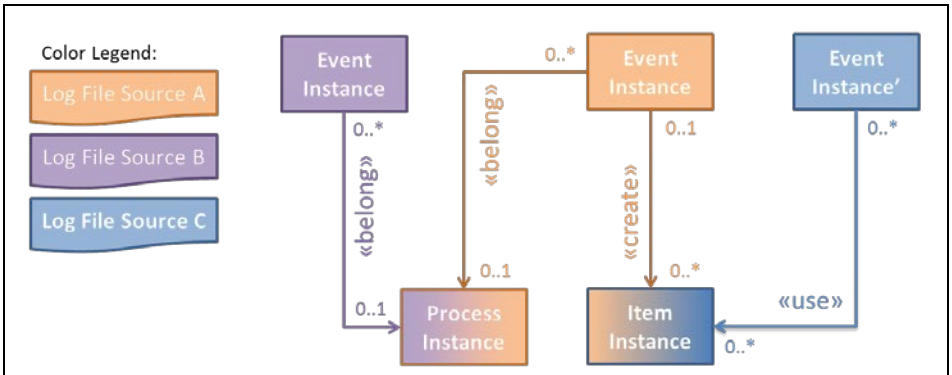


Figure 18 Missing Information Problem ASML

As we mentioned above, the logging architecture is designed for a different purpose than process mining. This means that a lot of information is stored in the log files. Fortunately, some of the information in the log files can help us to enrich the incomplete event instances. The Log File Source A in Figure 18 has event instances with minimal information (*Event Instance*) and for them we only need the Mapping-rule Repository. The relationship between the class Event Instance and the class Process Instance fulfils the requirements from the Standard Log Model. The same reasoning is valid for the Log File Source B. At the same time, in the Log File Source A there is information about the relation between the Event Instance and particular Item Instances that are created by these event instances. This information gives us coupes of Item Instance and Process Instance. The Log File Source C in Figure 18 has incomplete event instances (*Event Instance'*) without Process Instance ID. Moreover, the Log File Source C has extra information about a relation between these incomplete event instances (*Event Instance'*) and the same Item Instances created by the event instances from the Log File Source A. From the Log File Source A, we get complete event instances that created item instances and from Log File Source C we get incomplete event instances that use the same item instances from Log File Source A. By joining the incomplete event instances from Log File C with the item instances that they use, we enrich the incomplete event instances into complete event instances. The association between the Event Instances and created/used Item Instances is used to enrich the incomplete event instances (*Event Instance'*) in the Log File Source C. If we summarize, in the example of ASML we miss information about the join between certain event instances and process instances. In order to solve the join problem, we use a lookup table (associations between created/used item instances and process instances). For that reason the Real Process Model and the Standard Log Model are extended with the new entities *Item* and *Incomplete Event* and their appropriate instances. The extended Real Process Model is shown in Figure 19. Each *Process* in the final Real Process Model can have many *Events*, each *Event* can create many *Items* and each *Item* can be used by many *Incomplete Events*.

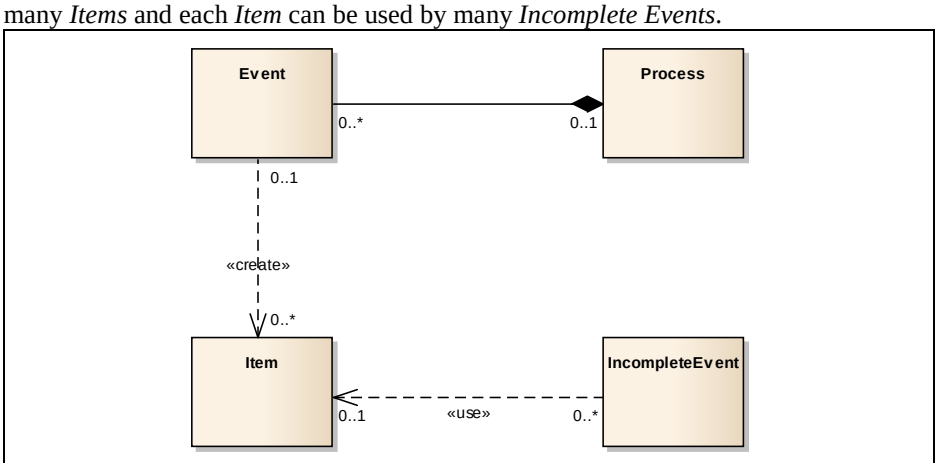


Figure 19 Extended Real Process Model

The Extended Standard Log Model with incorporated instances of the *Item* and *Incomplete Event* is shown in Figure 20. The Log can *have Incomplete Event Instances* that can be associated with a particular *Item Instance*. The *Item Instances* are created by a particular *Event Instance*.

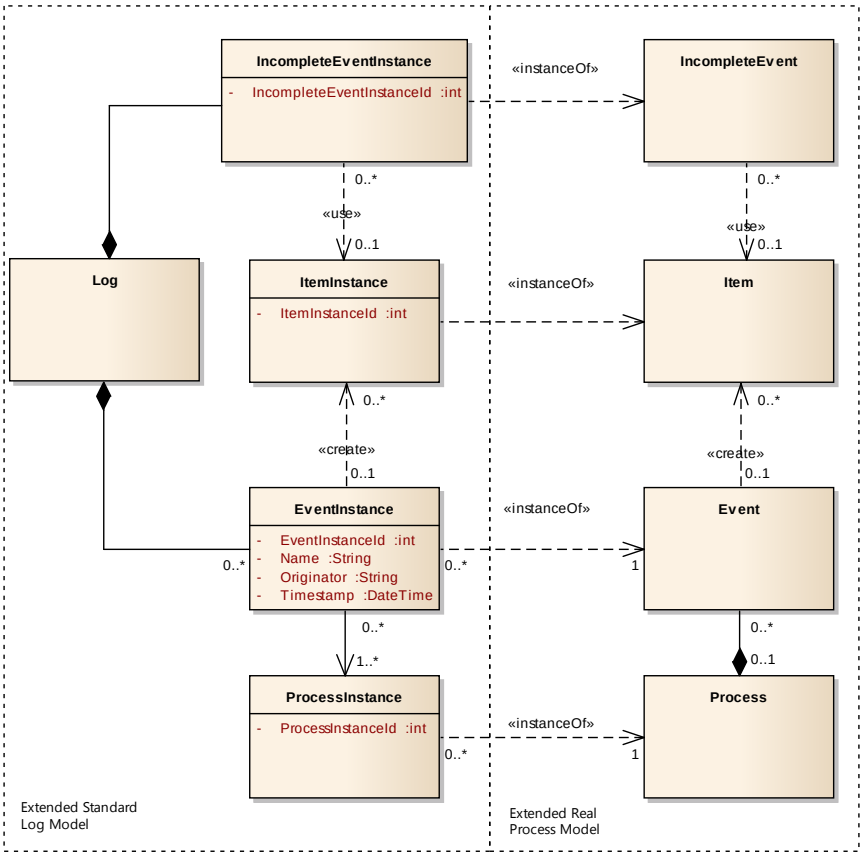


Figure 20 Extended Standard Log Model

### 4.2.4. Non-Unique IDs

One of the main data requirements stated in the subsection "4.1.2. Process Mining Concepts" is that the event and process instances must have Unique ID property. In the data analysis phase it was discovered that in a certain data set (of log files) there were process instances with same unique ID. In that case, we cannot always uniquely distinguish to which process instances the event instances belong. Therefore, it was necessary to apply an algorithm in order to enable this distinguishment. In the context of the observed data set, we applied a rule that says at the certain point of time only one process instance with a particular ID can be active. In that case, the new event instances are attached to the current active process instance. In order to implement this algorithm we had to introduce a mechanism for activation or deactivation of the process instances. One way to do this is to select significant event instances that activate and deactivate the process instances. With this approach we make sure that the event instances are attached to the correct process instance. This approach is not perfect and does not always guarantee correct answers. For instance, if there is no significant event instance because of the run-in or run-out effects there is no way to deactivate the current active process instance, and each next event instance will be added to one process instance.

### 4.2.5. Run-in and run-out effects

Run-in and run-out effects are effects when the analyzed log data represents a timeframe from the lifetime of a particular system. There are a lot of process instances that started before the observed timeframe and a lot of process instances that stopped after the observed timeframe. These process instances are incomplete and if they are not properly handled they add noise to the model. In the context of this project, each extracted process instance is being checked for the number and type of event instances that it has. If the process instance does not meet certain criteria such as: having a start event, having a stop event, and minimal number of events then is filtered out from further analysis. The start event, the stop event, and the minimal number of event instances are domain specific values and have to be defined by domain experts who are familiar with the observed process. Figure 21 depicts an example of the run-in and run-out effects. Based on the algorithm explained above, the process instance A and the process instance F are not a part of further analysis, because they do not meet the criteria. The process instance A does not have a start event instance in the observed timeframe and the process instance F does not have a stop event instance in the observed timeframe.

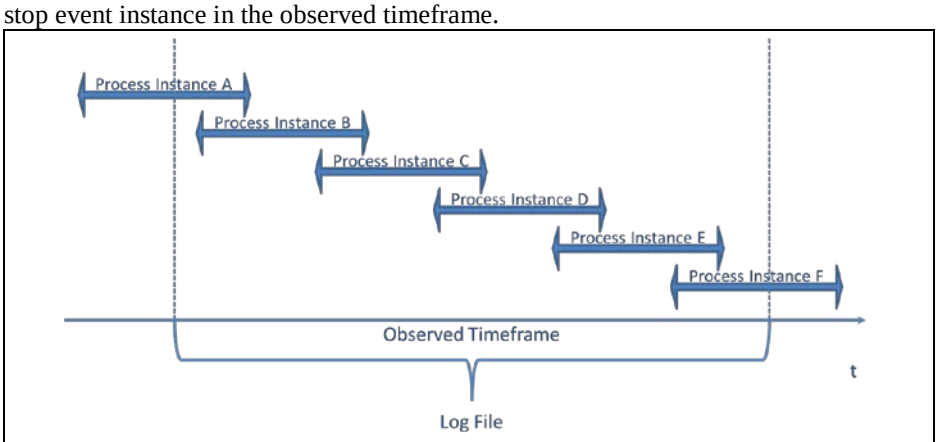


Figure 21 Run-in and run-out effect

### 4.2.6. Domain Specific Algorithms

In general, the problem of missing information and the non-unique IDs must be solved with domain specific algorithms. The domain specific algorithms include heuristics. Therefore, they do not always guarantee a correct solution. The algorithms based on a lookup table, like the one described above, guarantee exact matching, whereas others such as the solving the non-unique ID's problem do not always guarantee exact matching and can introduce an error in the results. For that reason, a mechanism that will validate the results based on domain algorithm has to be provided.

## *4.3 Process Miner Evaluation*

As a part of the domain analysis, a process miner evaluation was performed. The goal was to evaluate which process miner (process mining algorithm) gives the best results for extracting customer profiles.

In the process mining there are several process mining algorithms that give different structured process definition. The process mining algorithms considered in this project are: the alpha algorithm, the heuristics miner, the genetic miner, and the fuzzy miner. In the context of "Customer Profiles," two criteria of the algorithms are the most important:

- Semantics of the output
- Dealing with noise

Semantics of the output means what is the output of the model. This is important for the stakeholders because it gives them knowledge how to perceive the output. In a case of unclear meaning of the output, the output cannot be reused for further analysis. For instance, the alpha algorithm generates a Petri Net Model, the heuristics min-

er and the generic miner generate a Heuristics Net Model (with clear logical semantics), and the fuzzy miner does not have a clear semantics of the output.
The second criterion is dealing with noise. This is important for the purpose of this project, because in all the cases in the data analysis phase, real log files were used. The real log files are often incomplete and contain noise. Therefore, an algorithm that deals with noise is more appropriate for this project than the one that does not deal with noise. One of the requirements of the alpha algorithm is that it requires complete log file. Complete log file is a log file that has all the possible combinations of execution of a particular process. This requirement makes the alpha algorithm not appropriate for extracting customer profiles. The fuzzy miner can deal with noise, but the goal of the fuzzy miner is to provide hierarchical model of large and unstructured processes, which was not the case in the context of "Customer Profiles." The most suitable algorithms for extracting customer profiles are the heuristics miner and the genetic miner because they can deal with noise and they have clear semantic output.
The heuristics miner is based on the dependency among the events in the process and has several parameters that influence the algorithm results. The idea behind event dependency is elaborated in (Weijters, van der Aalst, & Alves de Medeiros, 2006) and is not going to be explained in this project. To emphasize the importance of the heuristic miner input parameters, three different Heuristics Net models were generated: Model A, Model B, and Model C. The models have same input log file but have different values for the following input parameters: Relative-to-best Threshold, Dependency Threshold, and the Number of Positive Observation. These parameters are related to the event dependency relation and play significant role in the miner output.
In order to evaluate the result, the proper completion (PC) fitness type is used as a comparison measurement. Proper completion is one (Rozinat, Alves de Medeiros, Günther, Weijters, & van der Aalst, 2008) of the metrics to evaluate the process mining algorithms. Table 4 shows the values for the input parameters and the proper completion (PC) value for each model. Higher value for the proper completion means that the generated model fits better to the input log file. One can easily conclude that by modifying the values for the input parameters, we can generate a Heuristics Net model that fits better to the input log file.

Table 4 - Heuristics Net Model Evaluation

| Model | Relative-to-best threshold | Dependency Threshold | Positive Observations | PC |
|---|---|---|---|---|
| Model A | 0.05 | 0.9 | 10 | 95.6% |
| Model B | 0.2 | 0.85 | 10 | 98.6% |
| Model C | 0.2 | 0.8 | 4 | 99.7% |

The Heuristics Net Models show abstraction of the process in log files based on the heuristics driven process mining algorithm. They can be used to express the main behavior (not all details and exceptions) registered in an event log. Figure 22 depicts the graphical representation of the Heuristics Net Models A, B, and C. The circles in Figure 22 represent events. The different color of the circles and the different numbering format of the event names represent different data source. The arcs in Figure 22 indicate the dependency between the events. The dependency measure indicates how certain we are that there is a dependency relation between two activities. A high value (close to 100%) means that there is a very high dependency relation between the connected events. The events can be generated by one or more participants. The models depicted in Figure 22 show process models generated from same input log files, but with different abstraction level. Model A has proper completion level of 95.6% and only one path, which means that the process model has single execution path:
Event I > Event 1 > Event 2 > Event II > Event 3 > Event III > Event IV > Event V > Event 4
The execution path covers 95.6% of the process instances in the log file. The difference between the Model A and the Model B & C is that Model B and Model C have fork events. We already emphasized that the Heuristics Net Models have clear semantics of the output. The fork events in this report are interpreted as a logical XOR function.

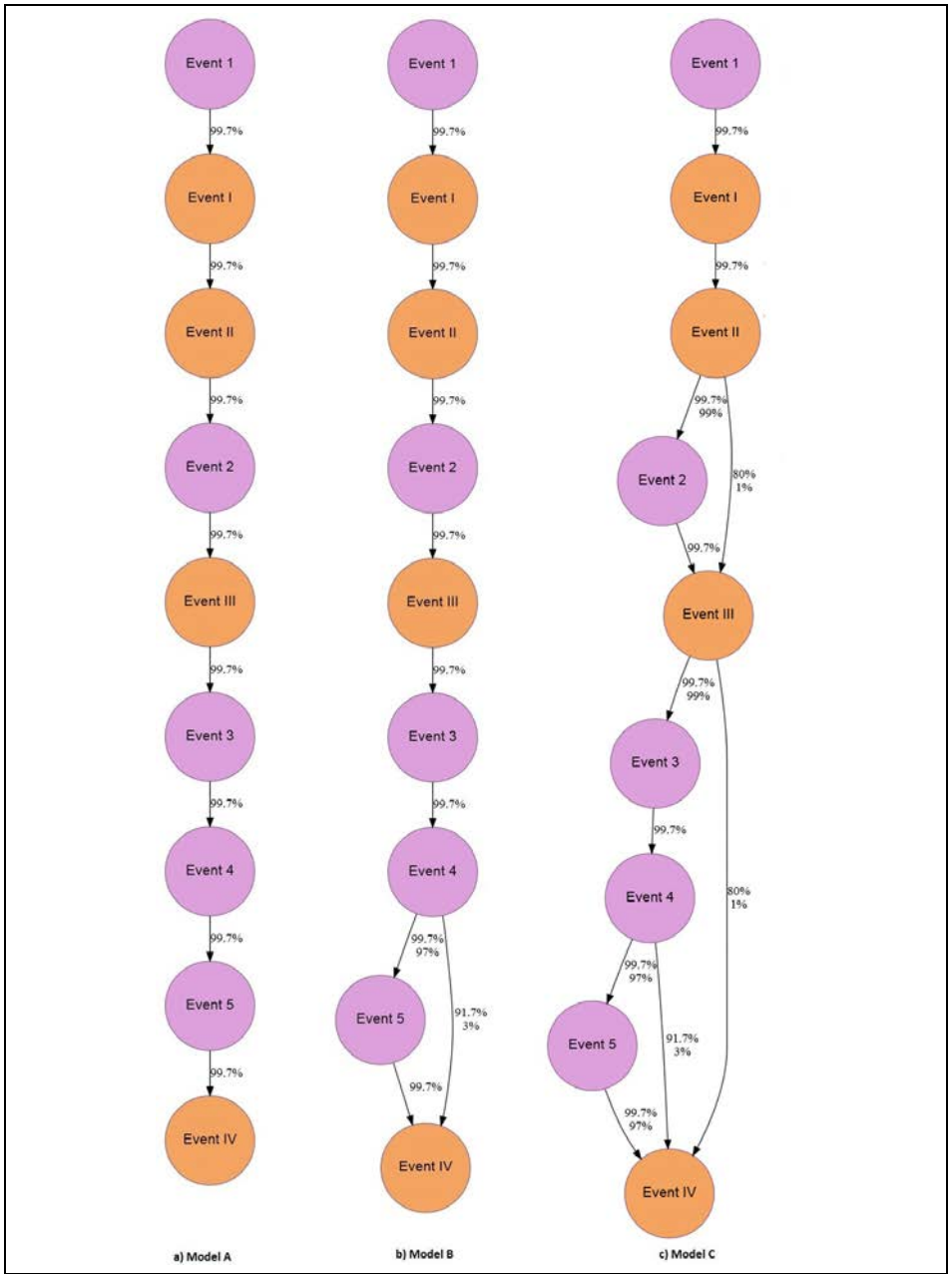

Figure 22 Heuristics Net Model Evaluation

Based on this, Model B has proper completion level of 98.6 % and two possible paths (XOR), which means that the process model has two execution paths:
Event 1 > Event I > Event II > Event 2 > Event III > Event 3 > Event 4 > Event 5 > Event IV
Event 1 > Event I > Event II > Event 2 > Event III > Event 3 > Event 4 > Event IV
An important thing about the Model B is that there are two numbers on the arcs from Event 4 to Event 5 and Event IV, respectively. The top number, as we already mentioned, describes the dependency measure. The bottom number shows in how many process instances Event 5 follows Event 4, and in how many process instances the Event IV follows Event 4. That means that the Model B has two execution paths with 97% and 3% coverage, respectively. In the same fashion the Model C covers 99.7% of the input log files and has 6 different execution paths with different coverage. The conclusion is that by modifying the input parameters for the heuristics miner, one can generate models with different abstraction level. It depends on the stakeholder to decide which model suits best to their needs. In one case single execution path that covers more than 90% of the input log file might be a sufficient result. In other case when the stakeholders need more detailed model or a model that has better log file

coverage then adjusting the input parameters is a necessity. A more detailed elaboration of the Heuristics Net Models is placed in the chapter "10. Verification & Validation."

