Process Mining Applied to the Test Process of Wafer Steppers in ASML

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

Abstract— **Process mining techniques attempt to extract non-trivial and useful information from event logs. For example, there are many process mining techniques to automatically discover a process model describing the causal dependencies between activities. Several successful case studies have been reported in literature, all demonstrating the applicability of process mining. However, these case studies refer to rather structured administrative processes. In this paper, we investigate the applicability of process mining to less structured processes. 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). This case study provides many interesting insights. On the one hand, process mining is also applicable to the less structured processes of ASML. On the other hand, the case study also shows the need for alternative mining approaches.**

I. Introduction

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. In short, it makes the wafer scanners that print the chips. These wafer scanners are used to manufacture semi-conductors (e.g., processors in devices ranging from mobile phones ad MP3 players to desktop computers). Wafer scanners are complex machines consisting of many building blocks and use a photographic process to image nanometric circuit patterns onto a silicon wafer, much like a camera prints an image on film. Because of competition and fast innovation, the time-to-market is very important. There is an ongoing effort to reduce the line widths 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 as, e.g., in [15], the subject of study is here the test process itself.

The basic idea of *process mining* is to discover, monitor and improve *real* processes (i.e., not assumed 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 other operational processes or systems can be monitored. For example, we have applied process mining to complex X-ray machines, high-end copiers, web services, careflows in hospitals, etc. 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 and can be used to (1) *discover* new models (e.g., constructing a Petri net that is able to reproduce the observed behavior), (2) check the *conformance* of a model by checking whether the modeled behavior matches the observed behavior, and (3) *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*.

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 this paper. 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. Then, we compared them to the idealized reference model. Finally, we used process mining to answer the question "Where is the most time spent in the test process?". In this paper, we show that recently developed process mining techniques can be used to answer the questions stated above. This is interesting since, so far, process mining has only been applied to rather structured processes (e.g., administrative processes supported through some information system [2]). For the case study we used our *ProM framework*[1]. 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 II 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 III. Section IV presents

A. Rozinat, C.W. Günther, and W.M.P. van der Aalst are with the Information Systems group, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands (email: a.rozinat@tue.nl; c.w.gunther@tue.nl; w.m.p.v.d.aalst@tue.nl).

I.S.M. de Jong is with ASML, P.O. Box 324, NL-5500 AH, Veldhoven, The Netherlands (email: ivo.de.jong@asml.com).

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

the results of this study, and concrete improvement actions for the ASML test process are proposed in Section V. Section VI concludes the paper.

II. Related Work

In this section, we first review related work on process mining and then review related work on test processes.

Since the mid-nineties several groups have been working on techniques for process mining [4], [5], [10], [12], [14], [25], i.e., discovering process models based on observed events. In [3] an overview is given of the early work in this domain. The idea to apply process mining in the context of workflow processes was introduced in [5]. In parallel Datta [12] looked at the discovery of business process models. Cook et al. investigated similar issues in the context of software engineering processes [10]. Herbst [19] was one of the first to tackle more complicated processes, e.g., processes containing duplicate tasks.

Most of the classical approaches have problems dealing with concurrency. The α-algorithm [4] is an example of a simple technique that takes concurrency as a starting point. However, this simple algorithm has problems dealing with complicated routing constructs and noise (like most of the other approaches described in literature). In [14] a more robust but less precise approach is presented. Heuristics [25] or genetic algorithms [13] have been proposed to deal with issues such as noise.

Process mining can be seen in the broader context of Business Process Intelligence (BPI) and Business Activity Monitoring (BAM). In [16], [24] a BPI toolset on top of HP's Process Manager is described. The BPI toolset includes a so-called "BPI Process Mining Engine". In [22] Zur Muehlen 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) [20]. 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.

After providing an overview of process mining literature, we discuss related work on the improvement of test processes. Most test process optimization techniques are currently focused on optimizing the test organization instead of the processes themselves. If the test process is optimized by technology-based optimization techniques, then these techniques are either only applicable in a single discipline or still very general. An example of detailed mono-disciplinary optimization techniques is the use of test coverage measures [21] for software testing. These detailed measures are used to determine the coverage of a set of test cases on the system under test. These methods are applied for small software systems where high quality levels are important and do not scale very well for large software systems.

