Production Process Monitoring Using Model-Driven Event Processing Networks

Falko Koetter(✉) and Tobias Krause

University of Stuttgart IAT, Nobelstr. 12, 70569 Stuttgart, Germany
{falko.koetter,tobias.krause}@iao.fraunhofer.de

Abstract. Economic realities make flexibility in production processes a necessity. Small batch production necessitates reuse of machines within different production processes. Monitoring in such production environments must adapt to process changes without impacting production machines and software. In this work we propose a novel method for production monitoring using event processing networks, separating machine and production processes, thus increasing flexibility and minimizing configuration efforts.

Keywords: Business process management · Process monitoring · Complex event processing · Event processing networks · Model-driven architecture

1 Introduction

The term *Industry 4.0* describes the increasing use of information technology, networking, smart objects, services and data in manufacturing [20]. Akin to the trend of business process management, production processes become IT-supported. In a production process, products are manufactured according to given orders using production resources and materials [5].

One part of Industry 4.0 is the monitoring and optimization of production processes with respect to cost, quality and time [14]. Production processes are stricter organized in single activities, while business processes provide more degrees of freedom during execution. The reason for this is that machines used in production need to be fine-tuned to work seamlessly together and provide a high throughput and quality, especially in assembly-line production. However, due to trends like just in time production and mass customization, production in smaller batches becomes a necessity and thus, more flexibility in production processes is needed [20, p. 90].

According to [20, p. 93] the majority of manufacturing companies note a lack of accurate and up-to-date process information, which makes manual intervention in production processes necessary. However, 72 % of companies estimate a high or very high potential to avoid these interventions if current process information is available via monitoring.

M.A. Jeusfeld and K. Karlapalem (Eds.): ER 2015 Workshops, LNCS 9382, pp. 89–98, 2015.
DOI: 10.1007/978-3-319-25747-1_9

In this work we describe a novel approach to monitoring of small-batch production processes. Extending an approach for model-driven process monitoring, we present a method to model, deploy and link multiple process monitoring instances in an event processing network (EPN). We apply this method to small-batch production processes, separating the machine, machine process and production process levels to facilitate easy rearrangement of machines for monitoring different batches with batch specific processes, goals and fault tolerances.

The remainder of this work is structured as follows. Section 2 gives an overview of related work. Section 3 describes the example scenario. Section 4 gives an overview of the model-driven process monitoring approach used in this work. Section 5 describes how EPNs can be utilized to monitor and reconfigure production processes. In Sect. 6 the prototype and evaluation are described. Finally, Sect. 7 gives a conclusion and outlines future work.

2 Related Work

In this section we give an overview of relevant work in the fields of production monitoring, distributed process monitoring and EPNs.

Monitoring of production processes measures Key Performance Indicators (KPI) like timeliness, workload, machine productivity, availability, reaction time, inventory and stock levels as well as product and process quality [8, p. 13ff]. When monitoring production processes, existing approaches differ in their scope, purpose and position within the process. To monitor single machines, so-called machine-integrated monitoring systems are used. An example is the monitoring and enforcement of thresholds to prevent damage to parts currently being worked on [23, p. 331ff]. Machine-integrated monitoring systems can use a wide array of sensors to measure mechanical, electrical, thermal, magnetic or chemical parameters [9, p. 387]. If the monitoring tool is integrated in the control, existing values of the controls like position of spindles and speed can be monitored, for example to prevent self-damaging of machines [23, p. 376ff]. On a larger scope, Manufacturing Execution Systems (MES) connect business management and production processes [8, p. 7]. MES provide overarching functionality, e.g. product tracking, monitoring data gathering, analysis and process monitoring [13]. MES are used in assembly line and mass production, but lack the flexibility to be used for small batch production [19, p. 157].

Niche solutions exist for specialized processes. For example, [18] describes a monitoring software for production logistics processes. Maintenance of machines can be costly if it leads to extended downtime, which makes monitoring and proactively detecting maintenance important [22]. Maintaining machines before an outage would occur is called *predictive maintenance* [17, p. 4-6]. Monitoring solutions to predict necessary maintenance can take into account cost and other factors [4]. Overall, the state of the art in production monitoring shows sophisticated solutions at the machine level and for large-scale processes. However, these solutions lack the flexibility needed in small batch production.