The results from the heuristics miner can be satisfactory for the stakeholders, but the heuristics miner requires process mining domain knowledge. Another way to get similar results without knowing the process mining domain is by using the genetic miner. The genetic miner (de Medeiros, van der Aalst, & Alves de Medeiros, 2007) uses genetic approach to generate multiple models that deal with noise and incompletes. To verify the experiments the same input log file was used for the genetic miner and the proper completion was selected. The experiments showed that by using the generic miner, a list of different models with different proper completion level can be generated. In that case, the stakeholders have to select the most appropriate results from the generated list. The list is ordered by the proper completion level, with the model with the highest proper completion on the top. The resulting list is shown in Table 5. The top three results from the list correspond to the Model C, Model B, and Model A respectively. The rest of the models have significantly lower proper completion (<5%) that makes them completely inappropriate for the stakeholders. The results show that by using the genetic miner the same results as with the heuristics miner can be obtained without deeper knowledge about the dependency relation and the heuristics miner input parameters.

Table 5 - Genetic Miner Results

| Generated Process Model | Proper Completion |
|---|---|
| Model 0 | 99.7% |
| Model 1 | 98.6% |
| Model 2 | 95.6% |
| Model 3 | 4% |
| Model 4 | 2.9% |
| Model 5 | 1% |
| Model 6 | 0% |

The conclusion from the process mining evaluation is that the heuristics and genetic miner that can deal with the noise and incompleteness of the log files and can generate models with different abstraction of the real process. The heuristics miner requires process mining domain knowledge. At the same time the genetic miner requires better understanding of the process that has to be extracted but no extended process mining domain knowledge is necessary.

## *4.4 Design Opportunities*

A few characteristics were identified early on as necessary for a good design in the context of this project. First and foremost was the desire for a functional prototype that shows the benefit of extracting customer profiles. The prototype should show models of customer profiles extracted by applying process mining techniques on logged data in a production environment.

The design opportunities in this project, from a high level point of view, are defined by the problem decomposition and its components: data collection and data analysis. There are two main challenges, which at the same time are the main design opportunities. One of them is mapping the ASML domain based on the current logging infrastructure to the process mining domain. The other one is dealing with the missing information. At the same time, the mapping procedure should be domain independent, so TNO-ESI can reuse it for different clients.

Based on the elaboration above and the design criteria explained in (van Hee & van Overveld, 2012) the following three design criteria were selected as the most appropriate for this project:

- Genericity
- Realizability
- Impact

Genericity – This design criterion refers to the extent to which the prototype can be used in other situations. It has to answer to the question is the prototype reusable and portable in another context. Genericity as a design criterion is related with the TNO-ESI's core idea of open innovation and delivering portable solutions. Portable solutions are solutions that can be easily ported to another customer or domain. In the context of this project, the genericity of the artifact is described and presented with the portability quality requirement elaborated in the chapter "6. System Requirements."
Realizability – This design criterion answers to the question can the project artifact be made (technologically and economically)? In the context of this project, the technical realizability is considered. This is important because, the basis of the project is applying process mining techniques in specific domains for the purpose of customer profiles. Applying process mining in a specific domain has a certain amount of issues and challenges that have to be solved. Therefore, the technical realizability is a solid design criterion for this project.
Impact – In the context of this project, this design criterion, describes the impact that it has on the other projects and on the environment where is executed. Figure 9 shows the position of the "Customer Profiles" in the context of the Virtual Fab concept. As it is explained in the chapter "2. Stakeholder Analysis," different stakeholders can have different benefits and the benefits depend on the final project artifact.
These three design criteria are revisited in the chapter Project Retrospective. From the list of design criteria in (van Hee & van Overveld, 2012), two design criteria were selected as not relevant for this project. The first one is the inventiveness design criterion. It answers to the question "How new it is?" In this project we apply academic topic in an industrial context and we extend the academic practice of applying process mining with analyzing and combining multiple log file sources. That means the innovation is present in this project, but is not the main focus. The other design criterion is the complexity. It answers to the question does it acquire a complex structure? In the context of this project, we do not consider the complexity level of the project artifact, mostly because it is a proof-of-concept.■

# 5.Feasibility Analysis

The main issues and challenges expected during the project are presented. A few of them are detailed in other chapters as well. This chapter also includes a risk list, along with mitigating measures.

## *5.1 Issues and Challenges*

A list of reviewed issues that are recognized as the ones with high impact on the resulting artifact are described in this chapter.

### 5.1.1. Domain Knowledge

Gaining the proper level of domain knowledge in order to deliver successful results is a significant issue and a challenge at the same time. The domain knowledge integrates several scientific disciplines starting from the lithography sub-disciplines and semiconductor production process, to data collection, data preprocessing, and process mining. The resulting artifact is tightly coupled with the amount of domain knowledge gained and the speed of gaining domain knowledge.

### 5.1.2. Domain Abstraction

TNO-ESI is driven by the idea of providing portable solutions. That means that this project, as a part of TNO-ESI, should separate domain specific from generic problems and aspects.

### 5.1.3. Different Output

In the chapter “2. Stakeholder Analysis” different groups and types of stakeholders are introduced. Every group of stakeholders has different expectations from the project. For instance, the Themis project expects additional information from customer profiles for test case selection in model-based testing; the Magenta project expects customer profiles as behavior models; the Factory Integration Department wants to use customer profiles to recognize the participant behind the behavior. All these constraints influence the resulting artifact, and the alignment of the stakeholder expectations and requirements is a big challenge.

## *5.2 Risks*

A list of detected risks that would have high impact on the resulting artifact and mitigating measures that were put into place to limit their impact are described in this chapter.

### 5.2.1. Log Data Availability

One of the main risks in this project is the log data availability. The customer log data is the main input for this project; therefore the log data variety, velocity, and volume have high impact on the project results. There are two possible undesirable scenarios: either the log data is not collected at all or the data is collected but there is no access, due to different reasons. In the first scenario, the main scope of the project would be defining data collection strategy, which means providing infrastructure that will support the collection of the log data necessary for extracting customer profiles. In the second scenario, the main strategy would only consist of providing access to

the data, which includes elaborating the benefits to the stakeholders. In both cases, the project will have a different amount of effort spent on different activities.

### 5.2.2. Scattered Infrastructure

In the context of data analysis, a scattered infrastructure is an infrastructure that supports different business goals without a common basis. In the context of this project an example of a scattered infrastructure would be an infrastructure that has to support a set of very different use cases that have little in common. In that sense, it is a high risk because the project has a wide scope and the output of the project could be a set of different small and not integrated solutions, instead of a coherent solution.


Figure 23 Scattered Infrastructure Risk Evaluation

Figure 23 shows the risk evaluation and mitigation strategy of the risk of having a scattered infrastructure. The main cause for the risk is having very different use cases. The most appropriate preventive control steps are use case selection and role model use case. These two preventive steps were applied in this project and are elaborated in the chapter "12. Project Management." On the right side is the main impact of this risk: wide project scope and the recovery steps to be taken if the risk happens. The two main recovery steps are grouping of the common use cases and applying domain-based classification. These two steps can be combined in one or used individually. In general, the idea is to group the use cases based on the domain they belong to, so the project scope has an acceptable size. The result from this recovery step is that instead of a set of many different use cases, there is a smaller set of groups of use cases.■

# 6.System Requirements

The requirement analysis applied in this project is explained. The chapter continues with the most important high-level functional and non-functional requirements recognized in this project. The main use case for the system architecture and design is elaborated.

## *6.1 Requirement Analysis*

Most of the results of this project are the basis for future research and application projects at TNO-ESI and TNO-ESI's customers. The artifacts of this project are going to be reused for different purposes; consequently all the requirements that were discussed with the stakeholders are part of the project backlog. However, because of the limited time budget and different priorities of the requirements, MoSCoW analysis (Brennan, 2009) was performed to prioritize the requirements in the project backlog. The MoSCoW analysis categorizes the requirements in one of the following groups:

• M - MUST: Describes a requirement that must be satisfied in the final solution for the solution to be considered a success.

• S - SHOULD: Represents a high-priority item that should be included in the solution if it is possible. This is often a critical requirement but one that can be satisfied in other ways if strictly necessary.

• C - COULD: Describes a requirement that is considered desirable but not necessary. This will be included if time and resources permit.

• W - WON'T: Represents a requirement that stakeholders have agreed will not be implemented in the current release, but may be considered for the future.

The requirements from the MUST group have the highest priority and the requirements in the COULD group have the lowest priority. The requirements from the WON'T group are not going to be implemented.

During the project, slight adjustments were made to the original requirements, mostly because of the candidate's and stakeholders' better understanding of the domain problem, and because of the impact of the initial results on the stakeholders.

## *6.2 Functional Requirements*

Based on the communication with the stakeholders, a list of high level functional requirements was created, which was translated into a list of user stories for the project backlog. The most important high level requirements are described in the following section. These requirements belong to the MUST category if it is not stated otherwise. The rest of the requirements are part of the project wiki page and project backlog available at the project management software tool used in this project ("12.4.Project Management Techniques".)

### 6.2.1. FR1 - Infrastructure for extracting customer profiles

The most important high level functional requirement is to design and implement a prototype of an infrastructure that supports extracting customer profiles. Every stakeholder has interests in this requirement, especially the ASML stakeholders because they are the first TNO-ESI clients that are using the tool. The infrastructure has to provide output in the form of processed data. The infrastructure should provide a solution for the process mining issues: mapping issue and missing information are elaborated in the section "4.2. Applying Process Mining."

### 6.2.2. FR2 - Input assumption violation check

The infrastructure for extracting customer profiles is designed and implemented on certain input assumptions. The TNO-ESI and ASML stakeholders at any time want to know whether the infrastructure behaves correctly when different input is used. The stakeholders want to be informed when the input assumptions are violated. Therefore, a mechanism that will check how well the infrastructure behaves on different input is required.

### 6.2.3. FR3 - Log conformance with a predefined model

The Themis project is the main stakeholder for this requirement. The requirement involves conformance checking of a log file with a predefined model. The requirement has to be elaborated with an example from the ASML domain.

### 6.2.4. FR4 - Evaluation of the process mining algorithms

The Themis project is the main stakeholder for this requirement too. This requirement involves evaluation of the process mining algorithms for protocol based communication. Initially, the process mining technique was developed for business process management, which is slightly different from protocol based machine process management. Therefore, an evaluation of the process mining algorithms based on the process mining knowledge, ASML domain data sets, and candidate's personal experience gained during the project was required. This requirement is covered in the section "4.3. Process Miner Evaluation."

### 6.2.5. FR5 - Procedure for process mining model generation

The project has to result in a procedure for generating process mining models. This procedure is based on the infrastructure that was developed as a part of the project and therefore every stakeholder has an interest in the requirement.

### 6.2.6. FR6 - Static and dynamic model of the participants

The stakeholder for this requirement is the Magenta project. This requirement involves a static and a dynamic model of the participants in a production environment. Based on these models the Magenta project can generate a state machine (behavioral model) for each participant.

### 6.2.7. FR7 – Improvement Opportunities

Another high level requirement that has to be delivered is a list of improvements of the used tools and the log files. At the same time it involves a list of improvement of the current logging at ASML and recommendations for designing more appropriate logging mechanisms for process mining. TNO-ESI is the main stakeholder of this requirement, because it shows the feasibility of the project and it gives new project opportunities.

### 6.2.8. FR8 - Participant classification based on their behavior

Several stakeholders from TNO-ESI and ASML are interested in this requirement. This requirement involves classification of the participants in a production environment based on their behavior. There are several solutions for this requirement, based on different classification algorithms and on different input variables that describe the participant behavior. This requirement is in the SHOULD category.

### 6.2.9. FR9 - Analyze resources with TRACE

In the observed production process, besides the different participants and the control messages, a set of items is also used. One way of tracing item usage in different activities is by using the Tracing Resource ACtivities by Esi (TRACE) (Trace Documentation, 2014) tool. This requirement involves generating a proper input for

the TRACE tool from the developed infrastructure. Themis and Magenta projects are the main stakeholders for this requirement, because it gives another insight (created/used items) in the production process. This requirement is in the SHOULD category.

## *6.3 Non-functional Requirements*

As it is stated in the chapter "2. Stakeholder Analysis," the main stakeholder for the non-functional requirements is TNO-ESI. Driven by the core idea of open innovation and delivering portable solutions, TNO-ESI expects that the final artifact can be reused for various clients (Figure 3). Therefore the main non-functional requirement is the portability of the proposed design, in the sense of how easy it is to reuse the system when the environment is changed. The meaning of portability in the context of this project is described in the following subsection.

### 6.3.1. NFR1 - Portability

Portability in (Bachmann, Len, & Robert, 2007) is a quality attribute of the software architecture that relates to the ease with which software that was built for one environment can be changed to run on a different environment. The different environment in the sense of this project is a different TNO-ESI customer, which means different logging infrastructures and data sources.
(van de Laar & Punter, Views on Evolvability of Embedded Systems, 2011) defines four categories of "How to respond to a change":

1) Alter nothing in the system (Generic Component)
2) Alter the parameters of the system (Parameterized Component)
3) Alter the system but not the architecture (Domain-specific Component)
4) Alter the system and its architecture

In order to measure the level of portability of the proposed system, each component and its variations are classified in one of the above categories. With the classification the stakeholder can understand how much the proposed system is portable and how easy it is to use for another client. In the context of this project, portability addresses the following concerns:

- What kind of change has to be made?
- Who makes the change, the domain expert or the developer?
- When is the change made?

If the proposed architecture has an isolated domain then the change can be made by a domain expert via domain knowledge binding. The same change can be made in the development phase or in a production environment. These concerns are the main driver for the system architecture analysis which is elaborated in the following section.

### 6.3.2. Secondary Non-functional Requirements

During the stakeholder meetings a few more non-functional requirements were recognized, but they were categorized as less important or not applicable for this project.

- NFR2 - Performance – In the context of this project, performance is related with the time demand in the data preprocessing steps. This requirement was partly met in the implemented infrastructure.
- NFR3 - Interoperability – Interoperability is about the degree to which two or more systems can usefully exchange meaningful information via interfaces in a particular context. In this project the infrastructure for extracting customer profiles should export models to different test frameworks. Due to the time budget limitations this requirement is not implemented.
- NFR4 - Scalability – In several stakeholder meetings, the scalability of the infrastructure in terms of number of observed processes, machines, and participants was discussed. This requirement was not considered because of the prototype nature of the project.

## 6.4 Main Use Case

In the context of this project the term use case is used in two different contexts. The first one is the customer use case. The “Customer Profiles” are extracting the actual usage of the system in a production environment. This is elaborated in the subsection “4.2.1. Identifying Process and Events.” The second context is the project use case. Figure 24 shows the project main use case, which includes Extracting the Production Process. Extracting the Production Process corresponds to the selection of a process. If we take a look to the functional requirements elaborated above, extract the production process corresponds to FR1 and FR5. The project use case is the main use case for the system design and architecture activities elaborated in the following chapters.

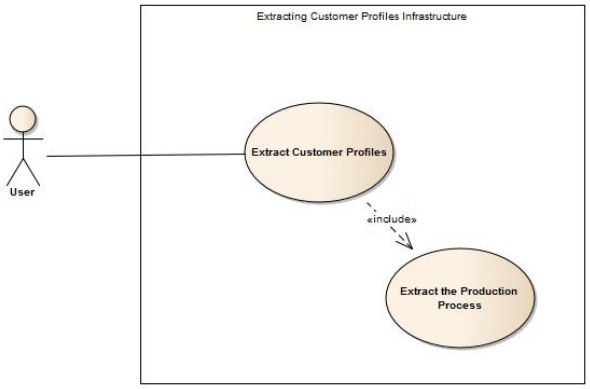


Figure 24 Project Main Use Case

■

# 7.System Architecture

The system architecture proposed and developed in this project is elaborated. The architectural reasoning and the design decisions that support the architecture are emphasized.

