Conformance Analysis of ASML's Test Process

A. Rozinat[1], I.S.M. de Jong[2], C.W. Günther[1], and W.M.P. van der Aalst[1]

[1] Department of Information Systems, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
{a.rozinat,c.w.gunther,w.m.p.v.d.aalst}@tue.nl
[2] ASML, P.O. Box 324, NL-5500 AH, Veldhoven, The Netherlands
ivo.de.jong@asml.com

Abstract. Process mining allows for the automated discovery of process models from event logs. These models provide insights and enable various types of model-based analysis. However, in many situations already some normative process model is given, and the goal is not to discover a model, but to check its conformance. The process mining framework ProM provides a *conformance checker* able to investigate and quantify deviations between the real process (as recorded in the event log) and the modeled process. The conformance checker is one of the few tools available today that is able support *regulatory compliance*, i.e., ensuring that organizations and people take steps to comply with relevant laws, regulations, and procedures. In this paper, we report on a case study where the ProM framework has been applied to the test processes of ASML (the leading manufacturer of wafer scanners in the world). In this case study, we focus on the conformance aspect and compare the test process as it is really executed to the idealized reference model that ASML is using to instruct their test teams. This revealed that the real process is much more complicated than the idealized reference process. Moreover, we were able to suggest concrete improvement actions for the test process at ASML.

Keywords: process mining, conformance checking, case study

1 Introduction

Corporate scandals have triggered an increased interest in corporate governance, risk management, and regulatory compliance. As a result new regulations such as the Sarbanes-Oxley Act, Basel II, HIPAA, etc. were introduced. Some of the key elements are: accountability, auditability, privacy, documentation, policy, and manageability of information. In this paper, we focus on compliance. In particular, we focus on the question *"Do organizations and people do what is documented in process models?"*. To address this question we conduct a case study where we apply ProM's *conformance checker* [22] to one of the processes of ASML.

To position our work, we first introduce *process mining*. The basic idea of process mining is to discover, monitor and improve *real* processes by extracting

knowledge from event logs. Today many of the activities occurring in processes are either supported or monitored by information systems. Consider for example ERP, WFM, CRM, SCM, and PDM systems to support a wide variety of business processes while recording well-structured and detailed event logs. However, also high-tech devices such as X-ray machines, web services, etc. record events. All of these applications have in common that *there is a notion of a process* and that *the occurrences of activities are recorded in so-called event logs*. Assuming that we are able to log events, a wide range of *process mining techniques* comes into reach. The basic idea of process mining is to learn from observed executions of a process. Process mining can be used to (1) *discover* new models (e.g., constructing a Petri net that is able to reproduce the observed behavior), (2) to check the *conformance* of a model by checking whether the modeled behavior matches the observed behavior, and (3) to *extend* an existing model by projecting information extracted from the logs onto some initial model (e.g., show bottlenecks in a process model by analyzing the event log). All three types of analysis have in common that they assume the existence of some *event log*.

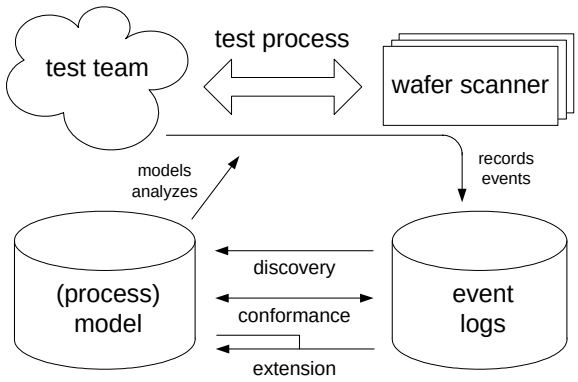


Fig. 1. Based on the event logs of the wafer scanners, three classes of process mining techniques are possible: (1) "discovery", (2) "conformance", and (3) "extension"

In this paper, we focus on conformance checking and apply ProM's *conformance checker* [22] to the test process of ASML. ASML is the world's leading manufacturer of chip-making equipment and a key supplier to the chip industry. ASML designs, develops, integrates and services advanced systems to produce semiconductors, e.g., wafer scanners that print the chips. There is an ongoing effort to reduce the line width on silicon wafer to enhance the performance of the manufactured semi-conductors. Every new generation of wafer scanners is balancing on the border of what is technologically possible. As a result, the testing of manufactured wafer scanners is an important but also time-consuming process. Every wafer scanner is tested in the factory of ASML. When it passes all tests, the wafer scanner is disassembled and shipped to the customer where

the system is re-assembled. At the customer's site, the wafer scanner is tested again. Clearly, testing is a time-consuming process and takes several weeks at both sites. Since time-to-market is very important, ASML is involved in an ongoing effort to reduce the test period. To assist ASML in these efforts, we applied process mining techniques to their test processes. Rather than focusing on fault detection, the subject of study is here the test process itself.

