CONCEPTUAL MODELING OF COMPLEX EVENTS IN INDUSTRIAL BUSINESS PROCESSES[8]

D. Mayer, J. Schimmelpfennig, C. Seel & P. Walter

IDS Scheer AG Research, Saarbrücken, Germany

E-mail: {dirk.mayer, jens.schimmelpfennig, christian.seel, philipp.walter} @ids-scheer.com

Abstract:

RFID technology and the Internet of Things provide a flood of information which can be regarded as a stream of simple events, e.g. when material passes an RFID gate. Yet the large amount of events that emerge in daily business cannot be handled manually. That's why Complex Event Processing (CEP) engines have been developed to analyze large streams of simple events and aggregate them to business relevant complex events. To describe the aggregation rules, these engines use technical and formal languages which are difficult to handle for business experts. Hence, business experts are often not properly involved into the formulation of complex events. Therefore this paper proposes a conceptual modeling language that allows business experts to formulate complex events according to business needs. The proposed modeling language is demonstrated in a quality management scenario in the automotive manufacturing industry and is defined with OMG's MOF standard.

Keywords: Business Process Management (BPM), Complex Event Processing (CEP), RFID, Business-IT-Alignment

1. INTRODUCTION

Objects in the Internet of Things (IoT) provide a large amount of data. For example in a manufacturing scenario assembly components can provide information about their state via RFID tags. Besides a unique identification for each component further data can be provided, such as date of production, weight, specification, region code, etc. Based on these data, events can be monitored, e.g. if a component has been removed from stock or its weight differs from the expected value, which could indicate a lack of quality. But these isolated pieces of information about one component are usually not relevant for business decisions, unless they are aggregated and put into a larger context. The aggregation of single events to complex and business relevant events is usually done by Complex Event Processing (CEP) engines [1], [2]. These CEP engines allow aggregating several basic events to complex events and provide a history of both types of events. Therefore complex events can be generated that trigger specific business processes. For example the complex event "5 % of the components delivered by one supplier within one month have a different region code than expected" can trigger a business process for changing the supplier. Despite their relevance for business processes complex events are commonly represented by query languages or XPath expressions [2], [3].

As the formulation of complex events requires business knowledge in the regarded domain, business users have to be involved in the definition of complex events. But formalized technical expressions that are needed for CEP engines are usually not applicable for domain experts, such as the stock manager in a manufacturing company. Therefore we propose a semi-formal and graphical modeling language that allows business users without deep

[8] This paper is based on our work in the research project ADiWa, which is funded by the German Ministry for Education and Research (BMBF) under funding reference number 01IA08006.

technical knowledge to formulate complex events by using a conceptual modeling language. The modeling language that is elaborated in this paper bridges the gap between Business and IT view and makes IoT accessible to business users and process owners via complex event processing. This language allows using the data provided by IoT objects in order to describe complex events and the aggregation of basic and complex events to (higher) complex events in any number of aggregation levels. Furthermore, complex events are related to business process modeling in order to describe which business processes they trigger or influence.

2. THEORETICAL FOUNDATIONS

Methodology Used

In an endeavor to contribute to the field of information system (IS) research, this paper follows the seven guidelines for design science in IS research by Hevner et al. [4]. Especially the guidelines of relevance and of the creation of an artifact as result of the design science process have been followed. The relevance of the research topic was derived from requirements the authors were confronted with during a research project funded by the German Ministry of Research and Education and in several industry consulting projects. The resulting artifact of the research process is a modeling tool and its underlying concept. Moreover, Weick's sense making paradigm [5] is taken into account which ensures the consistency of the developed modeling language and its integration into business process modeling approaches.

Terminology and Related Work

Before our complex event modeling concept is presented, the underlying terminology is briefly clarified in this section. However, tracing former and current terminological discussions would go beyond the scope of this article, so we confine ourselves to the characterization of the terms "event", "complex event" and "complex event processing".