## *7.1 Introduction*

In order to develop the most suitable system architecture, the attribute-driven design (Bass, Clements, & Kazman, 2012) methodology was applied. The attribute-driven design is a methodology to create software architectures that takes into account the quality attributes of the software. The methodology consists of the following steps:

1. Choose an element of the system to design
2. Identify the architecturally significant requirements (ASR) for the chosen element
3. Generate a design solution for the chosen element
4. Inventory remaining requirements and select the elements for the next iteration
5. Repeat steps 1-4 for each element of the system design until all the ASRs have been satisfied

For the purpose of this project, the attribute-driven design was applied in several iterations in a top-down approach. In the first iteration, the "element" to begin with was the system itself from high level point of view. The rest of the iterations are related with the design of the system components and they are part of the following chapter.

## *7.2 Architecturally Significant Requirements*

The second step of attribute-driven design is identifying the architecturally significant requirements for the system under design. Architecturally significant requirements are requirements that drive the architectural design and they are divided in three groups:

- functional requirements
- quality requirements
- constraints

The functional and quality attributes are elaborated in the previous chapter. From a system architecture point of view, the most important functional requirement is providing an infrastructure for extracting customer profiles (FR1) and the most important quality attribute is the portability of the provided infrastructure (NFR1). From functional point of view, the system architecture should realize the main use case depicted in Figure 24. During the design phases and the stakeholders meetings, there was no constraint recognized. The project artifact will be used as a prototype and therefore no particular design constraints were considered.

## *7.3 Architectural Reasoning*

From a functional point of view, the system under design receives input, processes the input and produces an output. From a more generic point of view, the system must process or transform a stream of input data. The input is represented by different log data and the output is the actual customer profiles. Before going further with the architectural reasoning, an important step is to review the quality attributes and to select which tactics suit this project the best. The most important quality attribute is the portability of the proposed architecture. In (Bachmann, Len, & Robert, 2007) are elaborated the following tactics of how to design a portable system

- Reducing the cost of modifying a single responsibility

- Increasing cohesion
- Reducing coupling
- Deferred binding

Based on the modifiability tactics analysis, we need an architecture that will reduce the cost of modifying a single component, which means we need a component-based architecture. Further, we need to increase the component cohesion, which means that each component has to be responsible for a semantically different task. For some type of log data, we might need a different preprocessing technique, which means we need one or more intermediate steps. In order to make the architecture generic and suitable for different domains, we need to provide an engine for domain knowledge binding. As we emphasized in section "4.2. Applying Process Mining," the architecture has to enable integration of a client specific Mapping-rule Repository and a Complementary Information Repository. This constraint leads to the deferred binding tactics, especially the start-up time deferred binding. In that case the customer can inject the domain knowledge in form of parameters, a library, or a repository. The architectural pattern evaluation (Bachmann, Len, & Robert, 2007) in Table 6 shows several architectural patterns that increase cohesion and reduce coupling, but only the Pipes and Filters pattern provides deferred binding time, more precisely Start-up binding. Therefore, the architectural pattern that the most completely satisfies the functional (FR1, FR5) and quality requirements (NFR1), is the Pipe and Filter pattern.

Table 6 - Architectural Patterns and Modifiability Tactics

| Pattern | Increase Cohesion | | Reduce Coupling | | | | | Defer Binding Time | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Maintain Semantic Coherence | Abstract Common Services | Use Encapsulation | Use a Wrapper | Restrict Comm. Paths | Use an Intermediary | Raise the Abstraction Level | Use Runtime Registration | Use Start-Up Time Binding | Use Runtime Binding |
| Layers | X | X | X | | X | X | X | | | |
| ***Pipes and Filter*** | ***X*** | | ***X*** | | ***X*** | ***X*** | | | ***X*** | |
| Blackboard | X | X | | | X | X | X | X | | X |
| Broker | X | X | X | | X | X | X | X | | |
| Model-View-Controller | X | | X | | | X | | | | X |
| Presentation-Abstraction-Control | X | | X | | | X | X | | | |
| Microkernel | X | X | X | | X | X | | | | |
| Reflection | X | | X | | | | | | | |

## *7.4 Pipes and Filters*

The Pipes and Filters architectural pattern (Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 1996) provides a structure for systems that process a stream of data. It divides the task of a system into several sequential processing steps and each processing step is encapsulated in a filter component. This means that the pattern provides high cohesion by task decomposition (each filter component is responsible for a different subtask). At the same time, the pattern reduces coupling by encapsulating the filter components. The most important benefit of the Pipes and Filters pattern is the start-up time binding. This means that one can easily bind domain knowledge at the moment of the filter component invocation. In the interest of simplicity, from this point, by component in the Pipes and Filters pattern we mean the filter component.

## 7.5 System Architecture

The generic system architecture for extracting customer profiles by using process mining techniques, based on the Pipes and Filters architectural pattern is given in Figure 25.

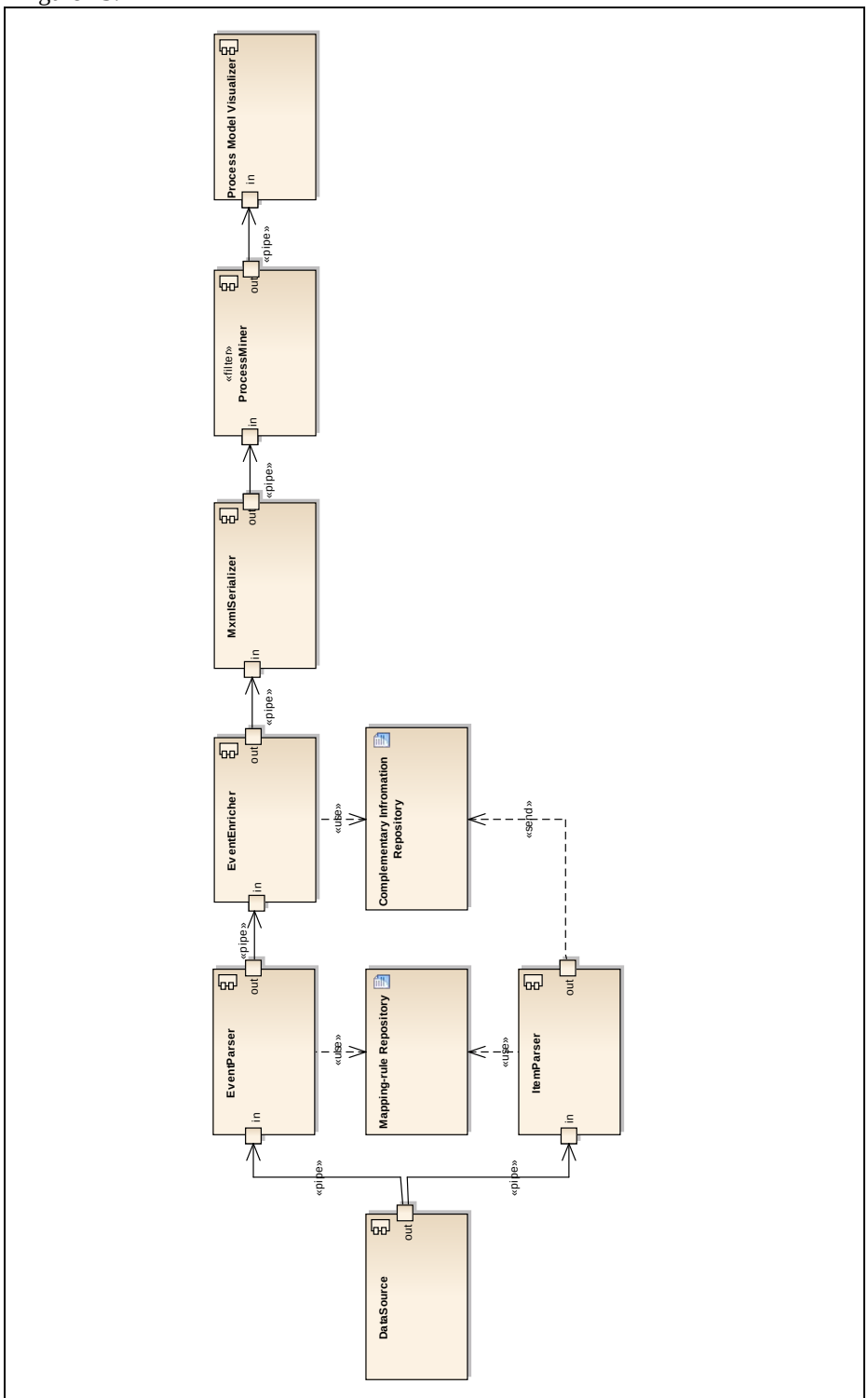


Figure 25 System Architecture for Extracting Customer Profiles

The pipeline starts with a Data Source component. Data Source is a special component that has only an output port. In general, a data source component can be anything, a file system, a relational data base, or a real time event listener. The role of the data source component is to feed the pipeline with input data. The data stream from the data sources goes to the Event Parser component. The Event Parser compo-

nent corresponds to the Extractor shown in Figure 15. This component solves the mapping challenge in process mining explained in subsection "4.2.2. Mapping Rule Repository." The Event Parser component uses customer specific mapping rules from the Mapping-rule Repository to extract event instances (*Event Instance*). The dependency relation between the Event Parser component and the Mapping-rule Repository explains the deferred binding functionality that the Pipes and Filters pattern offers. The mapping rules can be injected into the repository before the Event Component is executed.

In the section "4.2.3. Missing Information" we mentioned that the data sources can have additional information that can be used for event instance enrichment. The Item Parser component in Figure 25 has to extract complementary information and to store it in the Complementary Information Repository. In order to perform the extracting, the Item Parser component uses dedicated customer specific mapping rules from the Mapping-rule Repository.

If the extracted event instances are incomplete they are sent to the Event Enricher. The Event Enricher component corresponds to the Enricher shown in Figure 16. This component solves the missing information problem. The component enriches the incomplete event instance (*Event Instance'*) object with complementary information and sends Event Instances to the MXML Serializer component. In the case when the extracted event instances are compliant with the Standard Log Model, then the enrichment step is not necessary and the Item Parser and the Event Enricher component can be left out of the architecture.

The MXML Serializer component has the role of a transformer; it transforms the Event Instances (Figure 14) into the mining xml file, which is input for the Process Miner component.

The Process Miner filter is part of ProM 5.2 (Process Mining Tools, 2014). ProM 5.2is a generic open-source framework for implementing process mining tools in a standard environment. The Process Miner component extracts process models out of the input mining xml file by using process mining algorithms.

The pipeline in the architecture ends with a Process Model Visualizer. The Process Model Visualizer is a component that visualizes the generated process models. This component is also part of ProM 5.2.

The pipeline in the architecture does not necessarily have to end with a Process Model Visualizer. In general, the last component in the pipeline can transfer the data stream to another system. For instance, the process models extracted with the Process Miner component can be sent to a testing system. This is important, because it shows that the selected architectural pattern supports additional quality attributes such as extensibility and adaptability.

Applying customer profiles based on process mining at ASML showed that in the real production environment, more often there are multiple log data sources (Figure 18). That means that the architecture should provide different Data Source components for different log data sources. In that case, the generic system architecture for extracting customer profiles by using process mining techniques, based on the Pipes and Filters architectural pattern, looks like the one in Figure 26.

One can easily conclude that if there are multiple different log data sources as input to the system, then separate Data Source and Event Parser component are necessary for each type of log data source. Different Event Parser components have dedicated mapping rules in the Mapping-rule Repository.

If the event instances are incomplete there is an Event Enricher component for each Event Parser that parsers incomplete event instances. The Event Enricher components use the data stored in the Complementary Information Repository to enrich the incomplete event instances. Further on, the event instances are sent to the next component, Event Combiner. The Event Combiner component role is to collect all of the Event instances extracted from different data sources in a single Log object (Figure 14). In the literature (Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 1996), this variation of the Pipes and Filters pattern where a component has multiple inputs is called Join Pipes and Filters System. The proposed architectures shown in Figure 25 and Figure 26 depend on the data sources. If they contain additional information that can be used for event instance enrichment, then each data source has an Item Parser component (in Figure 26 is shown only one to illustrate the scenario).

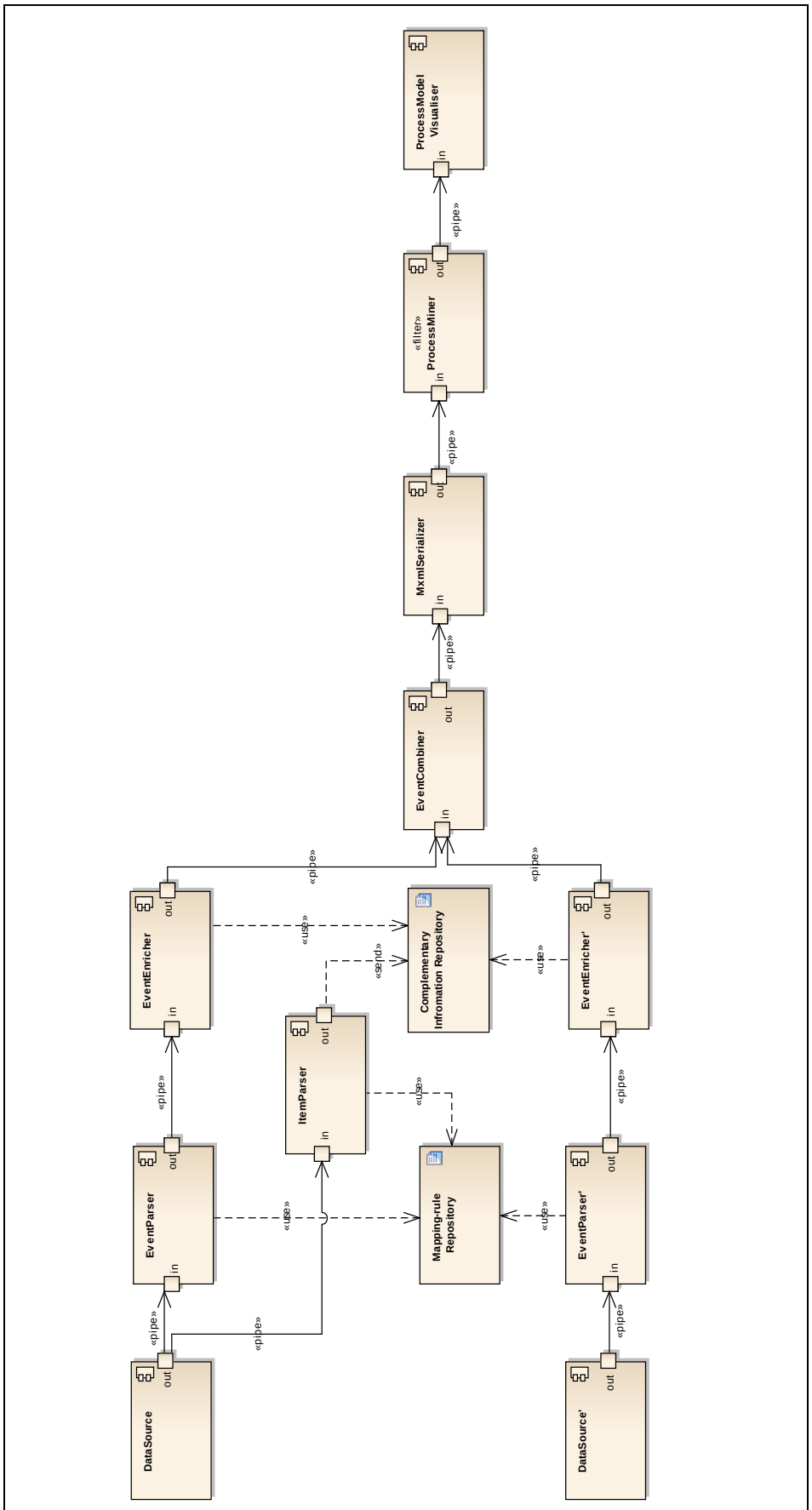


Figure 26 Multiple Data Source System Architecture

One of the requirements (FR9) is that the Trace tool should be used to analyze the results. For that purpose the generic system architecture for extracting customer profiles by tracing the used items during the main production process is shown in Figure 27.

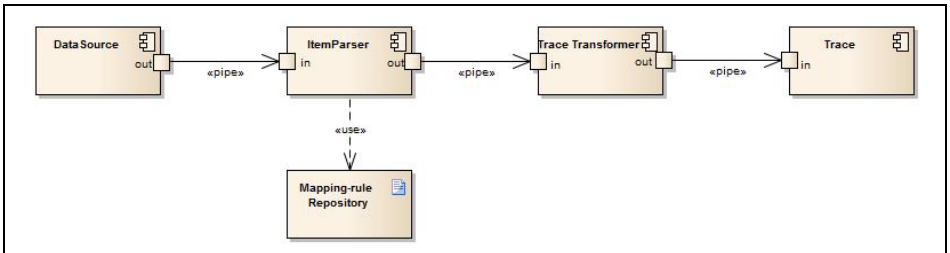


Figure 27 System Architecture Trace

Figure 25, Figure 26, and Figure 27 show UML component diagrams of three variations of the proposed architecture. None of the variations is final, because the final architecture depends on the number of data sources and the output of the system. Before we start designing an architecture for a certain TNO-ESI customer we have to answer the following questions:

- How many data sources does the client have?
- Do the data sources contain complete or incomplete event instances?
- Do the data sources contain complementary information?
- What is the output of the system (Process Models, Tracing Models, or log visualization?

With the answers on the questions above, a different implementation of the referenced architecture can be proposed for a different TNO-ESI customer.