More general test process optimization techniques apply risk-based test sequencing [6], [18] in combination with a certain stop criterion such that the test cases which cover the highest risk (high level test cases) are executed first. The disadvantage of this approach is that these high level test cases are often inconclusive about the root cause of a failure. A lengthy diagnosis action is required when these test cases fail. A sequencing algorithm which optimizes a test phase including diagnosis and fix actions is described in [9], [8], whereas the system test model which is used models a test problem independent of the discipline. The case study reported in this paper investigates the differences between the actual, executed test sequences and the planned test sequences.

III. Case Study

This section introduces the case study where process mining was applied to the test process of ASML's wafer scanners. After providing some background information (Section III-A), we describe the test process of a wafer scanner in more detail (Section III-B), and then look at the log data recorded during these tests (Section III-C). The event logs serve as input for our process mining techniques and the results of their analysis are described in Section IV.

A. ASML's Wafer Scanners

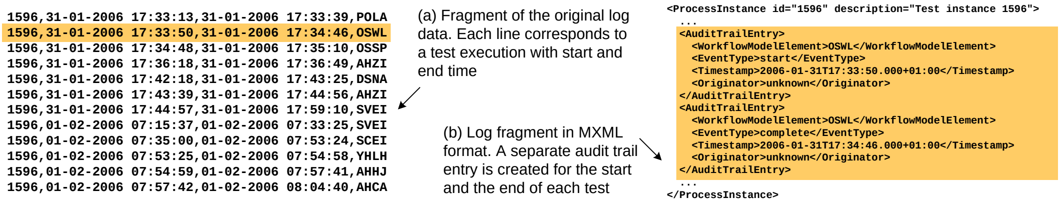
Nowadays, semi-conductors can be found in many appliances around us. Mobile phones, MP3-players, television sets and desktop computers contain processors and computer memory. These semi-conductors are manufactured in twenty-plus steps, called the semi-conductor manufacturing process. Images of a transistor pattern are placed on a silicon wafer with typical line widths of 90 nm and less. This imaging process is repeated up to 30 or more times to form a completely functional integrated circuit. The imaging of the pattern on a silicon wafer is done by a so-called wafer scanner. The semi-conductor manufacturing process is explained in detail in [11].

A wafer scanner consists of around one thousand building blocks. These building blocks can be considered a system in itself; they can consist of an entire electronics rack, thousands of lines of code, or a complete lens system. Most of these building blocks are manufactured at suppliers. Together, these building blocks form a scanner. After assembly of the building blocks, the wafer scanners are calibrated and tested. The calibration and test sequence of a wafer scanner ends with a system qualification phase. In this system qualification phase, the system performance in terms of throughput, overlay and imaging performance is measured. The throughput of a wafer scanner is a measure of the wafer production speed. The overlay of a wafer scanner is a measure of how accurate the different patterns are placed on top of each other in each next layer. The imaging performance of a wafer scanner determines the line width of the structures in the integrated circuit. The capabilities of the TWINSCAN$^{\text{TM}}$ XT:1900Gi wafer scanner used in this case study in terms of throughput, overlay and imaging performance are resp. ≥ 131 wph, ≤ 6 nm and ≤ 40 nm [7]. Moore's law[2] is forcing companies like ASML to constantly innovate and build machines that can handle silicon wafers with smaller line widths. This requires a continuous balance between performance (smaller line widths) and reliability (semi-conductors that can be produced without failures).