At any point in time, ASML's wafer scanners record events that can easily be distributed over the internet. Hence, any event that takes place during the test process can be recorded easily. The availability of these event logs and the desire of ASML to improve the testing process triggered the case study reported in [24]. Using process discovery, we tried to answer the question "How are the tests actually executed?", i.e., based on the event logs we automatically constructed process models showing the ordering and frequency of test activities. In this paper, we then compared them to the idealized reference model. This revealed that the real process is much more complicated than the idealized reference model that ASML is using to instruct the test teams. The reference model shows a rather structured process while in reality the testing process requires much more flexibility. Using conformance checking techniques, we investigated this further by answering the question "How compliant are the actual test executions to the reference process?". Through conformance checking we were able to quantify and pinpoint the deviations of the real test process from the idealized reference model. For the case study we used our *ProM framework*[3]. ProM is open source and uses a plug-able architecture, e.g., developers can add new process mining techniques by adding plug-ins without spending any efforts on the loading and filtering of event logs and the visualization of the resulting models [1]. Version 5.0 of ProM provides 230 plug-ins. For example, there are more than 15 plug-ins to discover process models from event logs.

The remainder of this paper is organized as follows. Section 2 reviews related work both in process mining and the test process optimization domains. Next, the context of the case study is described in more detail in Section 3. Section 4 presents the results of this study, and concrete improvement actions for the ASML test process are proposed in Section 5. Section 6 concludes the paper.

2 Related Work

Since the mid-nineties several groups have been working on techniques for process mining [3, 4, 10], i.e., discovering process models based on observed events. In [2] an overview is given of the early work in this domain.

The paper builds on the conformance checking techniques presented in [22]. These techniques are inspired by the fitness function used in genetic process mining [20]. Also related is the work by Cook [9, 8] where the event streams of a process model and a log are compared based on string distance metrics.

Process mining can be seen in the broader context of Business Process Intelligence (BPI) and Business Activity Monitoring (BAM). In [14, 25] a BPI toolset

[3] ProM can be freely downloaded from *http://prom.sf.net/*.

on top of HP's Process Manager is described. The BPI toolset includes a so-called "BPI Process Mining Engine". In [21] Zur Mühlen describes the PISA tool which can be used to extract performance metrics from workflow logs. Similar diagnostics are provided by the ARIS Process Performance Manager (PPM) [18]. It should be noted that BPI tools typically do not allow for process discovery and conformance checking, and offer relatively simple performance analysis tools that depend on a correct a-priori process model [17]. In [5] it is suggested that database technology can play an important role in assisting compliance with the internal control provisions of SOX.

Most of the work on conformance checking has been done on model analysis without taking into account event logs. For example, in [13] it is checked whether business processes are compliant with business contracts, and in [12] a non-monotonic deontic logic of violations is used to detect all obligations that will not necessarily be fulfilled by executing the model. In [19], the authors introduce OPAL, a compliance-checking framework, and related tools, including a static method to check business process models against compliance rules. In [11] semantically annotated process models and formal representations of compliance requirements are compared for auditing BPMN process models for compliance with legislative/regulatory requirements, and for exploring alternative modifications to restore compliance in the event that the processes are found to be non-compliant.

The conformance checking techniques used in this paper are generic and can be applied to various types of processes. Hence, it can be used to analyze the logs of ERP, WFM, CRM, SCM, and PDM systems. However, in this paper we apply our techniques to a particular type of process: testing ASML's wafer scanners. See [7, 6] for more information on testing and test design in this particular setting.

The case study reported in [24] already explores the applicability of process mining to improve ASML's test process, and, for example, analyzes idle times to shorten the time-to-market. In this paper, we investigate the differences between the actual, executed test sequences and the planned test sequences in more detail.

3 Case Study

This section introduces the case study where process mining was applied to the test process of ASML's wafer scanners. After describing the test process of a wafer scanner in more detail (Section 3.1), we look at the log data recorded during these tests (Section 3.2). The event logs serve as input for our process mining techniques and the results of their analysis are described in Section 4.

3.1 The Test Process

The whole test process of a waver scanner at ASML consists of three phases: (1) the calibration phase, (2) the test phase (the actual testing), and (3) the final qualification phase. The whole process takes several weeks. When finished, the wafer scanner is partly taken apart and shipped to a customer. A part of the

calibration and test phase is repeated at the customer site, after re-assembling the wafer scanner.