Based on the recommendations for documenting Pipes and Filters architecture in (Clements, et al., 2010) the filters are represented with UML components and the pipes are UML associations with a «pipe» keyword. For the purpose of this project the pipe elements are left out of the design and implementation. In the proposed architecture we assume that the components exchange event instances in different formats such as text files or memory objects and there is not buffer size limit. In the next chapter, the more specific pipes are presented with information flow association. Further on, each component can have an input, or an output, or both ports. Each port can provide or require services to the other component. Whether it requires of provides a service depends on the implementation of the component.

## *7.6 System Component Overview*

One of the benefits of the Pipes and Filters architectural pattern is to maintain the semantic coherence of the components. The semantic coherence is achieved by splitting the responsibilities among the components. Subsequently, each component in the architecture has dedicated responsibilities. The responsibilities are grouped by the way they process the input data. In the pattern definition (Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 1996) there are three types of filters:

- Enricher
- Refiner
- Transformer

One component can be responsible for data enriching by computing and adding information, for data refining by concentrating or extracting information, or for data transforming by delivering data in some other representation. The component operates independently and does not depend on other processing steps. Figure 28 shows classification of the system components in the proposed architecture based on their responsibilities. The Data Source and the Data Sink components have a special responsibility. The Data Source only has an output port and is responsible to provide data input to the system. The Data Sink only has an input port and is responsible to present the final processed data. The Trace and the Process Model Visualizer are Data Sink components in the proposed architecture.

The filter component inherits from the Data Source and the Data Sink components and has an input and an output port. The rest of the components inherit from the filter component. Depending on their special responsibilities each component implements a different functional interface. The Event Enricher enriches incomplete event instances and therefore implements the Enricher interface.

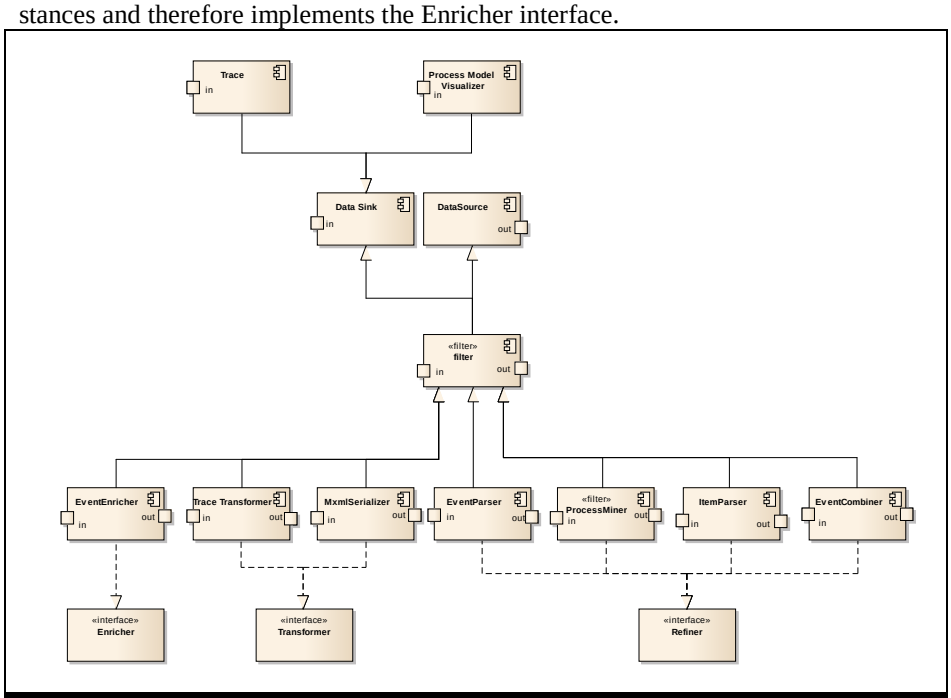


Figure 28 System Component Classification

The MXML Serializer and the Trace Transformer transform the event instances from one format to another and therefore they implement the Transformer interface. The Event Parser, Event Combiner, and the Process Miner refine event instances by concentrating (Event Combiner) or by extracting (Event Parser and Process Miner) information. The Item Parser extracts Item instances, and therefore implements the Refiner interface. In the following chapter the design decisions for each component are elaborated.

## *7.7 Component Portability Level*

In order to define how portable (see subsection NFR1 - Portability) are the components, three categories of portability were defined:

1) Generic Component – no alternation is needed
2) Parameterized Component – alternation of the component parameter(s) is needed
3) Domain-specific Component – complete alternation of the component is needed

Each category of component portability has a different color (Figure 29) and the same coloring is used for the diagrams in the following chapter. Along with the coloring, appropriate keywords are introduced to state the portability level of the component: «generic», «parameterized», and «domain-specific».


Figure 29 Component Portability Classification

■

# 8.System Design

The design of the components of the system architecture is elaborated. The goal, the variations, and the design decision of the most significant components are emphasized. In order to interpret the portability level of the components the amount of changes per component in case of porting to other TNO-ESI clients is discussed.

## *8.1 Introduction*

After the first iteration of the attribute-driven design methodology elaborated in the previous chapter, a few more iterations were executed. The goal of these iterations was to define the design of the most significant components in the proposed architecture. The architecturally significant requirements for each one of the most significant components were defined by the component functional requirements. For instance, the Data Source component has to provide input stream of data, the Event Parser has to extract an event instance from the input data, the Event Enricher has to enrich incomplete event instance into event instance, and the Event Combiner has to combine event instances from multiple sources. In this chapter the component design goes one level deeper in the details.

## *8.2 Data Source Design*

The main goal of the Data Source component is to provide input data for the system. According to (Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 1996), the data source represents the input to the system, and provides a sequence of data values of the same structure or type. Examples of such data sources are a file consisting of lines of text, or a sensor delivering a sequence of numbers.

Traditionally, the logging mechanisms store the logging data in text files. There are many reasons for that, but one of them (Common Log Format, 2014) is that the plain text format minimizes dependencies on other system processes such as: writing to a database or writing to a cloud storage system. The plain text format assists logging at all phases of computer operation including start-up and shut-down where such processes may be unavailable.

In the design of the Data Source we made the assumption that the input data is always stored in log files. Therefore, the main function of the Data Source component is to read the log files and to select a certain amount of data that represents a possible event instance. Storing the log data in plain text files gives freedom to the developers to define their own formats and styles. For that reason we assume that there are three types of log file structure:

- Plain Text Structure (including Coma-separated Value Structure)
- XML Structure
- Nested Structure

For the log files with plain text structure, we assume that each line of text represents data for single event instance. With this assumption the Data Source component has to read the log file line by line and send each line of text to the next component. In the Data Source for log files with plain text structure an explicit definition of the amount of data to select is not necessary. That makes the solution domain independent and generic. In the context of ASML, we witnessed log files with “Comma-separated values” (CSV) format. In the design of the Data Source component they are treated the same as the plain text files.

In case of XML log files, the data source has to read the XML log file and query for the nodes that represent possible relevant event instances. The XML Data Source component is generic and does not require domain specific implementation.

For the log files with nested structure, the design and the implementation depend on the domain specification. In general, log files with nested structure can be found in the industry and usually they define nested process instances with multiple event

instances. There is no standard specification of the nested structure (like number of nested statements, attributes, properties, etc.) and therefore the design of the data source component for structures like this is not trivial.
If we take a look at the proposed Pipes and Filter architecture, the data processing direction is from domain specific to generic. That means that the components on the left most side are more domain specific and the components on the right most side are more generic. This observation makes the Data Source component the most domain specific, which corresponds to the analysis elaborated above.
In order to illustrate the portability of the proposed Data Source components a component diagram is shown in Figure 30. The Text Data Source and the XML Data Source are generic components and can be reused for different TNO-ESI clients' without any change. The Nested Structure Data Source, because of the complexity of the structure is a domain-specific component.

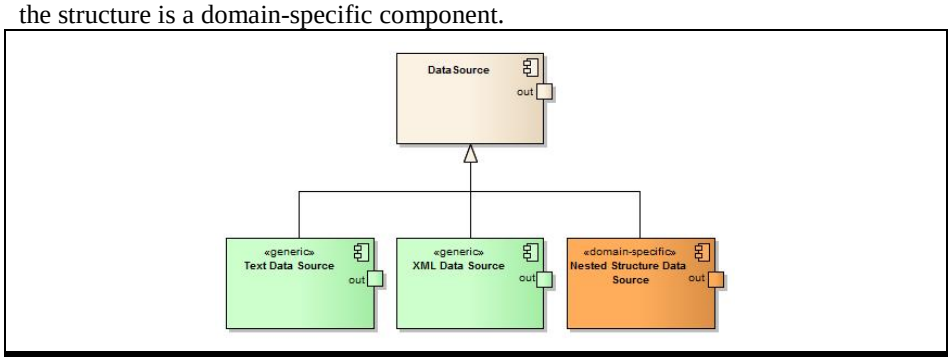


Figure 30 Data Source Variations

## *8.3 Event Parser Design*

In general, the Event Parser component has to extract an event instance from the input data. It has to read all the customer specific mapping rules from the mapping-rule repository and then to check whether the input data confirms to one of the rules. The design of the Event Parser component depends on the design of the corresponding Data Source component.

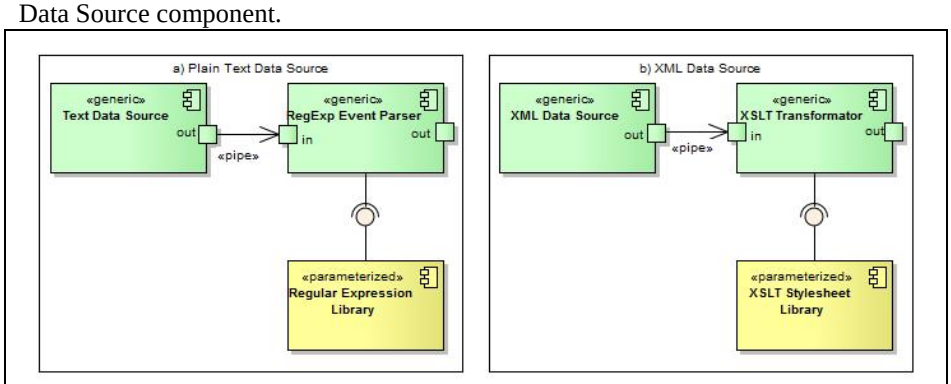


Figure 31 Event Extractor Variations

For Text Data Source the most common way to extract information from a text is by using regular expressions. For that reason, the customer specific mapping rules are actually an array of regular expressions loaded (Figure 31.a) in the Event Parser component. In this scenario the Event Parser component is named RegExp Event Parser. The RegExp Event Parser component checks whether the input data matches to one of the regular expressions. If there is a match, a new event instance is created and is sent to the next component in the architecture. One way to implement this use case is by using the strategy pattern. The strategy pattern (Gamma, Helm, Johnson, & Vlissides, 1994) is a software design pattern that enables an algorithm's behavior to be selected at runtime. It defines a family of algorithms, encapsulates each algorithm, and makes the algorithms interchangeable within that family. In the context of our problem (Figure 32), the pattern goal is to read the regular expressions at startup and to create encapsulated event parsers from each mapping rule. At the runtime, when the executeStrategy method from the EventParserContext is called, then each Event-

ParserImp is executed on the input string. If there is a match, a new event instance is created and is sent to the next component. If there are multiple matches or no match, statistical data (number of matches per regular expression) is collected for the FR2 - Input assumption violation check requirement (FR2).

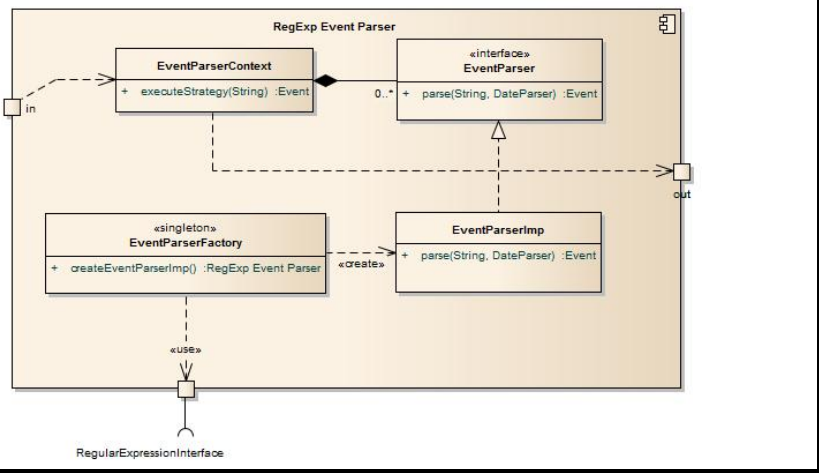


Figure 32 RegExp Event Parser Design

In the case of XML Data Source, the Event Parser component is actually a XSLT Transformer. It reads the (Figure 31.b) customer specific XSLT Stylesheet from the XSLT Stylesheet Library and transforms the input XML file in Standard Log Format.
If the extracted event instances from the RegExp Event Parser or from the XSLT Transformer do not conform to the Standard Log Model, then they have to be enriched with complementary information. For that purpose, the component model looks like the one shown in Figure 35. The RegExp Event Parser and the XSLT Transformer in Figure 35 send incomplete event instances to the Event Enricher. The design of the Event Enricher component is explained in the following section.
From this point of view, one can argue whether the Event Parser component has to realize the Refiner or the Transformer interface. In general, it depends on the tasks the component has. In the case of regular expression parsing, the component refines the input data and produces event instance as output. In the case of XML Data Source, the event parser transforms one form of presenting event instances to another.

## *8.4 Event Enricher Design*

In the chapter "4. Domain Analysis" we already emphasized that in applying process mining next to the mapping problem there is one more challenge. The missing information scenario and incomplete event instances is a real and common problem. Therefore, in the proposed architecture we introduced one more component, the Event Enricher. The Event Enricher has simple functionality, to enrich the incomplete event instances with the missing information. After the enrichment process, the event instances will be compliant to the Standard Log Model (Figure 14). The implementation of this functionality can be quite complex, and can consist of several sub functions, because it cannot be foreseen which information is missing for a different TNO-ESI clients. Therefore, the Event Enricher component in general is domain-specific and depends on the missing information.
An example of missing information is the join problem at ASML (Figure 18). One way to solve the join problem at ASML is by introducing a lookup table. In the design, the lookup table is customer specific and is bound to the Event Enricher component at start-up time. The design of the Event Enricher component in interaction with the rest of the components in the architecture is shown in Figure 33.

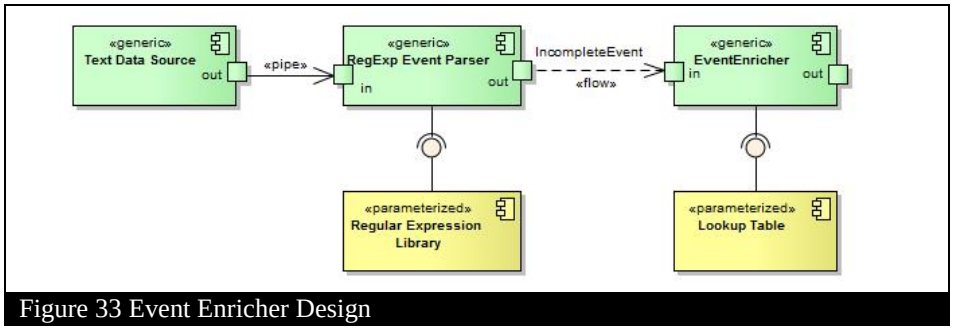


Figure 33 Event Enricher Design

The functionality of the Event Enricher can be easily solved with the Visitor Pattern (Gamma, Helm, Johnson, & Vlissides, 1994). The visitor design pattern is a way of separating an algorithm from an object structure on which it operates. The Visitor Pattern separates the Event Enricher from the incomplete event instances that have to be enriched. Figure 34 shows the internal design of the Event Enricher component. The ConcreteVisitor class uses information from the Lookup table to enrich the incomplete even instances. For this purpose, the incomplete event instance has to have an accept method that accepts the Visitor class.

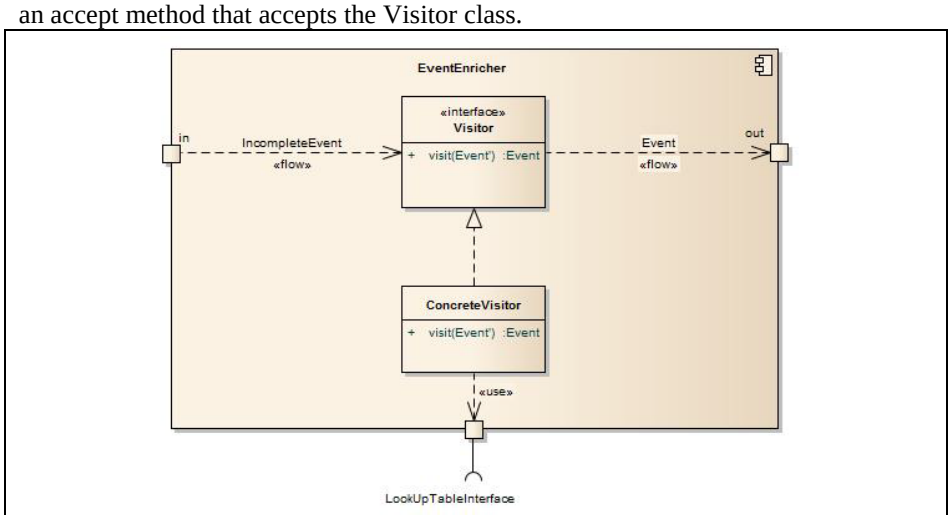


Figure 34 Event Enricher Internal Design

For the join problem illustrated in Figure 18, the ConcreteVisitor class is a generic class, because it knows exactly how to fix the join problem by using information from the lookup Table, which is domain specific. The join problem is only one of the possible missing information challenges. Therefore, the ConcreteVisitor class is not TNO-ESI client specific, but solves only a certain category of problems (Figure 18). For any different problem of missing information a problem specific implementation of the Visitor Interface and the missing information repository is required.
There might be a use case in the TNO-ESI clients' environment with multiple different data sources of incomplete event instances. If we assume that the incompleteness of the event instances is from same category as the join problem, then one possible design is shown in Figure 35.