Complex Event Processing (CEP) is a technology for processing large amounts of events in near real-time [15]. For example, the CEP Engine Esper

uses an event processing language (EPL) to filter, aggregate, change and generate streams of events [1]. An *event processing network* (EPN) consists of multiple so-called *event processing agents* (EPA) processing events in unison, consuming and producing them [15]. Such an EPN can be used for distributed event processing, to handle large amounts of events and to separate concerns and abstraction levels [16]. [16] shows how CEP can be used to monitor production processes. [2] shows a case study of using CEP to control and monitor a modular manufacturing line. However, the rearrangement for different batches or products is not tackled in both approaches. [21] shows an approach to connect a process engine with a CEP engine to monitor and adapt business processes using graphically modeled EPL rules. This approach however is limited to a single process engine and allows reuse only of EPL rules, not EPAs. [6] presents an approach to use event data for process control in a process engine. In comparison to our approach, a high degree of manual implementation is necessary.

3 Motivational Example

Consider a metal-working company producing parts which are used in other products, for example in the IT, automotive and furniture industries. Due to raising demands in flexibility [20, p. 90], most parts are produced in small batches using multiple machines as part of an individualized process for each part. The CNC (Computerized Mechanical Control) machines [12] are programmable to produce the desired part. Programs can be exchanged relatively quickly, but the overall process involving multiple machines is more complex to change. For a simplified process, we introduce three machines.

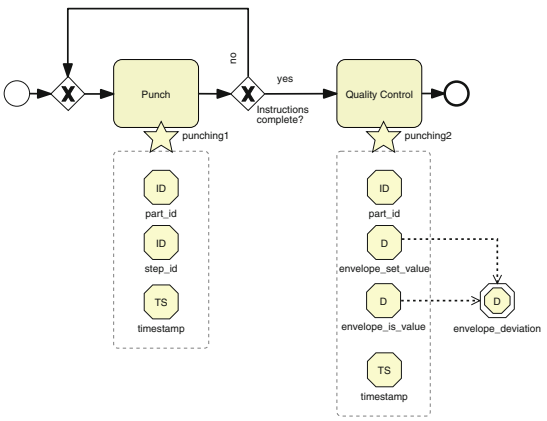

Fig. 1. Example process and goal model for punching machine.

Figure 1 shows the process model of the punching machine. Holes are punched into a metal part until all punching instructions are executed. Afterwards, a

machine-integrated quality control step takes place, comparing the shape of the part to a set reference using the envelope method [11]. Attached to the process model is a *goal model* in the ProgoalML notation. Measuring points (e.g. *punching1*) are attached to both activities, indicating that a measurement of monitoring data should take place. Inside the measuring points, the parameters to be measured are given. For example, in *punching2* attached to *Quality Control* the id of the current part, the timestamp of the measurement and the expected (set) and actual (is) value of the metal shape from the envelope method are measured. From both values, a KPI is calculated, indicating the deviation in percent according to the envelope deviation. Note that no concrete goal regarding deviation is given in the machine process. KPIs in ProgoalML are defined using a formula editor, providing mathematical operations, aggregations, etc. [10].

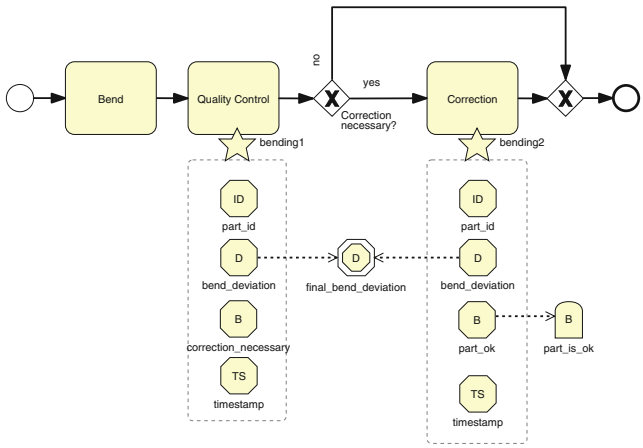

Fig. 2. Example process and goal model for bending machine.