Why is this test process so important for ASML? ASML operates in a market where the time-to-market of system enhancements and the time-to-market of new system types is critical. Wafer scanners are continuously enhanced. As a result, the number of manufactured wafer scanners of a single type is typically less than 50. With each new type, parts of the calibration and test phase are adjusted. On average five different system types are manufactured in parallel. The short time-to-market, the constant innovation, and the high value of wafer scanners make testing very important. Spending too much time on testing will result in high inventory costs and lost sales. However, inadequate tests will result in systems which are malfunctioning.

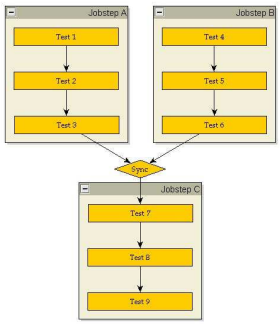


Fig. 2. Example sequence of three job steps with a synchronization point

Sets of calibration and test actions are grouped into so-called *job steps*. These job steps are executed according to a certain sequence. Only large changes in the system design result in changes in the job step sequence, so the job step sequence can be considered a fixed sequence across different systems. Some of these job-steps can be executed independently of each other. An example sequence of three job steps is depicted in Figure 2. Note that in ASML such structures are referred to as "sequences". However, strictly speaking these are process models rather than sequences. The synchronization point *sync* enforces that both job step A and job step B must be finished before job step C can start. Each calibration action or test case can fail. Some of the causes for test failure can require a replacement of a faulty hardware component. The duration of this replacement can take up to hours or longer. If such a failing test is in the example job step A, then the independent job step B can be started to ensure maximal progress. When the replacement hardware becomes available, either job step B is finished first and then job step A is finished, or the other way around. Job step C is started when job step A and B are both finished. Note that a failure in a test case in job step C results in no activity on the system (idle time) until the malfunctioning part of system is fixed and testing can continue.

Some of the causes for a failure can be fixed immediately. For example, some parameters in the system can be out of specification. This measurement information can now be used to adopt the control set-points in the system. After a second measurement, the parameters can be within specification and the test passes. Most of the software which executes the tests is constructed such that this fast-fix loop is automated. Testing, calibration and retesting is performed in a single test. Finally, a change in low-level machine parameters, because of a hardware replacement, can cause a re-execution of a previous job-step. For instance, the profile of some of the mirrors in a wafer scanner are measured and stored in X,Y and Z directions. This profile information is used in all positioning calibrations, such that the errors caused by the non-flat mirrors are minimized. Replacing these mirrors results in a new profile. For this reason, a large set of job steps needs to be redone if a faulty mirror is replaced in one of the last job steps in the sequence. In summary, job steps are executed according to a fixed sequence for a set of machine types. The sequence allows variation of the detailed tests within the limits of the synchronization points. The actual execution of tests results in failing test cases, which can result in a lengthy re-test of parts of the sequence depending on the failure at hand. For ASML, the goal is to minimize the waiting time for a hardware fix (idle time) and to reduce the re-execution of parts of the job-step sequence. This goal could be easily met by testing all components and building blocks thoroughly before and during system assembly. However, the increase in test effort would result in an increase of the total test duration and therefore an increase in time-to-market. This is the main reason that testing everything thoroughly beforehand is not considered a solution, so the main goal is a reduction of the duration of the test process and not cutting costs.

3.2 Log Data and Conversion

Each wafer scanner in the ASML factory produces a log of the software tests which are executed. The manual assembly and calibration actions are not logged and appear as idle time in this log. The wafer scanner is calibrated and tested using calibration and performance software, indicated in the logging as a four-letter code. The logging contains the start and stop moment of each test. The idle time, i.e., the time between stop of the previous test and the start of the next test, is not specified in detail. This idle time has a number of causes, ranging from inexperienced operators reading the procedures, the end of the automated test queue during the night to diagnosing a problem, or waiting for additional parts. Some parts of the test sequence are executed in an automated fashion. The operator starts a test queue which contains a set of test cases which are executed in a sequence. This test queue can also contain part of the recovery and retry sequence for certain failing test cases. The recovery or retry tests are executed depending on the outcome of a test in the queue.

An example fragment of the test log of one of the wafer scanners is depicted in Figure 3(a). Each line corresponds to the execution of one test. The number at the beginning of the line identifies the machine (i.e., the wafer scanner) that is

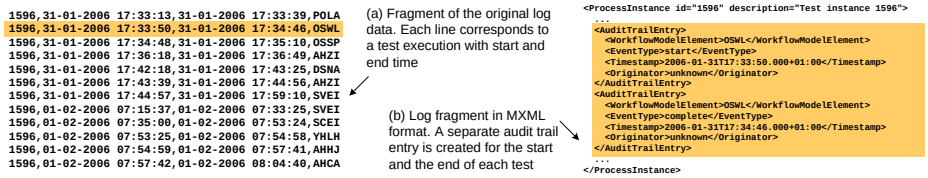


Fig. 3. Converting the log into the MXML format

tested. Afterwards the start time, the completion time, and the four-letter code for the executed test are recorded.[4]