B. The Test Process

The whole test process 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

[2] Moore (founder of Intel), commenting on the growth of the microelectronics industry in 1964, noted a doubling of the number of elements on a produced chip once every 12 months. For a decade that meant a growth factor of approximately 1000. Today, when Moore's Law is quoted, the time constant typically quoted is 18 months.



```
1596,31-01-2006 17:33:13,31-01-2006 17:33:39,POLA
1596,31-01-2006 17:33:50,31-01-2006 17:34:46,OSWL
1596,31-01-2006 17:34:48,31-01-2006 17:35:10,OSSP
1596,31-01-2006 17:36:18,31-01-2006 17:36:49,AHZI
1596,31-01-2006 17:42:18,31-01-2006 17:43:25,DSNA
1596,31-01-2006 17:43:39,31-01-2006 17:44:56,AHZI
1596,31-01-2006 17:44:57,31-01-2006 17:59:10,SVEI
1596,01-02-2006 07:15:37,01-02-2006 07:33:25,SVEI
1596,01-02-2006 07:35:00,01-02-2006 07:53:24,SCEI
1596,01-02-2006 07:53:25,01-02-2006 07:54:58,YHLH
1596,01-02-2006 07:54:59,01-02-2006 07:57:41,AHHJ
1596,01-02-2006 07:57:42,01-02-2006 08:04:40,AHCA
```

(a) Fragment of the original log data. Each line corresponds to a test execution with start and end time

(b) Log fragment in MXML format. A separate audit trail entry is created for the start and the end of each test

```
<ProcessInstance id="1596" description="Test instance 1596">
  ...
  <AuditTrailEntry>
    <WorkflowModelElement>OSWL</WorkflowModelElement>
    <EventType>start</EventType>
    <Timestamp>2006-01-31T17:33:50.000+01:00</Timestamp>
    <Originator>unknown</Originator>
  </AuditTrailEntry>
  <AuditTrailEntry>
    <WorkflowModelElement>OSWL</WorkflowModelElement>
    <EventType>complete</EventType>
    <Timestamp>2006-01-31T17:34:46.000+01:00</Timestamp>
    <Originator>unknown</Originator>
  </AuditTrailEntry>
  ...
</ProcessInstance>
```

Fig. 1. Converting the log into the MXML format

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. And 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.

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. 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. The work presented in this paper attempts to shorten the test process by applying process mining techniques to the existing test processes, i.e., we analyze the test process based on historical data to find bottlenecks and ideas for improvement.

C. 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 1(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 tested. Afterwards the start time, the completion time, and the four-letter code for the executed test are recorded.[3]

To analyze the log data with ProM we had to convert them into the MXML[4] format. This was realized by a custom-built converter plug-in for the *ProMimport framework*[5]. ProM*import* facilitates log transformation tasks and provides converter plug-ins for a wide variety of systems to the XML format used by ProM [17]. 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 1(b) depicts the MXML log fragment for the highlighted test from Figure 1(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 which's 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).

IV. Process Mining 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 V, 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 at processes related to patient flows, insurance claims, traffic fines, etc., then there are typically

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

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

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

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 until 16250. Finally, we can see that there are 720 different 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.

As a next step, we want to make use of the *timestamps* in the log. For example, we can easily add up the times between the start and the completion of a specific test to see how much it contributes to the overall test process duration. In combination with filtering, also more complex information can be derived from the log. For example, Table I shows the top 4 idle times that accumulated after each test for machine 1596, sorted by the sum of all measured values.

TABLE I

IDLE TIMES AFTER TEST EXECUTIONS FOR MACHINE 1596. TOP 4 SORTED BY SUM: MOST OF THE IDLE TIME ACCUMULATED AFTER TEST ’PYWZ’

Test Code	*Minimum (in Minutes)*	*Arithm. Mean (in Hours)*	*Sum (in Hours)*	*#*
PYWZ	97.0333	61.3256	429.2797	7
DSNA	0.0166	1.1433	147.4886	129
IWOW	1.1166	13.8753	111.0027	8
OSHY	0.0166	2.1286	87.2730	41

Detailed information about idle times, such as in Table I, is essential for ASML to minimize the overall test duration. However, before we can extract this information from the log, we again need to apply filtering mechanisms to the initial event log. Figure 2 visualizes how the example log fragment is transformed from representing test durations to idle times after these tests. As a first step, we replace each ‘complete’ event and its succeeding ‘start’ event by a ‘start’ and ‘complete’ event marking the transition between those tests.[6] Then, we abstract from the test that was executed afterwards. This is the log that was used as input to calculate the values in Table I. Nevertheless, it can also be useful to analyze the log on level 1 (i.e., directly after the application of the inversion filter) as the succeeding test code can provide insight into the nature of the idle time.