Figure 2 shows the process and goal model of the bending machine. A flat metal part is bent according to specifications to attain a desired three-dimensional shape. After bending quality control takes place, the deviation of the bent part from the ideal shape is measured. If necessary, a correction step is used to fix minor deviations in the bent shape [3]. In the goal model, a KPI measures the final bend deviation after correction and the goal *part_is_ok* is fulfilled if the part is without deviation after the correction step.

Figure 3 shows the process and goal model of the painting machine. A metal part is painted by dipping or spraying paint, after which the part needs to dry before the paint job is considered finished. The goal model only monitors paint level and stipulates that the paint level must be sufficient (i.e. above 5 %).

Using these three machines, a variety of metal parts can be created. Consider for example a metal part of a head gasket in a car. This part needs to be punched and bent precisely, as it is critical to sealing the cylinders. Not considering rust

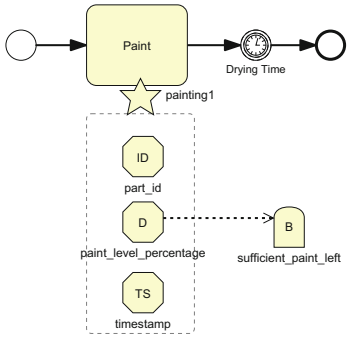

Fig. 3. Example process and goal model for painting machine.

protection, the part does not need to be painted, as it is not visible from the outside. In comparison, consider a metal part that is going to be the faceplate for an electronic device. This faceplate needs to be painted, as it is visible to the customer. The deviation in punching and bending is comparably relaxed. Both parts can be created with the three machines, but the overall processes have different steps and quality criteria.

4 Model-Driven Process Monitoring

In previous work we introduced aPro, a model-driven architecture for business process monitoring [10]. Figure 4 gives an overview of the most important components of the approach.

To monitor a process, first, a process model is graphically modeled and augmented with process goals, metrics and KPIs as shown in the motivational example. The process and monitoring model are stored in a *ProGoalML* file. This file is used as basis for automatic model transformation, which serves to create technical components from the conceptual model. A monitoring container is created, containing a dashboard and CEP engine, both configured with event processing rules detecting process instances from measurement patterns and calculating KPIs and goals as well as a dashboard configuration (*VisML* [7]). This monitoring container receives monitoring data via monitoring web services. To support a wide array of executing systems, ranging from process engines to legacy applications to machines, monitoring stubs are used. Monitoring stubs are integrated with the executing systems and send monitoring data to monitoring web services. To achieve this, events can be sent from the application, triggered to be sent by an application, sent by an external component monitoring the executing systems status etc. Some monitoring stubs can be automatically generated (e.g. shell scripts and Java classes), others need to be manually implemented. Note that monitoring stub integration is the only part of aPro which is not fully automated.

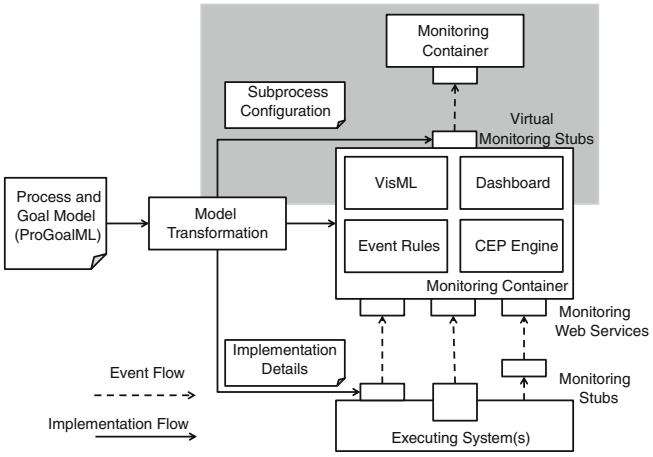

Fig. 4. Overview of model-driven process monitoring (focus of this work highlighted in grey)