## *8.5 Collecting Missing Information*

In the chapter "7. System Architecture," we mentioned that the additional information that we need to enrich the incomplete event instances can be found in the same log files that we use as an input in the system. Therefore, the architecture has to support additional information extraction from the Data Source components. For that purpose we introduced one more component, the Item Extractor component. The Item Extractor component extracts couples of ItemInstance and ProcessInstance. The couples are stored in the Lookup table. The interaction of the Item Extractor with the rest of the components is shown in Figure 36. The Event Enricher component uses

the couples stored in the Lookup table to enrich the incomplete event instances with the missing ***Process Instance ID***.

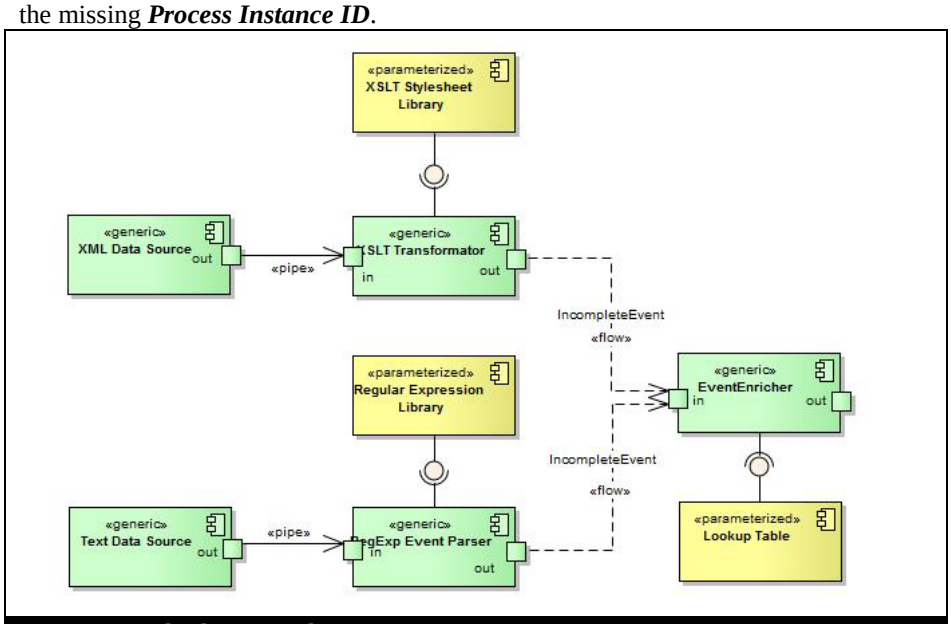


Figure 35 Multiple Incomplete Data Sources

The design of the Item Parser component is the same as the design of the Event Parser component (Figure 32). It uses dedicated regular expressions from the regular expression library and uses the Strategy Pattern to read the regular expressions at startup and to create encapsulated item parsers from each mapping rule.

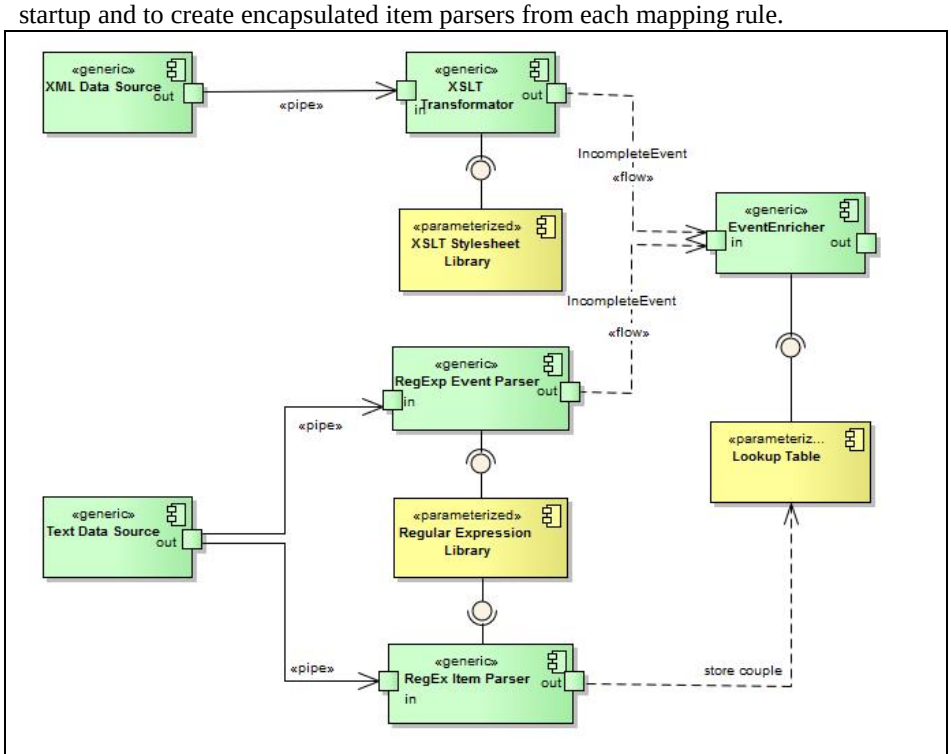


Figure 36 Collecting Missing Information Design

## *8.6 Event Combiner Design*

The Event Combiner component has domain-specific functionality, which makes the design of the component domain-specific. In general, the role of the Event Combiner component is to collect event instances linked with process instances from different data sources and to group them together. The main challenge is to recognize duplicate process and event instances. Treating duplicate process instances as different can

influence the process model of the observed process. Therefore, definitions of duplicate process instance and event instance are necessary. Usually, the definition of duplication is domain specific and includes heuristics. For instance, within ASML, there were multiple process instances with the same name. Therefore, we introduced a heuristic approach that assumes that there is only one active process instance with that name at one moment at the time.

## *8.7 Mxml Serializer Design*

The Mxml Serializer component is a transformer. It transforms an array of event instances into a mining xml file. For that purpose, it uses the OpenXES (pronounced as "open excess") library. OpenXES (Günther & Verbeek, 2014) is a reference implementation of the XES standard for storing and managing event log data. In general, the Mxml Serializer component converts event instances compliant with the Standard Log Model into entities of the OpenXES Model. In this case, there is a transformation from one generic model (the Standard Log Model) to another. Therefore, the design of the component is generic and can be reused for any TNO-ESI customer.

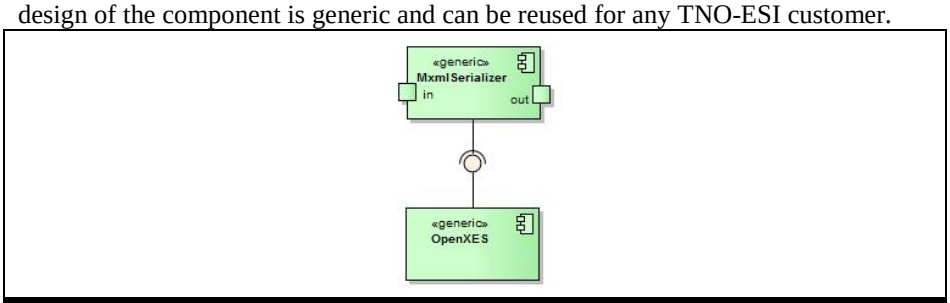


Figure 37 Mxml Serializer Design

## *8.8 Trace Transformer Design*

Similar to the Mxml Serializer component, the Trace Transformer component has to transform a data set from one model to another. The input for the Trace component consists of *resources* and *claims* that the resources were used in a certain time period. In the context of the "Customer Profiles," the resources are item instances used in the process instances (claims). In the case of the join problem, we already have a lookup table (Figure 38) with couples of item instance and process instance. Therefore, the Trace Transformer reads the couples from the lookup table and transforms them into resource and claim. From a more abstract point of view the architecture has to have a dedicated Data Source for item instances used in process instances. In that case, the architecture has dedicated pipeline for extraction and visualization of the item-process dependencies (Figure 27).

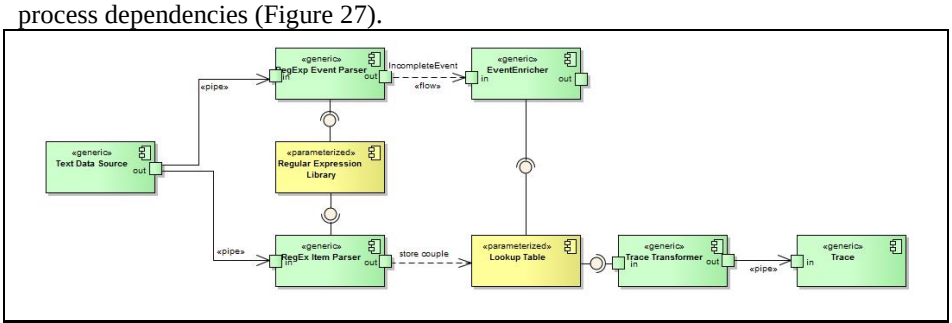


Figure 38 Trace Architecture with Lookup Table

## *8.9 Ready-made Components*

The Process Miner and the Process Model Visualizer components are part of the ProM 5.2 framework and are used as a ready-made parameterized component. The selection of the most appropriate process mining algorithm and the selection of the input parameters are elaborated in the chapter "4. Domain Analysis." The Trace component is a product developed by TNO-ESI. In this project is also used as a ready-made component.■

# 9.Implementation

The most significant details from the implementation of the project prototype are elaborated. The chapter also focuses on the implementation details and the implemented measurements to check the healthiness of the implemented prototype.

## *9.1 Introduction*

As a part of this project, a prototype was implemented. The prototype was based on the proposed architecture and it covered the use case from ASML. As we defined, the final architecture of the solution depends on the following aspects:

- How many data sources does the client have?
- Do they (data sources) contain complete or incomplete event instances?
- Do they (data sources) contain complementary information?
- What is the output of the system (Process Models, Tracing Models, or something else?

## *9.2 Prototype*

The prototype had to satisfy the functional requirements (FR1, FR2, FR5, and FR9) elaborated in the section "6. System Requirements." Therefore, based on the project main use case and the functional requirements the architecture of the prototype has three different data sources. Each of the data sources consists of plain text log files with different structure. Sequentially, three different data sources and three different RegEx Event Parsers were implemented. In the use case, only one of the data sources contains complete event instances and the other two data sources have incomplete event instances. For the data sources with incomplete event instances dedicated Event Enricher components were implemented. The multiple data sources impose an Event Combiner component that combines the event instances from the different data sources. The prototype had to generate customer profiles based on process mining and resource tracing. Therefore, appropriate Mxml Serializer and Trace Transformer components were implemented. The final architecture is shown in Figure 39. For the three RegEx Event Parser components a common RegEx Library was implemented. The library contains dedicated classes for each RegEx Event Parser component and each class has a set of methods that represent different regular expressions. In fact, there is a regular expression for each event (complete and incomplete) that can be found in the data source and that belongs to the observed production process.

## *9.3 Decisions*

During the implementation, several decisions from technology and component implementation aspect were made. They are elaborated in the next sections.

### 9.3.1. Technology

For the first part of the project, the data analysis phase, a set of Python scripts were developed to extract event instances from the log file. This phase contained several analytics use cases (see the chapter "12. Project Management".) The goal of the analytics use case was to explore information in the available data sets. The Python scripts were used to extract events with regular expressions.

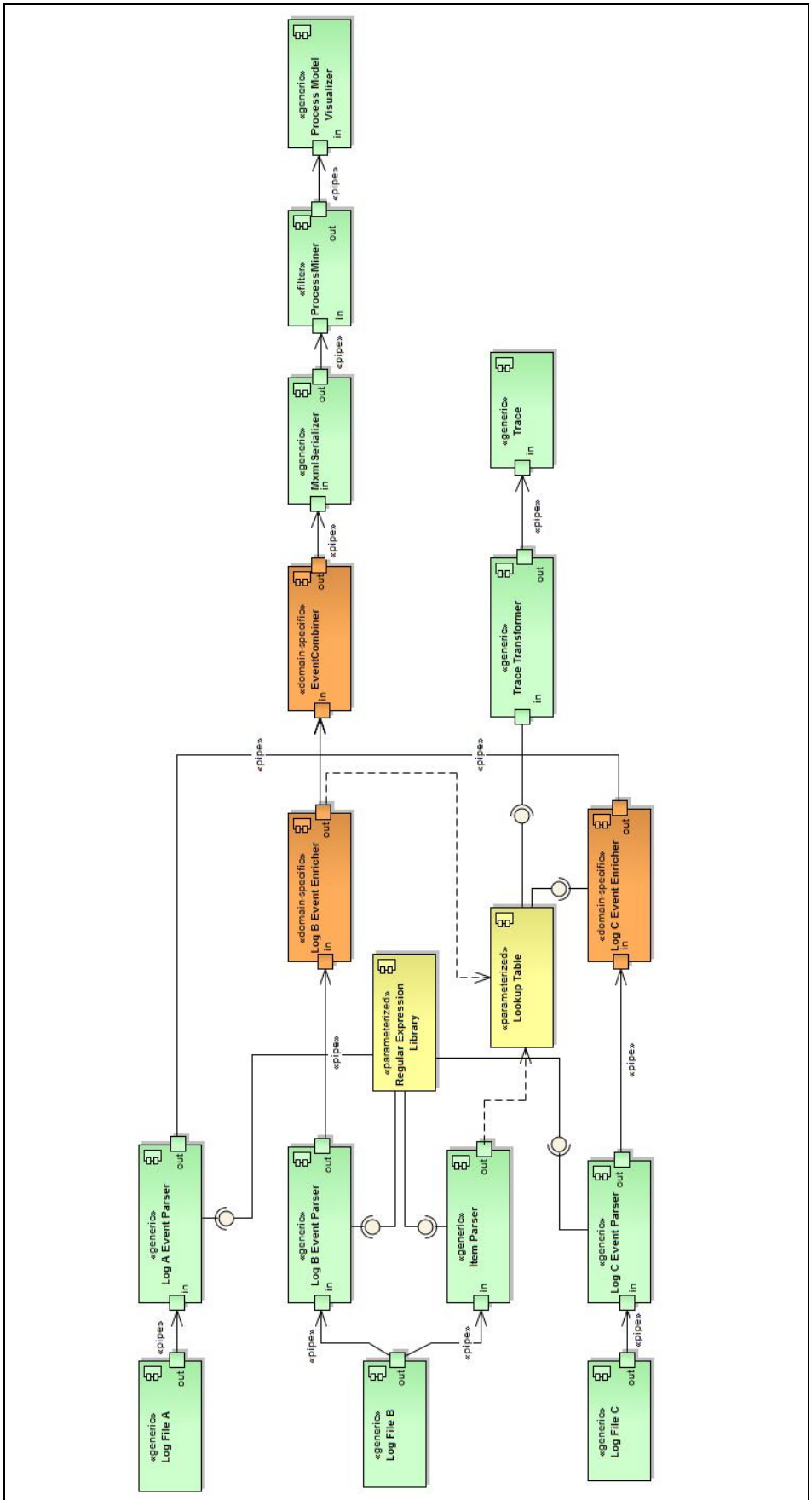


Figure 39 Implemented Architecture

The regular expressions from this phase were used for the Regular Expression Library implemented as a part of the prototype. The prototype is implemented in the Java programing language with Java SE Development Kit 7. There was no constraint from the stakeholders about the programing language selection. Based on candidate's personal preference the Java development environment was selected. Eclipse Kepler Service Release 2 was used as integrated development environment. This version is selected because it supports the Trace tool (Trace Documentation, 2014).

## *9.4 Prototype Healthiness*

In order to meet the requirement FR2, a mechanism to collect and to display statistical data was implemented. It is important for the stakeholders to get quick feedback about the prototype healthiness when there is a different input data set. Therefore, during the execution of the prototype two values are collected: the number of event instances and number of process. This information gives a very quick view of the healthiness of the prototype. The stakeholders have quick answer to the question: "Are we extracting the expected number of event instances and process instances from the log files?" The collected statistical data is presented with JFreeChart Library v.10.017, which is a Java chart library. One example output is depicted in Figure 40. The horizontal axis represents the number of files and the vertical axis presents the number of event and process instances respectively. The lower (blue) line shows the number of total process instances. The upper (red) line shows the number of event instances.

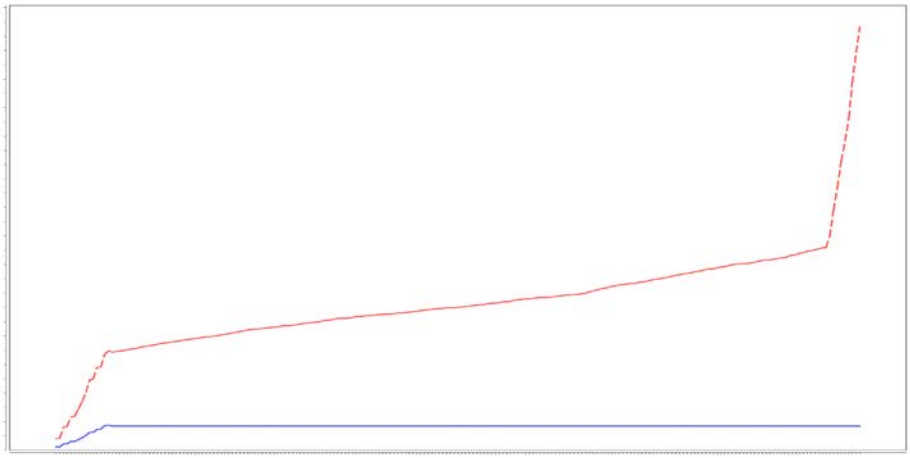

Figure 40 Prototype Healthiness

One can easy conclude that there are different slopes in the number of event and process instances extracted from the log files. This can be explained with the fact that as an input to the prototype, three different log files are used (Figure 18.) Each file has its own number of event and process instances. The blue line, in the beginning of the chart, grows till certain value and after that is constant. That means that only in the first type of log files there are process instances. The red line grows all the time, but with different slopes. This means that each log file type has different number of event instances. The result can be used to get quick overview whether the prototype extracts process and event instances or there is a problem.