To analyze the log data with ProM we had to convert them into the MXML[5] format. This was realized by a custom-built converter plug-in for the *ProM*import *framework*[6]. ProM*import* facilitates log transformation tasks and provides converter plug-ins for a wide variety of systems to the XML format used by ProM [15]. In the MXML format, a log is composed of process instances (i.e., cases) and within each instance there are audit trail entries (i.e., events) with various attributes. These attributes refer to, for example, data fields, timestamps, or transactional information (i.e., whether the activity was scheduled, started, or completed). Depending on the kind of information that is in the log, we may be able to answer different questions about the process. Figure 3(b) depicts the MXML log fragment for the highlighted test from Figure 3(a). One can see that the start and the completion of the test are captured by separate audit trail entries (including the corresponding timestamps), and that the enclosing process instance (i.e., the case) corresponds to the tested machine.

Note that the logging takes place on the test-code level, and that there is no reference to the job step in whose context the test is performed. However, in addition to the log data and the job step reference sequence, ASML also provided us with an additional document specifying which test codes should be executed in which job step. In this mapping, there are a number of tests that appear in more than one job step (i.e., are executed in different phases of the test process).

4 Conformance Analysis Results

In the following, we provide a summary of the results from analyzing the test process execution logs. (More details about the specific process mining techniques and used ProM plug-ins can be found in our technical report [23].) In Section 5, these results are then evaluated from an ASML perspective and concrete improvement actions are proposed.

In most domains, we usually see a large number of relatively short log traces, i.e., many process instances with just a few events. For example, when looking

[4] Note that both the actual machine numbers and the four-letter test codes have been anonymized for confidentiality reasons.

[5] The XML schema definition is available at *http://www.processmining.org/*.

[6] ProM*import* can be freely downloaded from *http://promimport.sf.net/*.

at processes related to patient flows, insurance claims, traffic fines, etc., then there are typically thousands of cases each containing less than 50 events. When we examine the log, it becomes clear that this test process has very different characteristics, since there are just a few cases (i.e., machines) but for each machine there may be thousands of log events. In the initial data set we faced process instances that contained more than 50000 log events (each indicating either the start or the completion of a specific test). As mentioned earlier, the test process of a wafer scanner lasts for several weeks and is partly repeated after the machine has been re-assembled at the customer, thus explaining the huge number of events per machine. From a larger set of machines we selected 24 machines that fulfilled our criteria: (1) the test process needed to be completed, (2) only include the test period on the ASML (and not the customer) site, (3) belong to the same family (recall that typically not more than 50 wafer scanners of the same type are produced), and (4) not be a pilot system (as a pilot system is used for development testing and not for manufacturing qualification). These 24 cases comprise 154966 log events in total, and the number of log events per process instance (i.e., the length of the executed test sequence) ranges from 2820 to 16250. Finally, we can see that there are 720 different types of audit trail entries in the log, which corresponds to 360 different four-letter test codes as each test is captured by both a 'start' and 'complete' event.

Furthermore, we are interested in analyzing the job steps, i.e., the test phases that can be associated to the reference sequence. To be able to analyze the log on the job-step level, we first have to apply certain *filtering* techniques. Recall that there is no information about job steps recorded in the log, but that we have obtained a document specifying which tests need to be executed for each job step. In this mapping, there are 184 out of the 360 detected test codes associated to a job step. This means that 176 of the four-letter codes cannot be connected to a specific job step (in the remainder of this paper we call them "unmapped" codes). They mainly correspond to additional (more specific) tests that are executed as part of the diagnosis process after a failure. At the same time, there are 49 out of the 184 mapped test codes that are associated to more than one job step, i.e., they occur in different phases of the test process (in the remainder we call them "multiple" codes). The rest of the four-letter codes (i.e., 135 test codes) can be unambiguously mapped onto a specific job step.

In Figure 4 we show as an example how a part of the log fragment from Figure 3 is transferred to the job-step level[7] using a combination of multiple log filters. As a first step, we mapped each of the unambiguous test codes onto their corresponding job step identifier, or the 'multiple' or 'unmapped' category if this was not possible. For example, Figure 4 shows that the tests 'OSWL', 'OSSP', and 'AHZI' are associated to the job step 'e', while the test 'DSNA' cannot be mapped to any job step (i.e., 'unmapped'). As a next step, we abstracted from all events that occurred between the first and the last event belonging to the same job step in a row. For example, in Figure 4 only the beginning of the first

[7] Note that, again, the actual job step names have been replaced by simple letter codes for confidentiality reasons.