During operation, events are generated by the monitoring stubs and sent to the monitoring container. The CEP engine processes these events with rules generated according to the conceptual model, assembling process instances from single events, calculating KPIs and goals and aggregating data. The results are visualized in a dashboard. While this model-driven approach allows to setup a process monitoring infrastructure without component configuration supporting arbitrary systems executing the process, the degree of abstraction is limited. aPro separates the conceptual monitoring model from the implementation in event processing rules. However, all monitoring events are processed and aggregated in the same CEP engine, regardless of their role within the monitored process, making a new model transformation and redeployment of the monitoring container necessary, whenever a change in the monitoring model is made.

5 Event Processing Networks

Considering the motivational example, the three machine processes are subprocesses of the overall production processes. To allow monitoring of these overarching processes, either the machine process monitoring has to be remodeled on the production process level or it has to be reused. While reusing measuring points via copy and paste recreates the behaviour on a model level, the monitoring stubs of the machine processes still need to be adapted during model transformation.

We propose using an EPN to facilitate hierarchical monitoring of machine level subprocesses and production level main processes. To enable the building of model-driven EPNs, a relationship between different monitoring containers acting as event processing agents need to be defined. Expanding on the monitoring stub concept, we propose using a monitoring container itself as a monitoring

stub for another, higher-level monitoring container. Thus, a subset of result data can be reused as a complex event in a downstream monitoring container.

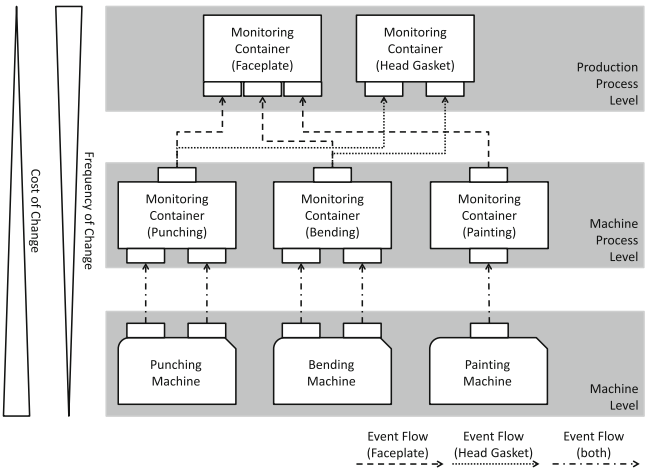

Fig. 5. Event processing network of production process scenario

Figure 5 shows the proposed concept applied to the production process scenario. On the *machine level*, production machines are instrumented with monitoring stubs gathering data as indicated in Sect. 3. On the *machine process level*, for each machine, a monitoring container receives the gathered data and processes it according to the goal model of the respective machine process. These monitoring containers monitor the machine processes, calculating KPIs and goals for each instance. For each process instance, they send a *complex event* to the monitoring container on the *production process level*, which monitors the production process. As indicated in Fig. 5, two configurations of the EPN are possible, depending on which production process is currently running.

Machine process level monitoring containers serve as virtual monitoring stubs for both production processes. Consider the production process for the faceplate in Fig. 6. This process uses all three machines to produce faceplates. Quality criteria specific to the use case have to be met by a part, otherwise it is scrapped before painting. These criteria are determined using data from the machine processes. Also, both completed and scrapped parts per hour are measured.

Three of the four measuring points of the faceplate production process represent the machine processes. In order to measure at these measuring points, the monitoring containers of the machine processes need to act as virtual monitoring stubs. As shown in the highlighted part of Fig. 4, a *subprocess configuration* is used to determine the virtual monitoring stubs. This configuration indicates which virtual monitoring stubs are to be created and maps data of the monitored process instance to the input of the monitoring stubs.