### 9.4.1. Component Implementation

The components in the prototype were implemented according to the component design elaborated in the chapter "8. System Design." Based on the amount of the data that had to be processed with the prototype it was decided that the components can be implemented in a push pipeline scenario. The push means that after one component is ready then it sends the output to the next component in the pipeline. Another implementation decision is that the components are part of one process; they are not separate execution processes in the operating system. Again, this decision is justified only if the amount of the data that has to be processed is in the same scale as the one that had to be processed with the prototype.

## *9.5 Deployment*

Considering the fact that the result of the software development is a prototype, special deployment analysis was not conducted. The prototype is implemented with the Java development environment and in general can be deployed in any production environment that hosts Java Virtual Machine. Next to the prototype, the process mining tool and the Trace tool are also implemented in Java and therefore they can be executed in the same development or deployment environment as the prototype.■

# 10. Verification & Validation

In this chapter the verification and validation techniques applied in the project are elaborated. Their goal is to make sure that certain rules are followed at the time of development of the prototype and also to make sure that the prototype fulfills the required specifications.

## *10.1 Validation*

The validation techniques should answer on the question: are we building the right thing? In a research project, the first thing that has to be done is to perform experiments and to communicate the research results with the stakeholders.
This project was executed in two phases. The first phase was research oriented and the second phase was software development oriented. The research results were communicated with the stakeholders. During the stakeholder meetings the results were directed towards reaching the stakeholder business goal: definition of customer profiles that satisfies the requirements. After several iterations the stakeholders agreed on the definition of the customer profiles based on process mining models and different visualization techniques. In the context of this project a customer profile is represented with Heuristics Net Models (Weijters, van der Aalst, & Alves de Medeiros, 2006), which are abstraction of the actual process. To get better understanding of the results, dotted and trace charts were introduced. With the experiment results from the data analysis phase a validation of the project output was performed. All the stakeholders agreed that the customer profiles defined with the charts and models satisfies their expectations.
In order to validate that the defined approach delivers the same results for different TNO-ESI clients, the approach had to be tested on different use cases. Different use case is use cases from different ASML's clients and from different customers from the TNO-ESI's clients. In this project, the validation was only performed on different use cases from different ASML's clients. Examples of the defined customer profiles extracted from different data sets are elaborated in the following subsections.

### 10.1.1. Dotted Charts

Dotted charts show a spread of event instances of an event log over time. The basic idea of the dotted chart is to plot dots according to the time. There is no abstraction or generalization in the dotted charts. They group event instances by the process instances.
In general, the dotted charts are used to get a quick overview of the processed log files. The dotted charts show the main process instance flow and can help in detecting deviations from the main production process. Figure 41 shows an example of a dotted chart generated as a part of the experiments executed with a data set from one of the ASML's clients. The horizontal axis presents the actual time and the vertical axis presents different process instances. The process instances are ordered by the event instance time of their first event instance. Each color presents a different event. Each colored dot presents different event instance. The dots that lie on the same horizontal line present event instances that belong to the same process instance. All the event instances follow the production process (which is presented with the diagonal line in the Figure 41. Therefore, it is easy to detect outliners (event instances that are not part of the expected behavior) or breaks in the production process (like the one in the middle of Figure 41.

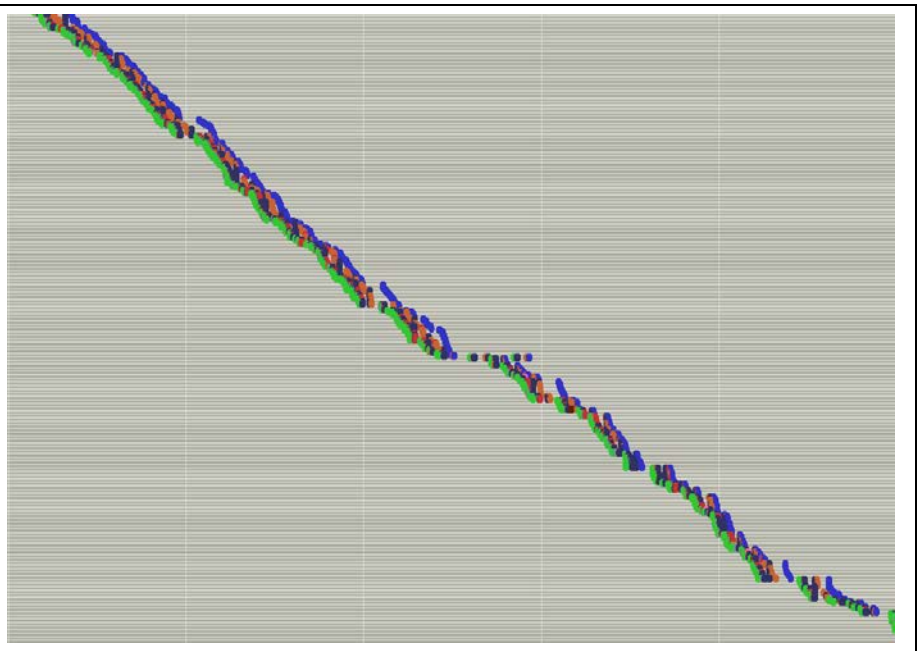

Figure 41 Dotted Chart - Production Process

Figure 42 shows zoomed in view of the Figure 41. Looking at the distribution of the colors it can be easy concluded that the event instances follow a certain pattern. The pattern they follow is extracted with the Process Miner Component. Another conclusion is that there are event instances that occur periodically for every process instance. That shows that in a production environment there are participants with different behavior, which is the first step in participant classification based on their behavior. Some of them are process triggered, and some of them are triggered by an external trigger but still belong to the same process instance.

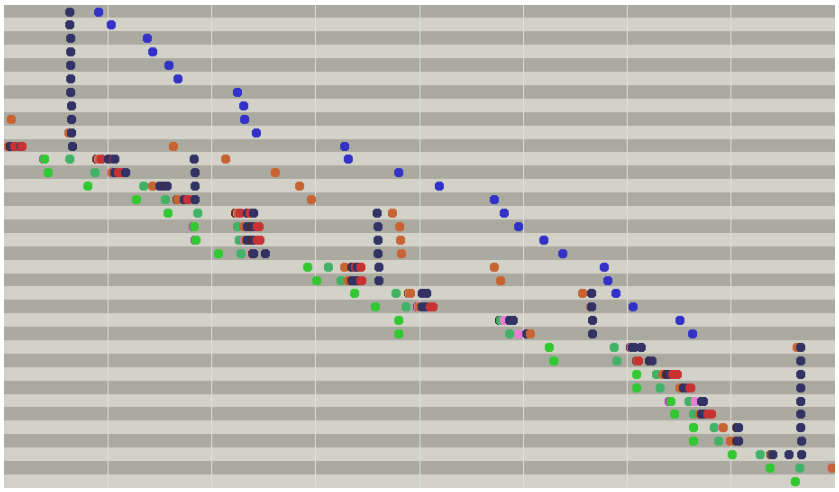

Figure 42 Dotted Chart - Zoomed In View

Figure 43 shows another view of the same data depicted in Figure 42. Here all the event instances are shown with relative time. It is easy to conclude that each process instance starts with the same event (same color). After that there is some regular distribution of the events, which means that each process instance follows some pattern. For the stakeholders this chart is important because different patterns in the time intervals can be observed. The pattern of event order execution can be extracted with the process mining algorithm, which is elaborated in the following section.

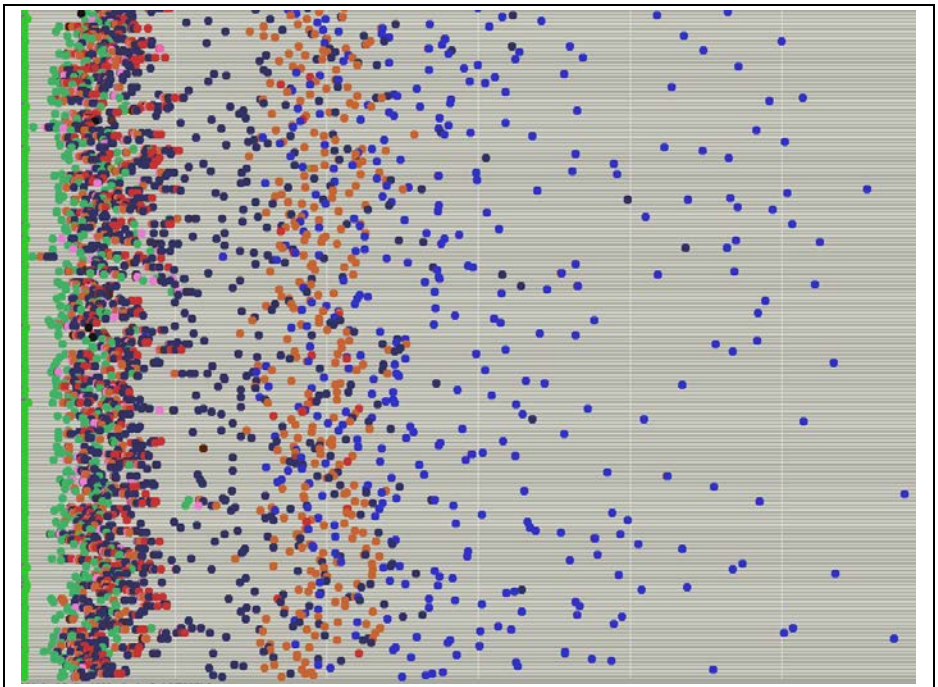

Figure 43 Dotted Chart - Relative Time View

### 10.1.2. Heuristics Net Models

Based on the problem depicted in Figure 18, a set of Heuristic Net Models were generated. A description of the Heuristics Net Models is given in the section "4.3. Process Miner Evaluation." In the Log File Source A, in Figure 18, four complete events that belong to the main production process have been recognized. These events are part of the Extended Log Model (Figure 20), and they are tagged as Event I, Event II, Event III, and Event IV. The model generated by using only the Log File Source A, is shown in Figure 44. One can easy conclude that the Process Model is straightforward from Event I to Event II, then to Event III, and then to Event IV.

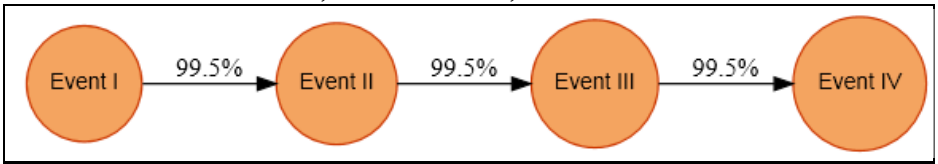


Figure 44 Log File Source A - Process Model

Figure 45 illustrates the Process Model generated by using only the Log File Source B from Figure 18. In the Log File Source B, five complete events that belong to the main production process have been recognized. These events are tagged as Event 1, Event 2, Event 3, Event 4, and Event 5. One can conclude that the model depicted in Figure 45 is very similar to the one depicted in Figure 44. The model presents the same Process as in Figure 44, but with different events.

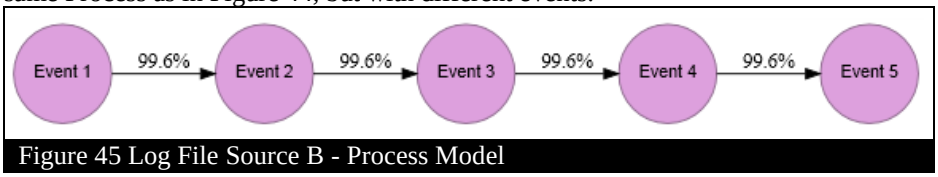


Figure 45 Log File Source B - Process Model

After communicating with the stakeholders, it was concluded the models show expected behavior. The stakeholders expect that the events are executed in the same order as the one in the process model. The next step was to combine the Log File Source A and Log File Source B, and to generate a process model that will describe the event execution. The result is depicted in Figure 46. The process model in Figure 46 shows the order of the execution of the events from the two Log File Sources

A&B. The high value of the dependency measure among the events tells that the process is executed straightforward and without any significant deviations.

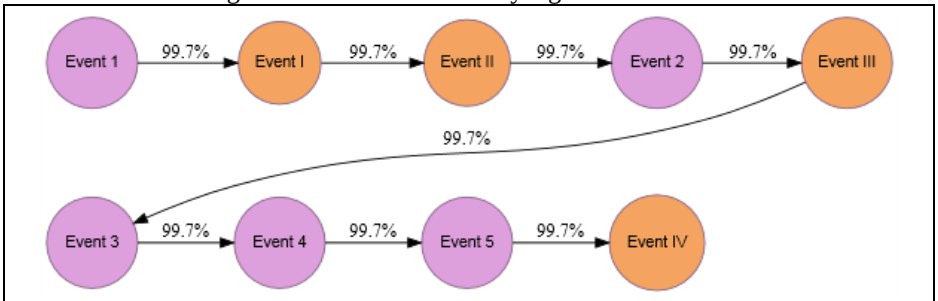


Figure 46 Log File Source A&B - Process Model

For the stakeholders, this model has more value than the previous because of several reasons. First, the model shows multiple participants, the different event color means different log source file, but in this case it also means different participants. Second, the model shows expected event execution order. The stakeholders expect straightforward execution of the events in some order.

The models elaborated above mostly show expected behavior. During the meetings with the domain experts it was concluded the presented models do not bring new business value. Therefore, in the analysis it was included the Log File Source C. It was expected that there are multiple events that belong to the same process. The data analysis showed that there are multiple events executed by different participants, but they are incomplete events. This problem is depicted in Figure 18. The final data set contains event instances from the three log file sources. In the data set there are different event instances from different events and from different participants. Figure 47 depicts a Heuristics Net model of the events from the three log file sources and three different participants. In this case still different color means different log file source and different participant. From the Log File Source C was recognized only one event, Event A. The most important thing about this model is the deviation from the expected event order execution. The deviation is depicted with the fork branch from Event IV to Event V and Event A. For the purpose of this project, the fork events in the Heuristics Net Models are treated as logical XOR functions, which mean that after the Event IV it can execute Event A or Event V, but not both. The deviation tells that not each process instance has the same execution path. One can notice that there are two numbers on the arcs from Event IV to Event V and Event A, respectively. The top number, as we already mentioned above, describes the dependency measure. The bottom number shows in how many process instances Event 5 follows Event 4, and in how many process instances the Event A follows Event 4. In other words, 75.7% of the process instances have the following execution path, which exclude Event A:

Event 1 > Event I > Event II > Event 2 > Event III > Event 3 > Event 4 > Event 5 > Event IV

and 24.3% means that 24.3% of the process instances have the following execution path, which include Event A:

Event 1 > Event I > Event II > Event 2 > Event III > Event 3 > Event 4 > Event 5 > Event A > Event IV

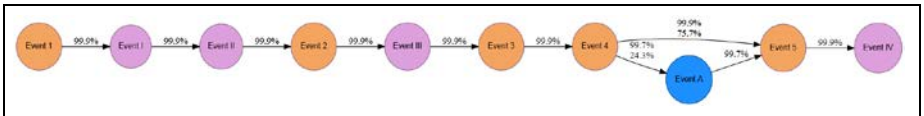

Figure 47 Final Data Set Heuristics Net (Sources A,B,&C)

For Themis stakeholders the process model actually tells that with two test cases (based on the two execution scenarios) they can simulate the actual usage of the system with the selected participants. One of the test cases covers 75.7% percentage of the process instances (it can be interpreted as 75.7% of the time) and the other test case covers the 24.3% of the process instances.) For Magenta stakeholders the generated process models are important because they satisfy the functional requirement FR6 - Static and dynamic model of the participants. From the process models a Uni-

fied Modeling Language (UML) Sequence Diagram can be extracted and from the Sequence Diagram can be extracted a UML State Machine Diagram for each participant in the model.
The Heuristics Net Models in practice are far more complex. For Instance, Figure 48 depicts a Heuristics Net Model of the same process as in the previous models, but different events (Event B, Event C, Event D, and Event E) from the Log File Source C.