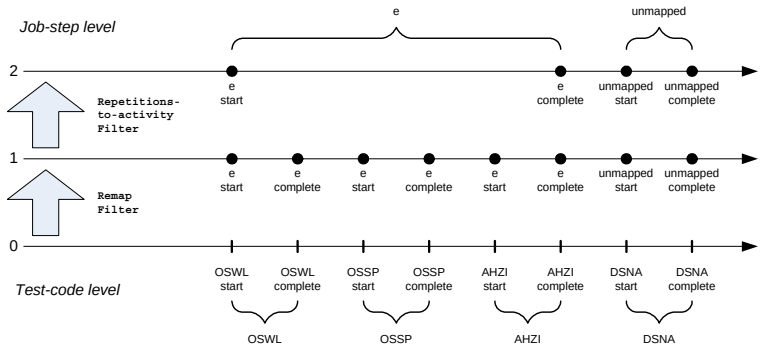
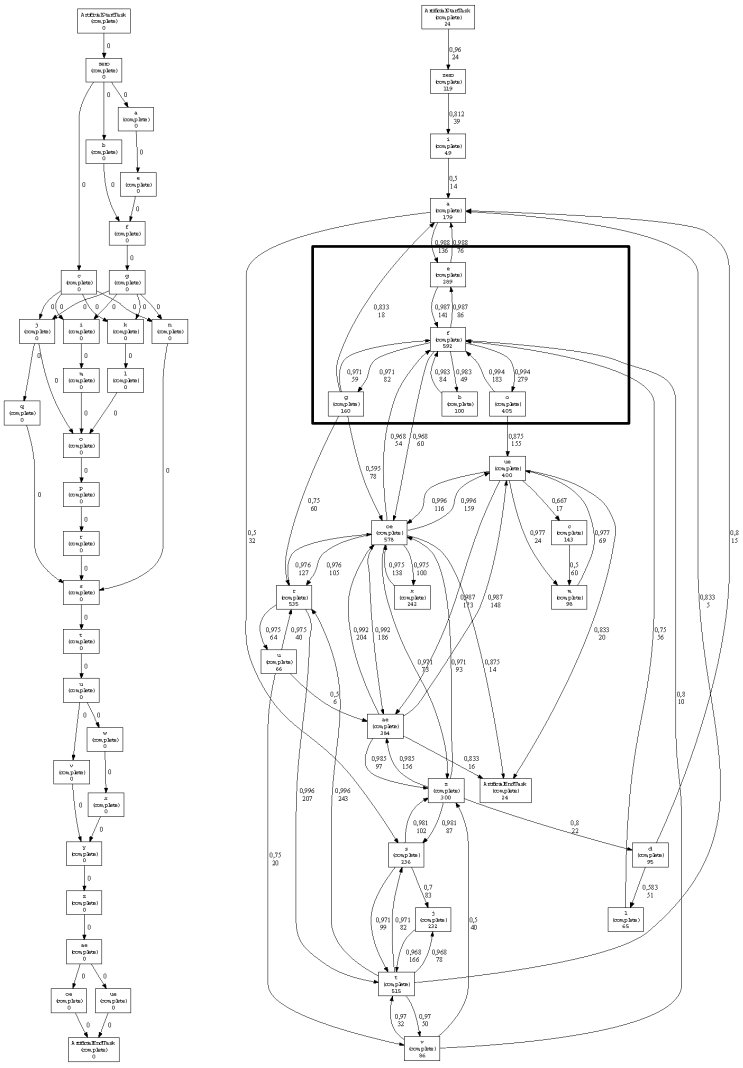


Fig. 4. A combination of filtering techniques was applied to bring the log data from the test-code level to the job-step level

occurrence of a test in job step 'e' (i.e., test 'OSWL') and the end of the last occurrence in job step 'e' (i.e., test 'AHZI') is retained. Note that using this mapping, now also idle times within one job step are covered by the overall job-step duration (for example, the idle time between the completion of test 'OSWL' and the start of test 'OSSP'). As a result, only changes between job steps become visible in the log, which we will use in the following for process discovery on the job-step level.

We now want to apply process *discovery* techniques to gain insight into the actual flow of the test process to find out where re-executions were often necessary. Process discovery algorithms automatically construct a process model based on the behavior that was observed in the event log. While it is interesting to visualize dependencies on the test-code level, we also want to analyze the process on the job-step level to compare the discovered model to the existing reference sequence. The translated reference sequence is depicted in Figure 5(a), and it reflects the normal flow of the test sequence if nothing goes wrong (i.e., if no test fails). We already know that in reality parts of the test sequence need to be repeated in certain occasions. This also becomes visible in the discovered model based on the log filtered for *job step executions* (cf. Figure 4), which is depicted in Figure 5(b). Note that the discovered process model allows for considerably more paths than the reference model. Figure 6(a), shows the framed part of the mining result in more detail, where one can easily recognize the repetitive nature of the *real* (as opposed to the ideal, i.e., reference) test process. Note that the numbers next to the arcs show the importance of the different paths (the lower number indicates how often this connection was observed in the log, while the upper number indicates the heuristic strength of the corresponding connection).