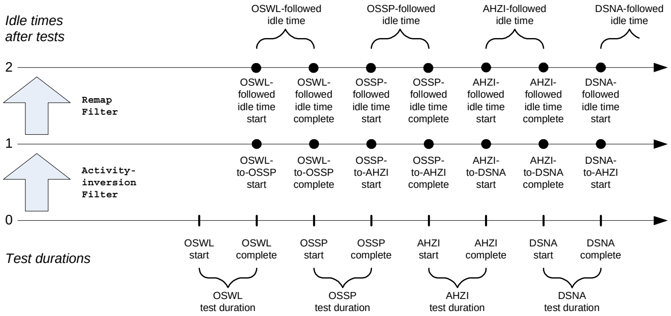


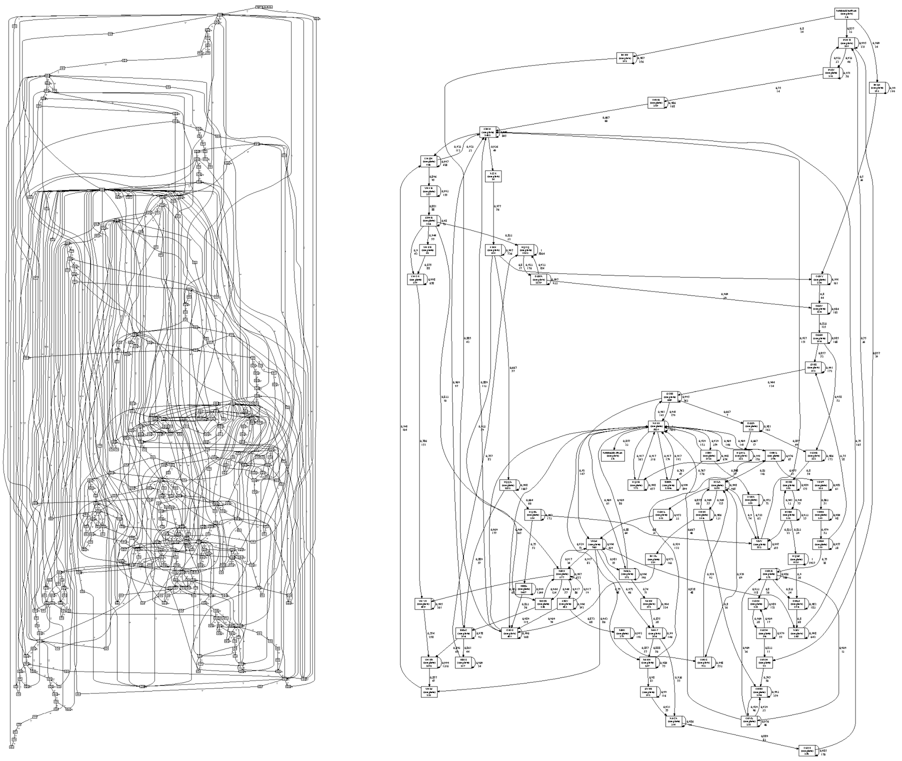
Fig. 2. An inversion filter allows to analyze the idle times instead of the actual test durations. Then we abstract from the succeeding test

We have seen that using log inspection and filtering tools we can already answer questions about general log characteristics, such as simple frequency measures, the distribution of events over time, throughput times, and basic statistics about test durations and idle times. As stated in Section III-B, the second goal for ASML—next to the minimization of idle times in the test process—is to reduce the re-execution of parts of the job-step sequence. We, therefore, want to apply now 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. However, the nature of our test log poses some challenges. On the one hand, there are only a few process instances available, which at the same time are very long, and contain logged tests that are executed in different phases of the test process (i.e., ‘multiple’ codes). On the other hand, we already know that the process is very flexible (as parts of it might need to be redone depending on the outcome of the performed tests, resulting in many variations in the test sequences), while most of the traditional discovery algorithms described in literature assume that the underlying process is “structured” (i.e., the number of possible paths through a process is limited or the paths have some regular form).