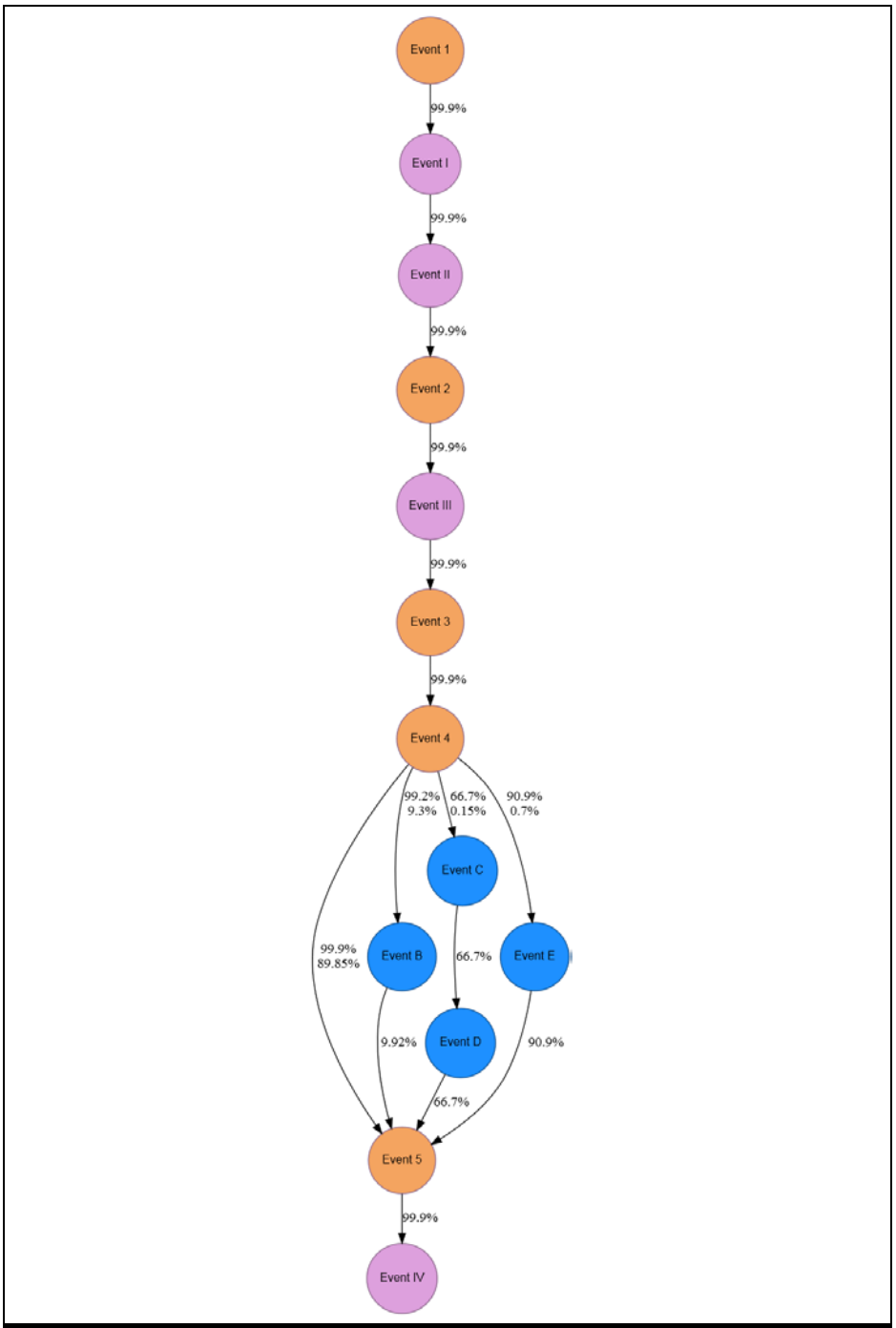


Figure 48 Complex Heuristic Net Model (Sources A,B,&C)

Before we analyze the results from Figure 48, we want to assume that between the Event 4 and the Event 5, any possible permutation of Event B, Event C, Event D, and Event E can happen. If a repetition of the events is allowed then we have infinite number of possible scenarios. If there is no repetition allowed then we have 65 possible scenarios. Similar to the previous observations, from the model depicted in Figure 48, we can create four execution scenarios. The idea is that, when we have the only a

design model, then the number of test scenarios can be 65 and more. By using the customer profiles we can lower that number to four actual test scenarios. Moreover, these four test scenarios have different value for the stakeholders. Table 7 shows the test scenarios with the appropriate percentage of occurrence of the process instances. With the results we can explain the idea of selecting the most appropriate test cases, or business wise test case selection. The test cases with high coverage are more important for the stakeholders than the one with lower coverage. Based on the results, we do not have to create a test case for each possible event sequence order. With only 4 test cases we can reach 100% path coverage. Furthermore, if we select the first and the second test case, then we cover 99.15% of the cases in the log file, which for different purposes can be a sufficient path coverage.

Table 7 - Event Execution Paths

| Path | Event Execution Order | Coverage |
|---|---|---|
| I | *Event 1 > Event I > Event II > Event 2 > Event III > Event 3 > Event 4 > Event 5 > Event IV* | 89.85% |
| II | *Event 1 > Event I > Event II > Event 2 > Event III > Event 3 > Event 4 > Event 5 > Event B > Event IV* | 9.3% |
| III | *Event 1 > Event I > Event II > Event 2 > Event III > Event 3 > Event 4 > Event 5 > Event C > Event D > Event IV* | 0.7% |
| IV | *Event 1 > Event I > Event II > Event 2 > Event III > Event 3 > Event 4 > Event 5 > Event E > Event IV* | 0.15% |

### 10.1.3. Trace Charts

In the subsection “4.3.2. Missing Information” we already mentioned that in the log files there is additional information that fortunately can be used to enrich the incomplete event instances into event instances that are compliant to the Standard Log Model. In this project we use the item instances used or created in the process instances to connect the incomplete event instances to the process instances.

The Heuristics Net models described above show the process execution from control point of view. They do not tell anything about the resources used in the process execution. If the resources present different data that is used in the production process, then with the current data stored in the Lookup table, an overview of the data flow can be created. In our infrastructure, in the Lookup table there are couples of ItemInstance and ProcessInstance. These couples can tell the information which item was used or created as a resource in which process instance. One way to visualize this is by using the Trace Tool. The integration of the Trace Tool in the proposed architecture is elaborated in the subsection “8.8. Trace Transformer Design.” Figure 49 depicts one output of the Trace Tool.

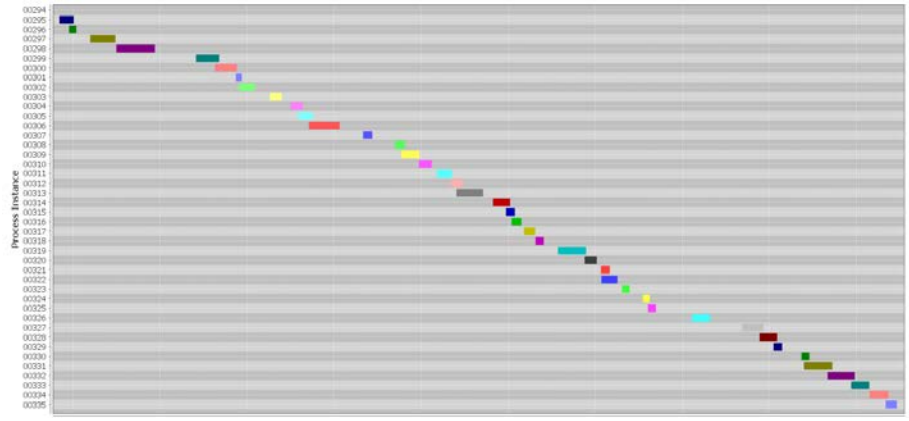


Figure 49 Item Usage in the Production Process

The output is very similar to the one depicted in Figure 41. It shows the same production process, but from different perspective (ItemInstance usage/production). The horizontal axis represents the relative time (first process instance) and the vertical

axis presents the process instances. Each colored rectangle presents different Item Instance. The length of the rectangles shows how long a certain Item Instances was used/ produced. In a fact, the length of the rectangle is the time in which the process instance related with that Item Instance was executed. The purpose of this output is to give quick overview to the stakeholder about the production process but from different perspective. For instance, it might be a scenario in which same Process Instance is using/producing multiple Item Instances, or same Item Instance(s) is used/ produced by multiple Process Instance(s).

## *10.2 Verification*

The verification techniques should help in answering the questing: Are we building the thing right? In general, verification in a project with a research part is ambitious. It is ambitious because no one can predict the research results and because the results can show that our approach is not feasible for the purpose.
In this project, the results from the data analysis phase were used as a basis for the verification. In fact, it means that the data analysis and the software development phase should have the same results. The only difference is that the research results are obtained manually and the results in the software development phase were obtained by using the developed infrastructure.

### 10.2.1. System Testing

By using the approach elaborated above system testing was performed. The completely integrated system was tested to verify that it meets its requirements. In a sense, it tests that:

- the data source components are obtaining data from the data sources
- the Event Extractor components are extracting the expected number and types of events
- the Event Enricher components are converting incomplete event instances into complete event instances
- the Event Combiner components are combining event instances
- the correct Heuristics Net Models are generated from the Process Miner Component
- the correct visualization is generated from the Process Mining Visualizer

By getting the final result (visualized Heuristics Net Models) and comparing it with the same one obtained manually in the data analysis phase, one can verify that the system is doing the right thing.

### 10.2.2. Unit Testing

As a part of the development the mapping rule repository was developed. The Mapping-rule Repository was filled with regular expression that extract event instance attribute from a certain text. The number of events that are part of the generated process models was growing during the project and a certain level of assurance that the developed repository supports all the data variations of the event instances among the data sources was necessary. Therefore, unit testing was performed and there were written unit tests for each regular expression in the Mapping-rule Repository.■

# 11. Conclusions

The results and the lessons learned are elaborated in this chapter. The results are grouped around the developed prototype, the designed architecture that supports extracting customer profiles, and the comprehensive domain analysis. The problems and issues that were faced in the project are transformed into suggestions for improved extracting of customer profiles. The chapter ends with the most important lessons learnt during the project.

## *11.1 Results*

In the previous chapter we have shown that the process models comply with the stakeholder needs. ASML stakeholders were satisfied with the results because they can model the typical and atypical behavior of different participants in the production environment including the participants manufactured by ASML and by other manufacturers. Based on the interaction between the third party equipment and the ASML equipment, we can model the behavior of the third party equipment, which is an essential step in understanding the production process.

### 11.1.1. Prototype

There are multiple project results and they are all relevant from different perspectives. One of the most important results is that the project provided a prototype that shows the feasibility of using process mining to obtain customer profiles. It is validated by the stakeholders that the Dotted chart, Heuristics Net Models, and the Trace Charts give sufficient insight in the customer profiles: the actual context and usage by the different customers of the equipment. For TNO-ESI it is important because it can be used to support model-based testing and architectural reasoning activities for different clients. ASML can use the prototype as a standalone tool to support the ongoing customer profile efforts.

### 11.1.2. Portable Software Architecture

Based on the positive reception of the prototype, an appropriate software architecture was designed to support extracting customer profiles for different TNO-ESI clients. The architecture provides a solution for the problems in applying process mining such as: mapping domain specific model to Standard Log Model and missing information in the log data. The components in the architecture were designed to require minimal changes in the case of switching to another TNO-ESI customer. The components are categorized based on their level of portability. Based on this categorization, TNO-ESI can calculate the amount of change effort that has to be done to port the architecture to different TNO-ESI clients. The system architecture and design is elaborated in the chapters: “7. System Architecture” and “8. System Design.”

### 11.1.3. Comprehensive Domain Analysis

From the data analysis part, a comprehensive domain analysis was derived. The main issues in applying process mining were emphasized. In order to be able to generate customer profiles based on process mining techniques, data requirements were defined. For that purpose, the log data source has to have the following information for each event instance: Event Instance Unique ID, Event Instance Originator, Event Instance Timestamp, Event Instance Class, and Process Instance Unique ID. These requirements are a basis for the Standard (Figure 14) and the Extended Log Model (Figure 20).

As a part of the domain analysis, an evaluation of the process miners for the purpose of customer profiles (FR4) was done. That helps in selecting the most appropriate process mining algorithm for different business goals. For instance, in one case the stakeholders would like to get the most common process structure, and in other case the stakeholders would like to see more exceptional or atypical behavior. The process miner evaluation is elaborated in the section "4.3. Process Miner Evaluation." The conclusion is that the heuristics and genetic miner are the miners that can deal with the noise and incompleteness of the log files and can generate models with different levels of abstraction of the real process. The heuristics miner requires process mining domain knowledge. At the same time the genetic miner requires better understanding of the process that has to be extracted but no extended process mining domain knowledge is necessary. The selection of the process miner depends on the stakeholder requirements, stakeholder process mining domain knowledge, and stakeholder understanding of the process that has to be extracted.

## *11.2 Suggestions*

Based on the needs and challenged that arose during the architecture design a set of suggestions for the logging infrastructures and for the process mining domain was created. This is one of the main requirements - FR7. The set of suggestions can help in more efficient process mining and can be applied to any TNO-ESI customer.

### 11.2.1. Logging Infrastructure Suggestions

The most important suggestions for the logging infrastructure are:

- Conscious manufacturer decision on log file syntax and semantics
- Global unique identifiers
- Use case aware event logging
- More precise time stamp
- Consistent time zone usage

The general suggestions start with logging infrastructure redesign. Within ASML we witnessed redundant data with missing information. The same data is logged on multiple locations, but still without relevant information for process mining. One of the suggestions is that the relevant minimal information based on the Standard Log Model should be logged. This leads to a more generic suggestion: conscious manufacturer decision on the log file content. In general, we do not want the customer that uses the equipment to decide what data will be logged. That puts in question the feasibility of the customer profiles. Without data, it is not possible to create customer profiles. Therefore, the suggestion is that the manufacturer, for instance ASML, should have predefined logging settings that cannot be overwritten by their customers.

Another important suggestion for the logging infrastructure is global unique identifiers across the entire logging domain and not on log file level. Having multiple log files is not a problem if there are unique identifiers for the entities they log. This requires some kind of central agreement, for example a component responsible for system wide unique identifiers or a standard naming schema.

In order to avoid the join problem there are two suggestions. The first one is using use case aware event logging. The log events should be annotated with a process instance that is part of a customer use case. In a complex system of systems, logging process instances on operating system level will not help in creating customer profiles. System level logging is good for diagnostics, but for customer profiles it must be connected with a customer use case. Use case aware event logging should be used even in a case when the logging infrastructure is spread across multiple log files.

The more concrete suggestions are based on the problems that were faced in the data sets. Mostly these problems are time stamp related. One recommendation is using standardized time stamp for the events. For instance, 01.05.2014 and 05/01/2014 can represent same date but also different date. Knowing the fact the end users of the ASML products are located all around the world, including the time zone in the time stamp has to be consistent. The last suggestion is for the time stamp accuracy. Most logging infrastructures are part of legacy systems and the time stamp is based on accuracy of one second. That would not be a problem, if we apply process mining for a process executed by humans. In the machine world where the high-tech equipment

executes thousands of operations per second, a higher precision in the time stamp is a necessity.

### 11.2.2. Process Mining Suggestions

Next to the logging infrastructure suggestions there are suggestions for the process mining algorithms and tools. First of all, the tool (Process Mining Tools, 2014) used to generate process model is an academic tool and is not yet mature for industrial use. For instance, importing preprocessed log file with 100k event instances resulted in memory issues with the tool. Performing complex algorithmic operations resulted in freezing tool's user interface. Therefore, many preprocessing steps were executed before running the process mining tool. These steps implicitly are part of the proposed system architecture (Figure 25.) The suggestion is that for industrial use of the process mining tool, especially in the era of big data, serious optimizations are necessary.

Another problem that was faced is the order of the event instances with same time stamp. When there are two event instances with same time stamp, the process mining algorithms based on the dependency relation use the order of the event instances in the log file. That might be correct if we only use one log file as a data source and we can assume that only one process writes event instances in the log file. In that case, the event instances are written in a first come – first serve fashion. In our case, we have a fusion of event instances from multiple log files into one log file. The order of the event instances with same time stamp is arbitrary. This leads to various heuristics net models, which are not necessarily semantically correct. Consulting a domain expert was necessary to find out which models are correct. One suggestion is when there are event instances with a same time stamp, the order of the event instances in the log file should be ignored and this should be optional input parameter in the process mining algorithms. Another solution is to implement semantic process mining where appropriate domain knowledge can be injected. For instance, when we have event instances with the same timestamp, then the end user should be able to define the order of the event instances in the file.

## *11.3 Lessons Learned*

The most important lessons from a design point of view are explained in the following subsections.

### 11.3.1. Building a Bridge between Academia and Industry

One of the hard lessons that had to be learnt along the way was how to bridge academia and industry. Following the TNO-ESI approach (Figure 1), the goal of this project was to use the process mining as a state-of-art academic discipline to achieve a real business goal in the industry. There is one significant problem in bridging academia and industry. In delivering academic results there are a lot of obstacles such as: missing data, noisy data, amount of data, and many more. One way to deal with the obstacles is to introduce assumptions that simplify the observed problem. The industry results are based on the business goal and often the obstacles have to be solved; they cannot be ignored or assumed. Therefore applying research technique for industrial purpose is not a trivial task. We have shown that the mapping issue in applying process mining, which is already recognized as a challenge, is not the only one. Solving only the mapping issue does not give results that meet the business goal of the customer profiles. We needed more information and we faced the missing information challenge. It is a big challenge because the relevant information is spread among different log file sources. The missing information can be found in different data sources, or it might be known by domain experts. In the worst case scenario the missing information cannot be found at all. That illustrates the complexity of the problems that can be faced in building a bridge between academia and industry.

### 11.3.2. Providing a Portable Solution

The goal of TNO-ESI is to develop generic know-how that can be reused in different domains. Abstraction of a certain solution can be a challenging task. Designing and

implementing a solution based on a certain example is quite a deterministic activity because we have an input and an output. Trying to get the same output of the system when we have a different input (more often undefined, because of the abstraction) requires high analytical and critical thinking skills. The designer has to oversee all the possible different inputs to the system.

### 11.3.3. Applying Test Driven Development

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test, and finally refactors the new code to acceptable standards. Applying TDD in this project helped in providing assurance to the candidate and to the stakeholders that the project artifacts are consistent. Especially it helped in implementing the Mapping-rule Repository where it was necessary to implement a set of regular expressions for each possible event instance. As the project grew and the data sets were changing the unit tests provided assurance that the proposed solution supports all the data variations.■

# 12. Project Management

An overview of the project management techniques applied in the project is given.