So far, we have seen that it is possible to automatically discover models which represent the behavior that was observed in the event log. But how well is the actual process represented by these models? And to which extent does the observed process comply with the behavior specified in the reference model?



(a) Reference model (b) Discovered process model

Fig. 5. Translated reference sequence and discovered process model on job-step level

Where in the process do most of the deviations occur? These are questions that are addressed by *conformance* techniques. In the following we use conformance checking [22] to analyze the conformance of both the reference model and the discovered model on the job-step level (cf. Figure 5) with respect to our test log.

Next to visualizing the discrepancies between an event log and a given process model, conformance checking can also be used to measure the degree of *fitness* based on the amount of missing and remaining tokens during log replay [22],

Table 1. Fitness values (f) indicating the degree of compliance for each of the test instances with respect to both the reference model and a discovered process model. Clearly, the discovered model fits much better than the reference model

Machine ID	*Reference Model*	*Discovered Model*	*Job-step Events*	*Test-code Events*
0431	$f = 0.309$	$f = 0.751$	238	6504
0278	$f = 0.385$	$f = 0.828$	270	6136
0185	$f = 0.376$	$f = 0.717$	206	5710
0466	$f = 0.356$	$f = 0.745$	422	8162
0391	$f = 0.384$	$f = 0.727$	159	3902
1722	$f = 0.334$	$f = 0.760$	301	6270
1694	$f = 0.397$	$f = 0.782$	526	10408
1256	$f = 0.410$	$f = 0.744$	222	5722
1343	$f = 0.399$	$f = 0.701$	130	5360
1981	$f = 0.357$	$f = 0.667$	551	12670
1754	$f = 0.402$	$f = 0.776$	192	16250
1662	$f = 0.414$	$f = 0.769$	182	3830
1453	$f = 0.405$	$f = 0.596$	164	6410
1298	$f = 0.378$	$f = 0.424$	170	3852
1876	$f = 0.356$	$f = 0.753$	150	4538
1656	$f = 0.368$	$f = 0.656$	126	2820
1099	$f = 0.424$	$f = 0.672$	193	3946
1919	$f = 0.337$	$f = 0.727$	205	5048
1348	$f = 0.410$	$f = 0.638$	184	5240
1596	$f = 0.410$	$f = 0.581$	224	5784
1164	$f = 0.376$	$f = 0.672$	499	10860
1032	$f = 0.324$	$f = 0.706$	301	6896
1794	$f = 0.394$	$f = 0.734$	114	2972
1160	$f = 0.405$	$f = 0.770$	186	5676
Average	$\mathbf{f = 0.375}$	$\mathbf{f = 0.711}$	246.458	6456.917

i.e., it quantifies to which degree the log traces comply with a given process model. This fitness analysis clearly indicates that the discovered model is much more representative for the observed test process than the reference model (cf. fitness values in Table 1). Table 1 contains the fitness values for each of the test instances with respect to both the reference sequence and the discovered model on the job-step level as depicted in Figure 5, whereas possible values range from 0.0 (corresponds to the case where the model and the log do not fit at all) to 1.0 (i.e., model and log fit to 100%). Furthermore, it shows how many job step executions were contained in the filtered log for each machine (column before the last column in Table 1), and how many test code events were originally recorded for this machine (last column in Table 1). Finally, in the bottom row average values are given for all the 24 machines. We can see that, although the discovered process model does not completely "match" the behavior observed in the log, it clearly fits much better than the reference sequence. This is not surprising as we already know that—in contrast to the discovered model—the reference model does not capture the possible repetitions in the process at all, but it

describes the ideal flow of the process if nothing goes wrong. So, the discovered model is a much better representation of the test process as it took place, which demonstrates that process mining can provide insight into how processes are *really* executed.

5 Evaluation From ASML Perspective

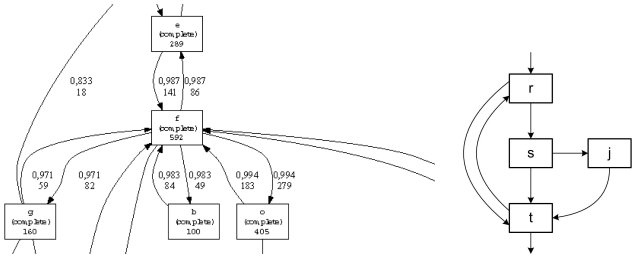
To identify concrete improvement suggestions, we evaluated the presented process mining results from an ASML perspective. For this, the order of job steps was analyzed. The job-step order is the sequence in which job steps are executed in the factory. Some variation is allowed, but not too much. We investigated whether—according to the discovered model as in Figure 5—the test process followed the reference process (including the allowed variations).