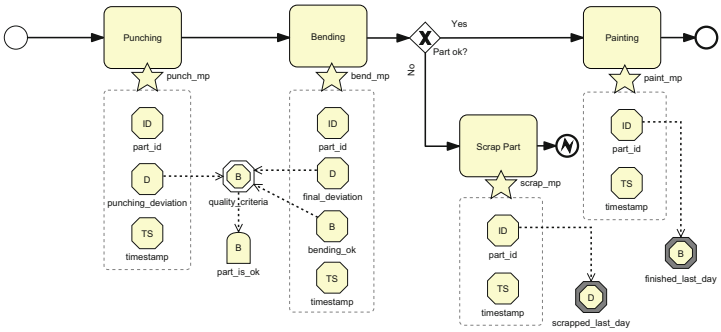

Fig. 6. Process and goal model for faceplate production

```
<forwarding>
  <serverUrl>http://localhost:8080/Faceplate/</serverUrl>
  <measuringPoint>punch_mp</measuringPoint>
  <mappings>
    <mapping>
      <element>envelope_deviation</element>
      <parameter>punching_deviation</parameter>
    </mapping>
    <mapping>
      <element>part_id</element>
     <parameter>part_id</parameter>
    </mapping>
  </mappings>
</forwarding>
```

An example subprocess configuration is listed above, creating a virtual monitoring stub for the measuring point *punch_mp* withing the punching process. *serverUrl* and *measuringPoint* elements are used to address the monitoring web service. *Mappings* determine which *elements* of the punching process goal model are mapped to which *parameters* of the *punching_mp* measuring point. Note that multiple virtual monitoring stubs can be contained within the configuration.

Using subprocess mappings, the measuring points of the faceplate process can be provided with events. The fourth measuring point, *scrap_mp* corresponds to no machine and needs to be triggered whenever a part is scrapped. This can be done automatically or manually using an automatically generated webform monitoring stub [10].

If instead of faceplates different parts shall be produced, another production process has to be used, for which individual steps, quality criteria, KPIs etc. can be modeled. Using the subprocess mappings, the machine processes and monitoring stubs can be reused without change. Thus, the production process level is flexible while avoiding unnecessary, costly changes on the machine level.

6 Prototype and Evaluation

We implemented subprocess mapping and virtual monitoring stubs within the aPro prototype described in [10]. The prototype provides web-based modeling of process and goal models as well as model transformation and monitoring container deployment. We used the prototype to model all five processes, deploy them and create the EPN between monitoring containers.

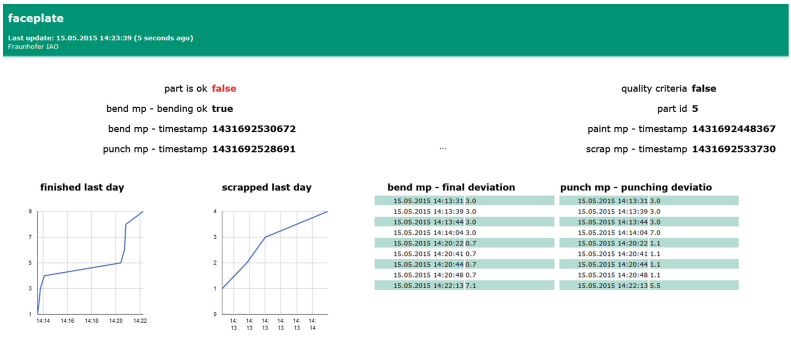

Fig. 7. Live dashboard of faceplate process with synthetic data

Figure 7 shows the generated dashboard of the faceplate production process. We evaluated the EPN using synthetic test data generators posing as monitoring stubs for the machine processes. We found the EPN to work as described, passing data from machine process monitoring containers to production monitoring containers. Switching production processes was achieved by changing the subprocess configuration files. While proving the feasibility of the concept, we have not yet evaluated the prototype within a real production process environment. We are currently searching for industry partners to perform such trials.

7 Conclusion and Outlook

In this work we described a method for production process monitoring using a model-driven monitoring architecture and connecting multiple monitoring containers to form an event processing network. This approach effectively separates sub and main processes and makes reuse and reconfiguration feasible, minimizing the need for change on lower levels when high level processes change. We evaluated this approach with a prototype using the example scenarios.