## *12.1 Introduction*

The current trend in project management is more focused on the business goal (Fowler, 2014). Therefore, a project implementation (Collier, 2011) in an agile fashion is a necessity. At the same time, the risk of log data availability (in sense of unavailable data and inaccessible data) required high level of flexibility in the project management. In order to evaluate the feasibility of the project and to eliminate possible risks, the project was conducted in two parts. The first part was analytical part. It included analytics use cases, whose goal was to define the customer profiles. The first part resulted with few executed analytics use cases that gave sufficient results. The results were used in the second part of the project. The second part also included analytics use cases, but the main accent was on infrastructure design, implementation, and validation. More detailed decomposition is given in the following section.

## *12.2 Work-Breakdown Structure (WBS)*

The project officially started with a kick-off meeting at ASML. For that purpose, a project roadmap (Figure 50) was created. The project roadmap contains different information intended for different groups of stakeholders including the main project phases and their breakdown.

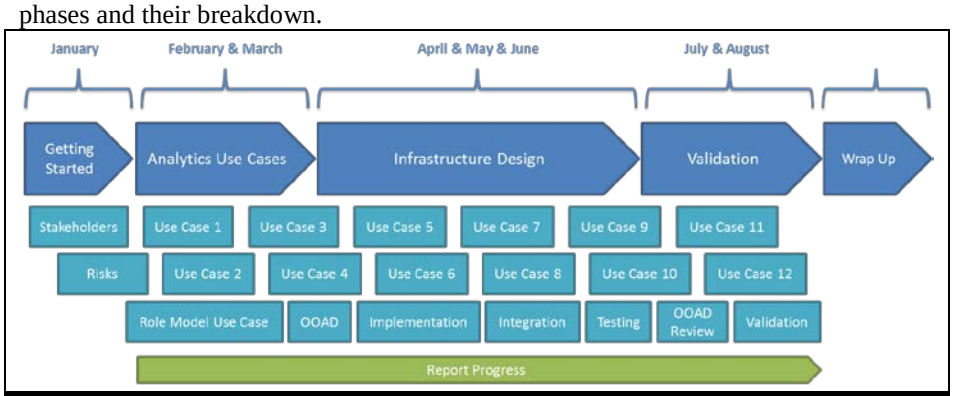


Figure 50 Customer Profiles - Project Roadmap

The project was decomposed in five phases: Getting Started, Analytics Use Cases, Infrastructure Design, Validation, and Wrap Up. Each phase had different end goal, but some of them had overlapping goals. For instance, the data analysis started in the second phase as a main activity, but also continued in the third phase. The report activity was part of the second, third, and the fourth phase. Each phase consists of several steps. The initial and the actual planning of the steps are elaborated in the following section.

## *12.3 Project Planning*

### 12.3.1. Initial

The first phase "Getting Started" included Introduction, Stakeholders Analysis, and Risk Analysis. In the second phase "Analytics Use Cases," a few analytics use cases were planed (Use Case 1 & 2) together with a role model selection. The data analysis continued in the third phase "Infrastructure Design." For this phase Object-oriented Analysis and Design, a Technology Overview, Implementation, and Integration were planned. The fourth phase "Validation" included Design Review, Valida-

tion and report writing. The final phase included project artifact delivery and project result presentation. The planned activity distribution in the project phases is shown in Table 8. The numbers represent the percentage of all the available time dedicated to a particular activity. The number calculation is based on planned days per activity with a step of 5%. In general, 5% of the time means 2-3 days.

Table 8 - Planned Activity Distribution

| | Data Analysis | OOAD | Implementation | Integration | Testing | OOAD Review | Validation | Report |
|---|---|---|---|---|---|---|---|---|
| Analytics Use Cases | 90% | | | | | | | 10% |
| Infrastructure Design | 15% | 25% | 30% | 20% | | | | 10% |
| Validation | | | | | 20% | 10% | 20% | 50% |

### 12.3.2. Final

The project was executed following the planned phases, but the activity distribution was not the same as the planned one. The Integration activity was completely removed from the plan, because it was decided that there is no integration necessary in this project. The time planed for Integration in the "Infrastructure Design" phase was used for Data Analysis, OOAD, and Implementation.

Writing the report activity took more time than the planned time. There are several reasons for that. First of all, a lot of decisions in the domain analysis were made later in the data analysis, and therefore a lot of changes in the report had to be done. At the same time providing an abstract solution that is portable to different TNO-ESI customers challenged the early design decisions. Reviewing the design decisions and updating the report influenced the initial planning. The final activity distribution in the project phases is shown in Table 9.

Table 9 - Final Activity Distribution

| | Data Analysis | OOAD | Implementation | Integration | Testing | OOAD Review | Validation | Report |
|---|---|---|---|---|---|---|---|---|
| Analytics Use Cases | 90% | | | | | | | 10% |
| Infrastructure Design | 25% | 25% | 45% | 0% | | | | 5% |
| Validation | | | | | 5% | 20% | 5% | 70% |

## *12.4 Project Management Techniques*

In different project phases different project management techniques were applied in order to achieve maximum optimization of the available time.

### 12.4.1. Analytics Use Case Selection

In the second phase "Analytics Use Cases", Analytics Use Case Selection was applied. This technique was introduced in the project to mitigate the risk of scattered infrastructure (see subsection "5.2.2. Scattered Infrastructure)." The technique minimizes the variations of the use cases in the project. It has two conditions that have to be satisfied. The first one is attribute description of the use cases. Each use case was described with the following attributes: analytics that is used (process mining, statistical analysis), log files format, number of aspects per log file, different machines, number of machines, and process reengineering or discovery. The second condition

is that with every new use case only one attribute change is allowed. Therefore, every next use case will differ by only one attribute. The second condition is illustrated in Figure 51. For example, Figure 51 shows that the Use Case 2 differs from the Use Case 1 only by the log file format and that the Use Case 4 differs from the Use Case 3 by two attributes, but from the Use Case 2 differs only by the type of the machine (equipment). The analytical use cases 1 – 8 in the project roadmap are executed with the Analytics Use Case Selection technique.

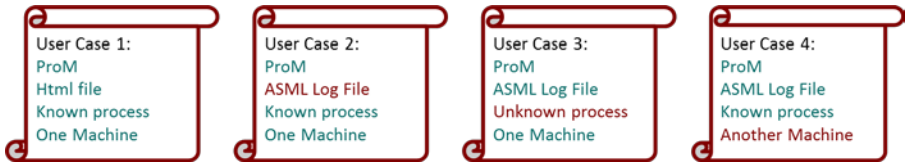


Figure 51 Analytics Use Case Selection

## 12.4.2. Role Model Use Case

Role Model Use Case technique is another technique used in the project to mitigate the risk of scattered infrastructure. This technique was applied at the end of the second phase "Analytics Use Cases." The goal of this technique is to select the most representative analytics use case as input for the infrastructure design. In this project the role model use case is a customer profile of the main production process (elaborated in "4.2. Applying Process Mining.")

## 12.4.3. Agile Development

In general, the whole project was executed in agile fashion. Every functional requirement for the infrastructure, but also for the analytical part was translated into user story and stored in the Project Backlog. For each sprint a certain amount of user stories was selected and executed. In the second phase "Analytics Use Cases", the use cases had more research character (meaning more uncertainty) and therefore the sprint length was three weeks. In the third and the fourth phase the sprint length was shortened to two weeks. Figure 52 shows a Burn down chart from one of the sprints executed in the phase "Infrastructure Design". For the purpose of agile project management, the TNO-ESI project management tool was used. The tool is based on Redmine project management platform that supports a backlog, a wiki, a tracking management, and a source code repository features.

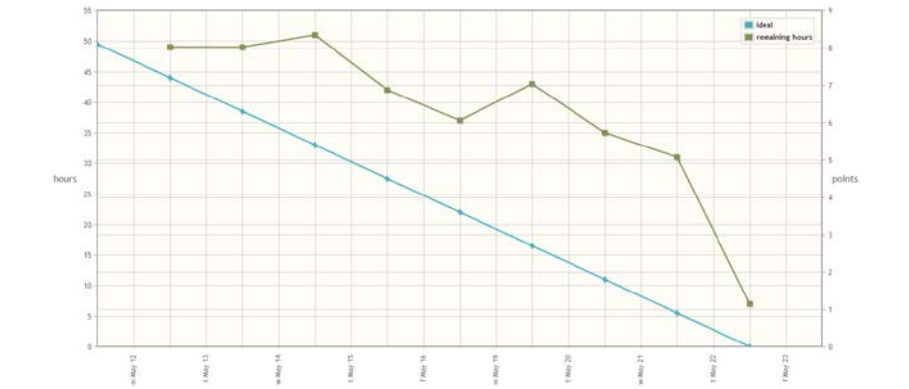


Figure 52 Burn down Chart - Infrastructure Design

■

# 13. Project Retrospective

In this chapter, a reflection on the candidate work is presented. It comprises the good practices, important moments and decisions, and the things that could have been done better. The chapter ends with revisiting the design criteria relevant for this project.

## *13.1 Reflection*

One of the main characteristics of this project is the variety of stakeholders. Figure 7 shows the different priorities of the different stakeholders. From one side, TNO-ESI stakeholders are more interested in the portability of the solution and are more oriented towards exploration. They need to explore all the possibilities and to provide the most suitable solution. On the other side, the ASML stakeholders are not concerned with the portability of the solution and are more goal-oriented. They have concrete problems and they need fast and stable solutions. At the same time, TU/e is interested in the project implementation process. Working in an environment like this and keeping all the stakeholders happy is a big challenge. One of the approaches that worked well in this project is the balance between the stakeholder priorities. This approach requires experience and it can bring risks. One of the risks is losing the balance and delivering results closer to one side. Keeping the TNO-ESI non-functional requirements on the top of the list, but at the same time looking at the ASML problem level (functional requirements) has brought satisfactory results. Balancing between the different stakeholder interests is not enough; the balance should lead to homogeneous project. Therefore, next to the balance, for a successful project, a selection of the complementary requirements and project management techniques is necessity.

Another important aspect in this project is getting the domain knowledge. It is a main precondition to delivering satisfactory results. In the context of this project, the domain knowledge consists of the process mining domain and the ASML domain. The thing that helped in this project is that the first part of the project was completely dedicated to getting the domain knowledge in terms of data analysis and process mining. Based on the results from the first part, a set of requirements was created as input for the infrastructure design and implementation. During the first part of the project, analytics use case selection (“12.4.2. Analytics Use Case Selection”) was applied. The use case selection mitigated the risk of scattered use cases but also provides step by step learning of the domain and the domain problems. There is a lot of material for process mining available online and therefore, a process mining domain expert was consulted in the middle of the first phase. The consultation helped in better understanding of the process mining problems. For instance, it was found out that the ProM tool version 5 is more suitable that the version 6. Moreover, during the consultations a process mining algorithm comparison was performed. The rule of thumb is that in getting the domain knowledge, an iterative learning can be beneficial. Also an early consultation with the domain experts, especially when the domain is a part of academic research is essential. In this context, early consultation means at the time when the candidate has an appropriate level of the domain knowledge, because only then the right questions can be asked.

## *13.2 Design opportunities revisited*

The design criteria that were selected as relevant for this project are: Genericity, Realizability, and Impact. Genericity in this project is presented by the non-functional requirement NFR1: Portability. There are several design decisions that were made in order to provide a portable solution. The first decision is to provide a modular archi-

tecture with coherent components based on the Pipes and Filters architectural pattern. The proposed system design is portable to different domains that confirm to the Extended Log Model elaborated in the section "4.2. Applying Process Mining." In order to measure the portability level of the proposed system an appropriate portability measurement was introduced. Each component in the system, based on the level of changes that has to be made, has a portability level.

Realizability (technical) is related with the feasibility of the project artifact. The implemented prototype showcases the implementation of the functional requirements, but also, what is more important; it shows the realizability of customer profiles based on process mining. During the project there were several design decisions that improved the realizability of the project. For instance, Figure 44, Figure 45, and Figure 46 depict Heuristics Net Models generated from customer log files. These models already show the feasibility of the idea to apply process mining for extracting customer profiles. The question that arises here is whether these models are relevant for the stakeholders or not. Do they provide sufficient business information for the stakeholders or not. The design decisions to include additional log files and to solve the problem with incomplete event instances resulted in delivering more appropriate results for the stakeholders (such as the models depicted in Figure 47 and Figure 48.)
The third criterion is the impact of the project artifact. In the context of this project it is considered the societal impact of the project artifact. The same design decisions for the previous design criteria are important for this criterion too. The genericity of the project provides higher impact because a larger set of stakeholders are satisfied. Providing generic solution means that the approach can be applied at any TNO-ESI client, not just ASML. With the realizability, the impact of the project is higher because it tells to which extend the project is realizable. The results from this project were accepted with a lot of enthusiasm from the stakeholders, especially within the ASML domain. Additional meetings with a broader group of stakeholders were conducted to share the results. One of the examples of the impact of this project is that the project "Customer profiles" plays a significant role in different activities within TNO-ESI (applying model-based testing) and ASML (Virtual Fab concept.)■

# Glossary

| | |
|---|---|
| PDEng | Professional Doctorate in Engineering |
| TU/e | Eindhoven University of Technology |
| TNO-ESI | Embedded System Innovation by TNO |
| IC | Integrated Circuit |
| ADD | Attribute driven Design |
| MXML | Mining eXtensible Markup Language - an XML-based syntax for event logs storing |
| UML | Unified Modeling Language |

# Bibliography


Bachmann, F., Len, B., & Robert, N. (2007). *Modifiability Tactics (Technical Report CMU/SEI-2007-TR-002).* Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University.

Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in pratice* (3rd ed.). Addison-Wesley Professional.

Brennan, K. (2009). *A Guide to the Business Analysis Body of Knowledge.* International Institute of Business Analysis.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-oriented software architecture: a system of patterns* (Vol. 1). New York, NY, USA: John Wiley & Sons.

Callo Arias, T., America, P., & Avgeriou, P. (2013). A Top-Down Approach to Construct Execution Views of a Large Software-Intensive System. *Software: Evolution and Process*, 233-260.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., . . . Stafford, J. (2010). *Documenting Software Architectures: Views and Beyond.* Addison-Wesley Professional.

Collier, K. W. (2011). *Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing.* Addison-Wesley Professional.

*Common Log Format*. (2014, July 15). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Common_Log_Format

de Medeiros, A. K., van der Aalst, W., & Alves de Medeiros, A. K. (2007). Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 245-304.

Fowler, M. (2014, January 20). *Thinking about Big Data*. Retrieved from Martin Fowler: http://martinfowler.com/articles/bigData/

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

Günther, C. W., & Verbeek, E. (2014). *XES Standard Definition.* Eindhoven: Eindhoven University of Technology.

Hage, W. R., Malaisé, V., Segers, R., Hollink, L., & Schreiber, G. (2011). Design and use of the Simple Event Model (SEM). *Web Semantics: Science, Services and Agents*, 128-136.

Images - ASML's customer magazine. (2013). *Integrated metrology delivers tightest on-product overlay(1)*. Eindhoven, The Netherlands: ASML. Retrieved from http://www.asml.nl/doclib/productandservices/images/asml_20130225_2013-0008_ASML_Images_magazine_winter_Final.pdf

*Key Competence Areas*. (2014, April 14). Retrieved from TNO-ESI: http://www.esi.nl/innovation-focus/competence-areas/

*Process Mining Tools*. (2014, January 15). Retrieved from ProM Tools: http://www.promtools.org

Rozinat, A. (2012, February 3). *Data Requirements for Process Mining*. Retrieved July 9, 2014, from http://fluxicon.com/blog/2012/02/data-requirements-for-process-mining/

Rozinat, A., Alves de Medeiros, A., Günther, C. W., Weijters, A., & van der Aalst, W. M. (2008). The Need for a Process Mining Evaluation Framework in Research and Practice. In A. ter Hofstede, B. Benatallah, & H.-Y. Paik (Eds.), *Business Process Management Workshops* (pp. 84-89). Berlin: Springer Berlin Heidelberg.

*Trace Documentation*. (2014, June 13). Retrieved from TRACE: http://trace.esi.nl/

van de Laar, P., & Punter, T. (2011). *Views on Evolvability of Embedded Systems.* Springer.

van de Laar, P., Tretmans, J., & Borth, M. (2013). *Situation Awareness with Systems of Systems.* Springer.

van der Aalst, M. W., & van Dongen, B. F. (2005). A Meta Model for Process Mining Data. *EMOI-INTEROP* (p. 12). Porto (Portugal): CEUR-WS.

van der Aalst, W. (2011). *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Berlin: Springer-Verlag.

van Hee, K., & van Overveld, K. (2012). *New criteria for assessing a technological design.* Eindhoven: Stan Ackermans Institute.

Weijters, A. J., van der Aalst, W. M., & Alves de Medeiros, A. (2006). *Process Mining with the Heuristics MinerAlgorithm.* Eindhoven, The Netherlands: Department of Technology Management, Eindhoven University of Technology.

# About the Author

Miroslav Janeski received his Diploma of Engineering (2008) and MSc degree in Intelligent Information Systems (2011) from the Faculty of Electrical Engineering and Information Technologies, Skopje, Macedonia. The main task of his master thesis entitled “Forecasting financial time series” was to find an optimal artificial neural network based model for forecasting Balkan stock exchanges. As a result he published two papers in international conferences, ICT Innovations 2010, Ohrid, Macedonia and ICIC 2011, Zhengzhou, China. After graduation he worked for four years as a software specialist at Neocom, Macedonia. His specialized in design and development of web applications for government agencies. In 2012 he joined the PDEng programme in Software Technology at the Eindhoven University of Technology. His final project as a part of the PDEng programme is entitled “Customer Profiles: Extracting usage models from log files.” The project was executed at ASML under supervision of Embedded System Innovation by TNO.

3TU.School for Technological Design, Stan Ackermans Institute offers two-year postgraduate technological designer programmes. This institute is a joint initiative of the three technological universities of the Netherlands: Delft University of Technology, Eindhoven University of Technology and University of Twente. For more information please visit: www.3tu.nl/sai.