When we investigated whether the real process followed the reference process, considering the allowed variations, we obtained three types of results: (1) job steps that are actually executed on a different place in the reference sequence (i.e., deviations from the process model shown in Figure 5(a)), (2) groups of highly connected job steps, and (3) job steps that are not in the reference sequence but in the test log. In the following, we describe them in more detail.

(1) It appeared that job step 'i' was positioned in 81% of the cases just after the 'zero' job step, i.e., at the beginning of the discovered process model, while—according to the reference sequence—it should be executed in the middle of the test process. While looking for possible root causes for this difference, we realized that a newer version of the reference sequence was released in the end of 2006. The main change in the new reference sequence was that job step 'i' and 'j' were positioned just after the 'zero' job step at the beginning of the test sequence. The analyzed systems were build up according to the new sequence for job step 'i'. Interestingly, job step 'j' was still found in the original position. If job step 'j' is really to be executed in the beginning of the sequence, then active steering should take place to align the test execution. Note that we also re-checked the conformance of the test log with respect to the updated reference sequence, but the fitness values did not change significantly (on average $f \approx 0.45$).

(2) Two highly connected groups of job steps are included in the discovered process model. The first group is depicted in Figure 6(a), a strong connection between job step 'f' and a number of other job steps: 'e', 'b', 'g' and 'o'. These connections are bi-directional between 'f' and the other job steps. A reason for this effect could be that any execution of the job steps 'e', 'b', 'g' and 'o' results in a re-execution of job step 'f'. Job step 'f' is a relatively short job step which can be executed automatically. As a result, the entire test set is executed. Specific parts of the test set in job step 'f' could be faster when job step 'f' needs to be executed after job step 'e', 'b', 'g' and 'o' are executed. In general, speeding up job step 'f' is beneficial because it is executed multiple times in the entire sequence.

The second highly connected group is centered around job steps 'r', 's', 't', and 'j'. The mined process showed the following pattern (see Figure 6(b)). Job



(a) Framed area in Figure 5(b) (b) Other group

Fig. 6. Highly connected groups of job steps, which have been identified based on the process mining results presented in Section 4

step 'r' and 't' are bi-directionally connected. Job steps 'r', 'j' and 't' are illumination steps, while job step 's' is a non-illumination step. The root cause of a failure of job step 't' is solved by job step 'r'. A re-execution of job step 'r' causes a re-execution of job step 's' (and possibly 'j'). An improvement proposal would be to introduce a more thorough test in job step 'r' (i.e., add a similar test to the one in job step 't') which causes that, if the failure occurs, it already occurs in job step 'r' and can be immediately fixed in job step 'r'. This prevents the re-execution of job step 's' (and possibly 'j').

(3) One of the feedback loops revealed that job step 'd' is executed, although it is not in the reference sequence. Job step 'd' is currently not investigated to be improved to decrease the cycletime, because this job step is not supposed to be executed. The process mining results revealed that job step 'd' is executed as part of a recovery plan. Job step 'd' could be further investigated for cycle time reduction.

The above analysis illustrates that process mining can be applied to check the conformance of processes, i.e., deviations can be detected and analyzed.

6 Conclusion

Using a test process in ASML, we have illustrated the applicability of ProM's conformance checker. The case study clearly shows that, given an event log and a process model as input, conformance checking can be used to detect deviations. The severity of these deviations can be qualified and possible causes can be analyzed. Hence, conformance checking is a useful tool in assessing regulatory compliance.

The case study is a bit a-typical, i.e., regulatory compliance is often associated with regulations such as the Sarbanes-Oxley Act, Basel II, and HIPAA. These regulations focus on banks, insurance companies, governmental agencies, hospitals, etc. However, ProM's conformance checker is generic and applicable to any notation that can be mapped onto Petri nets. Moreover, the case study within ASML illustrates the trend that more and more devices are connected to

the internet. Another example is the "CUSTOMerCARE Remote Services Network" of Philips Healthcare (PH). This is a worldwide internet-based private network that links PH equipment to remote service centers. Any event that occurs within an X-ray machine (e.g., moving the table, setting the deflector, etc.) is recorded and can be analyzed [16]. The logging capabilities of the machines of PH illustrate the increasing availability of event data. The omnipresence of such detailed logging will have dramatic effects on compliance. While today many processes are not auditable because vital information is missing, it is clear that much more audit data will be available in the near future.

Acknowledgements

This research is supported by the Technology Foundation STW, EIT, and the Tangram and IOP programs of the Dutch Ministry of Economic Affairs. Furthermore, we want to thank ASML for their cooperation.