In future work, we would like to apply the work in a real-life production environment. Additionally, we would like to further investigate the formation of EPNs between different processes, in particular more user-friendly ways to visualize and change an EPN.

References

1. Esper EPL Reference. http://esper.codehaus.org/esper-4.11.0/doc/reference/en-US/html/index.html
2. Ahmad, W.: Formal modelling of complex event processing and its application to a manufacturing line (2012)
3. Boettger, U.: Messtechnik mit kurzem draht zur biegemaschine. Industrieanzeiger. http://www.industrieanzeiger.de/home/-/article/12503/28869833/Messtechnik-mit-kurzem-Draht-zur-Biegemaschine/
4. Denkena, B., Bluemel, P., Kroening, S., Roebbing, J.: Condition based maintenance planning of highly productive machine tools. Prod. Eng. **6**(3), 277–285 (2012)
5. Ingenieure, V.D.: Vdi-richtlinie: Vdi 5600 blatt 1 fertigungsmanagementsysteme (2013)
6. Janiesch, C., Matzner, M., Mller, O.: Beyond process monitoring: a proof-of-concept of event-driven business activity management. Bus. Process Manage. J. **18**(4), 625–643 (2012)
7. Kintz, M.: A semantic dashboard description language for a process-oriented dashboard design methodology. In: 2nd MODIQUITOUS 2012 (2012)
8. Kletti, J.: Manufacturing Execution Systems-MES. Springer, Berlin (2007)
9. Klocke, F., König, W.: Fertigungsverfahren 1: Drehen, Fräsen, Bohren, vol. 1. Springer-Verlag, Heidelberg (2008)
10. Koetter, F., Kochanowski, M.: A model-driven approach for event-based business process monitoring. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 378–389. Springer, Heidelberg (2013)
11. Kopka, T., Schwer, A., Faulhaber, W.: Sensoren sichern die stabilitaet im stanzprozess. BLECH InForm **5**, 48–51 (2004)
12. Koren, Y.: Computer Control of Manufacturing Systems. McGraw-Hill, New York (1983)
13. Louis, P.: Manufacturing Execution Systems. Springer, Berlin (2008)
14. Lucke, D.M.: Ad hoc informationsbeschaffung unter einsatz kontextbezogener systeme in der variantenreichen serienfertigung (2014)
15. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Boston (2001)
16. Luckham, D.C., Frasca, B.: Complex event processing in distributed systems. Technical report CSL-TR-98-754. Stanford University (1998)
17. Mobley, R.K.: An Introduction to Predictive Maintenance. Butterworth-Heinemann, Boston (2002)
18. Münzberg, B., Schmidt, M., Beck, S., Nyhuis, P.: Model based logistic monitoring for supply and assembly processes. Prod. Eng. **6**(4–5), 449–458 (2012)
19. Pfeifer, T., Schmitt, R.: Autonome Produktionszellen: Komplexe Produktionsprozesse Flexibel Automatisieren. VDI-Buch. Springer, Heidelberg (2006)
20. Spath, D., Ganschar, O., Gerlach, S., Hämmerle, M., Krause, T., Schlund, S.: Produktionsarbeit der Zukunft-Industrie 4.0. Fraunhofer Verlag (2013)
21. Vidačković, K., Weiner, N., Kett, H., Renner, T.: Towards business-oriented monitoring and adaptation of distributed service-based applications from a process owner's viewpoint. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 385–394. Springer, Heidelberg (2010)
22. Weck, M., Brecher, C.: Werkzeugmaschinen: Maschinenarten und Anwendungsbereiche. VDI-Buch. Springer, Heidelberg (2005)
23. Weck, M., Brecher, C.: Prozessüberwachung, prozessregelung, diagnose und instandhaltungsmaßnahmen. Werkzeugmaschinen 3: Mechatronische Systeme, Vorschubantriebe, Prozessdiagnose, pp. 267–404 (2006)