The Heuristic Miner is one of the algorithms that can deal with such less structured processes [25]. It tries to abstract from low-frequent behavior based on certain heuristics to connect the activities in the process. Figure 3(a) depicts the initial model that was discovered based on the whole log. The discovered model is “spaghetti-like”, i.e., it is huge (nodes corresponding to the 360 tests) and it is unstructured. Such spaghetti-like models are

[6] Note that this mechanism only works because we have no interleaving tests in the log. Each test that is started will first be completed before the next one is started.



(a) Discovered Process Model based on log with only 'complete' events. Contains 360 activities

(b) Discovered Process Model based on log with only 'complete' events that occurred in all the 24 cases. Contains 70 activities

Fig. 3. Process models on test-code level discovery using ProM

not caused by limitations of the algorithm but by the inherent complexity of the testing process. Therefore, we subsequently applied further filtering techniques to abstract from certain tests in the process. This helps to yield smaller models. As an example, we show the model in Figure 3(b). This model contains only 70 different activities and reflects the view on all *common* tests in the process (i.e., tests that were performed for each of the machines). To be able to compare the discovered model to the existing reference sequence, we also analyzed the process on the job-step level. Due to a lack of space, this model cannot be shown in this paper.

V. Evaluation and Improvement Suggestions

To identify concrete improvement suggestions, we evaluated the presented process mining results from an ASML perspective. First, the dominant feedback loops in the test process were analyzed (Section V-A). Feedback loops in the process indicate that a job step fails and a previous job step needs to be re-done. After the previous job step is re-done, often parts of the already executed sequence must also be re-executed. This is taking valuable time. Second, possible root causes for some of the idle times are given (Section V-B).

A. Dominant Feedback Loops

Feedback loops in the mined process indicate that a certain job step failed and caused that a job step, which was positioned earlier in the sequence, needs to be re-executed. Ideally, failures, fixes, and re-execution of test cases are performed in the job step that failed. Given technological constraints related to the construction of a wafer scanner, this is not always feasible. Moreover, many different faults can cause a particular failure. Process improvement in this area should start with the most important failures and feedback loops. The following dominant feedback loops (→) have been observed between job steps in the discovered process model: (1) z → (via d) → a, (2) z → (via d) → l, (3) t → a, and (4) v → f.

In general, this shows that the job steps 'z', 't', and 'v' are job steps which are capable of finding the errors that were missed in the previous job steps. Errors detected in these steps cause the test process to be "rolled back". Test cases that fail in these job steps should be placed in the lower level job steps as additional test case to test the overall performance. This allows an early failure and an early fix. Process improvement should focus on these four dominant feedback loops for these systems. More specifically, the first and second feedback loop visit job step 'd'. This is possibly to determine if either job step 'a' or job step 'l' needs to be executed to fix this problem. The test cases in job step 'd' that perform this diagnosis could be useful as a standard test case in job step 'a' and 'l'.

B. Idle Times during Test Procedure

Using idle time for scheduling automatic test actions is one of the possibilities to reduce the overall test duration. In Section IV we described how the idle time accumulating after a certain type of test can be determined using a combination of filtering techniques. Table I shows the analysis results for one of the 24 machines. Most of the idle time on this machine accumulated after executing the tests (1) 'PYWZ', (2) 'DSNA', and (3) 'IWOW', which are also among the 4 tests that accumulated most of the idle time for all the 24 machines together.

(1) The 'PYWZ' test is used to stabilize the wafer scanner, which takes hours. Therefore, it is started at the beginning of the weekend at the end of the 'zero' job step. The wafer scanner enters idle time if stabilization finishes earlier. This can easily be solved by adding the test cases of the next job step to this test set. For the machine in Table I, at least 97 minutes, and on average 61 hours accumulated after 7 executions of this test.