References

1. W.M.P. van der Aalst, B.F. van Dongen, C.W. Günther, R.S. Mans, A.K. Alves de Medeiros, A. Rozinat, V. Rubin, M. Song, H.M.W. Verbeek, and A.J.M.M. Weijters. ProM 4.0: Comprehensive Support for Real Process Analysis. In J. Kleijn and A. Yakovlev, editors, *Proceedings of the ICATPN 2007*, volume 4546 of *Lecture Notes in Computer Science*, pages 484–494. Springer-Verlag, Berlin, 2007.
2. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
3. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
4. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
5. R. Agrawal, C.M. Johnson, J. Kiernan, and F. Leymann. Taming compliance with sarbanes-oxley internal controls using database technology. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006)*, page 92. IEEE Computer Society, 2006.
6. R. Boumen, I.S.M. de Jong, J.M.G. Mestrom, J.M. van de Mortel-Fronczak, and J.E. Rooda. Integration and Test Sequencing for Complex Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 39(1):177–187, 2009.
7. R. Boumen, I.S.M. de Jong, J.W.H. Vermunt, J.M. van de Mortel-Fronczak, and J.E. Rooda. Test Sequencing in Complex Manufacturing Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 38(1):25–37, 2008.
8. J.E. Cook, C. He, and C. Ma. Measuring Behavioral Correspondence to a Timed Concurrent Model. In *Proceedings of the 2001 International Conference on Software Mainenance*, pages 332–341, 2001.
9. J.E. Cook and A.L. Wolf. Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. *ACM Transactions on Software Engineering and Methodology*, 8(2):147–176, 1999.

10. A. Datta. Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches. *Information Systems Research*, 9(3):275–301, 1998.
11. A. Ghose and G. Koliadis. Auditing Business Process Compliance. In B.J. Kramer, K.J. Lin, and P. Narasimhan, editors, *Fifth International Conference on Service-Oriented Computing (ICSOC 2007)*, volume 4749 of *Lecture Notes in Computer Science*, pages 169–180. Springer-Verlag, Berlin, 2007.
12. G. Governatori, J. Hoffmann, S. Sadiq, and I. Weber. Detecting Regulatory Compliance for Business Process Models through Semantic Annotations. In *4th International Workshop on Business Process Design*, 2008.
13. G. Governatori, Z. Milosevic, and S. Sadiq. Compliance Checking Between Business Processes and Business Contracts. In *10th International Enterprise Distributed Object Computing Conference (EDOC 2006)*, pages 221–232. IEEE Computing Society, 2006.
14. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.C. Shan. Business process intelligence. *Computers in Industry*, 53(3):321–343, 2004.
15. C.W. Günther and W.M.P. van der Aalst. A Generic Import Framework for Process Event Logs. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops, Workshop on Business Process Intelligence (BPI 2006)*, volume 4103 of *Lecture Notes in Computer Science*, pages 81–92. Springer-Verlag, Berlin, 2006.
16. C.W. Günther, A. Rozinat, W.M.P. van der Aalst, and K. van Uden. Monitoring Deployed Application Usage with Process Mining. BPM Center Report BPM-08-11, BPMcenter.org, 2008.
17. H. Hess. Monitoring, Analyzing and Optimizing Corporate Performance: State of the Art and Current Trends. In *Agility by ARIS Business Process Management*, pages 235–250. Springer-Verlag, Berlin, 2006.
18. IDS Scheer. ARIS Process Performance Manager (ARIS PPM): Measure, Analyze and Optimize Your Business Process Performance (whitepaper). IDS Scheer, Saarbruecken, Gemany, http://www.ids-scheer.com, 2002.
19. Y. Liu, S. Mueller, and K. Xu. A Static Compliance-Checking Framework for Business Process Models. *IBM Systems Journal*, 46(2):335–361, 2007.
20. A.K. Alves de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. Genetic Process Mining: An Experimental Evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007.
21. M. zur Mühlen and M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In R. Sprague, editor, *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33)*, pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.
22. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, 2008.
23. A. Rozinat, I.S.M. de Jong, C.W. Günther, and W.M.P. van der Aalst. Process Mining of Test Processes: A Case Study. BETA Working Paper Series, WP 220, Eindhoven University of Technology, Eindhoven, 2007.
24. A. Rozinat, I.S.M. de Jong, C.W. Günther, and W.M.P. van der Aalst. Process Mining Applied to the Test Process of Wafer Steppers in ASML. *Accepted for publication in IEEE Transactions on Systems, Man, and Cybernetics–Part C*, 2009.
25. M. Sayal, F. Casati, U. Dayal, and M.C. Shan. Business Process Cockpit. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 880–883. Morgan Kaufmann, 2002.