(2) The 'DSNA' test ensures the reliability of the wafer handler, one of the sub-systems in a wafer scanner. This test is executed during the available night hours throughout the entire job step sequence. Sometimes, the test queue finishes early. This can be resolved by adding additional 'DSNA' test cases to the test set. For the machine in Table I, at least 0.01 minutes, and on average 1.14 hours accumulated after 129 executions of this test.

(3) After the execution of the test 'IWOW', at least 1 minute, and on average 13 hours accumulated after 8 executions on the same machine. The test was executed twice as the last test before the weekend, and once before the Christmas holidays.

VI. Conclusion

We have demonstrated that current process mining techniques can already answer many questions, even yield concrete suggestions for process improvement also in as complex environments as the wafer stepper qualification phase of ASML. However, due to the rapid technological advancements, the analysis results presented in this paper are likely to be outdated already for the next series of wafer steppers than the ones that we analyzed. To enable a *continuous improvement* of the test process in ASML, process analysis should be best carried out in an iterative manner.

Additional information sources that could be used to improve the data set are available, but were not used in this case study. For example, SAP data is available with (manually entered) job step start and stop data, and the start and stop moment of the entire test sequence. More importantly, however, further research is needed to develop process mining techniques that are particularly suitable for analyzing less structured processes like the highly dynamic test process of ASML. We found similar logs in other domains, such as health-care processes. In these situations, the resulting models are often overly complex and confusing (i.e., "spaghetti-like"), which makes them hard to extract useful information from.

Acknowledgements

This research is supported by the Technology Foundation STW, EIT, and the Tangram and IOP programs of the Dutch Ministry of Economic Affairs.

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, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business Process Mining: An Industrial Application. *Information Systems*, 32(5):713–732, 2007.

[3] 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.

[4] 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.

[5] 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.

[6] S. Amland. Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study. *Journal of Systems Software*, 53(3):287–295, 2000.

[7] ASML. Website ASML NV, Veldhoven, www.asml.com, 2007.

[8] R. Boumen, I.S.M. de Jong, J.W.H. Vermunt, J.M. van de Mortel-Fronczak, and J.E. Rooda. A risk-based stopping criterion for test sequencing. Internal Report SE 420460, Eindhoven University of Technology, January 2006. Submitted to IEEE Transactions on Systems,Man,and Cybernetics-Part A: Systems and Humans.

[9] 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. *Accepted for IEEE Transactions on Systems,Man,and Cybernetics-Part A: Systems and Humans*, 2006.

[10] J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.

[11] Chipworks corporation. Advanced semiconductor manufacturing handbook. Technical report, Chipworks corporation, Januari 2000.

[12] A. Datta. Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches. *Information Systems Research*, 9(3):275–301, 1998.

[13] A.K. Alves de Medeiros. *Genetic Process Mining*. PhD thesis, Department of Technology Management, Technical University Eindhoven, 2006.

[14] B.F. van Dongen and W.M.P. van der Aalst. Multi-Phase Process Mining: Building Instance Graphs. In P. Atzeni, W. Chu, H. Lu, S. Zhou, and T.W. Ling, editors, *International Conference on Conceptual Modeling (ER 2004)*, volume 3288 of *Lecture Notes in Computer Science*, pages 362–376. Springer-Verlag, Berlin, 2004.

[15] S.M. El-Shal and A.S. Morris. A fuzzy expert system for fault detection in statistical processcontrol of industrial processes. *IEEE Transactions on Systems, Man, and Cybernetics–Part C*, 30(2):281–289, 2000.

[16] 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.

[17] 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.

[18] M.J. Harrold, D. Rosenblum, G. Rothermel, and E. Weyuker. Empirical studies of a prediction model for regression test selection. *IEEE Transactions on software engineering*, 27(3):248–263, 2001.

[19] J. Herbst. A Machine Learning Approach to Workflow Management. In *11th European Conference on Machine Learning*, volume 1810 of *Lecture Notes in Computer Science*, pages 183–194. Springer-Verlag, Berlin, 2000.

[20] 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.

[21] British Computer Society Special Interest Group in Software Testing. *Standard for Software Component Testing*. British Computer Society, 2001.

[22] 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.

[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] 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.

[25] A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.