An event is considered a data representation of a real-world happening. Thus it is associated with that point in time when the underlying real-world incident happened.[9] Which data is stored to represent this incident as an event has to be decided depending on the scenario. Normally event modeling aims for lightweight events that can be quickly transferred and processed in large amounts. Events are immutable, i.e. they should never be altered once they are created. Events are temporary, i.e. they are rather intended for real-time or short-term processing instead of long-term storage and ex-post evaluation. They are neither intended to replace the operational control flow (which remains subject of established technologies like EAI), nor as a means of point-to-point communication. Normally they are generated, broadcasted and processed by a large number of application systems simultaneously, thus forming a so-called event cloud comprising thousands of events.

Although generating events is a simple and fast procedure for every single system participating in the event cloud, they allow to reconstruct a situational "big picture" across an arbitrary number of systems by identifying sets of interdependent events. If a situation arises that is identified by a characteristic sequence of events, a new event is generated which announces this situation. Since the event is a summary of many "simple" events, it is called a complex event. The process of aggregating a set of events according to a predefined rule describing such a situation is called complex event processing (CEP).

[9] Some approaches even associate an event with an extended time span instead of a singular point in time.

A popular example for application of CEP is credit card fraud detection [6]. It is presupposed that for every credit card transaction, the banking IT generates an event including the amount, the beneficiary, the card number and the location of card usage. A specialized monitoring system then scans this “event noise” for patterns that indicate credit card fraud: e.g. when a card is used in New York at 3 PM and in Paris at 4 PM, this strongly indicates fraud since no one can travel that fast from New York to Paris.

From a general implementation point of view, there is a multitude of standards and technologies available for developing CEP solutions. Open source and commercial frameworks, emerging and mature solutions, simple and complex packages – even a concise review of the available software would go beyond the scope of this article. Therefore we do not presume a specific CEP technology to build our conceptual modeling approach upon. Instead, the proposed modeling method aims at abstracting from the underlying CEP technology. Of course this implies that the expressiveness of the conceptual layer must be adaptable to the technical realization – however, this is beyond the scope of this article and will be treated in a later publication.

3. MODELING OF COMPLEX EVENTS

Application Scenario

To outline the benefits and to exemplify the usage of conceptual modeling of complex events we chose an assembly line production process in the automotive sector. Within the assembly line each future motor vehicle is passing through different stations where delivered preassembled components are installed. According to lean and just-in-time production these components are delivered exactly at the right time and in the right amount to the station where they are assembled. With each incoming vehicle, the assembly worker at a station is instructed which parts to assemble. The parts are identified by RFID tags to ensure that e. g. the correct variant for a certain vehicle is assembled.

Today following Kaizen [7] and total quality management principles each employee working on the assembly line can stop the line by signaling a problem – we will furthermore call this an “incident report”. A responsible head to the station actuates the stop, checks and analyzes the problem and resumes production after the problem is solved. Beyond that, there is an additional final inspection of each finished vehicle which may also reveal formerly unnoticed quality defects.

This procedure is considered an effective and efficient reaction to emergent problems as they are handled “right at the source” whenever possible. However, subtle increases of the incident and/or quality defect frequency often remain unnoticed for some time, although they may indicate a larger underlying problem, e.g. a defective lot of parts. Hence it would be very valuable to detect such problems at an early stage. Unfortunately recognizing such problems requires complex considerations of a lot of factors and thus could not be modeled in an intuitive manner.

Within the next sections, we describe a modeling concept to deal with such problems using the example given above. First we exemplify the conceptual modeling layer and then we shortly outline the technical realization.

Conceptual Modeling

In current business process management approaches, processes are documented using semi-formal modeling methods. The Event-driven Process Chain (EPC) [8] is a widely accepted standard in process modeling. Here, business relevant events and corresponding business functions are put into sequence in order to define the behavior of a business process. A process model thus defines a sequence of actions which are triggered by certain events and which produce certain events. However, this linear control flow with its strong emphasis on business functions makes it difficult to express complex situations like the slowly increasing incident rate exemplified above. It is not necessarily impossible to express such conditions in an EPC, but the model complexity would be immense. Therefore we propose to introduce a new modeling language called Event Structure Model (ESM) which supplements the EPC. It allows for defining situations by specifying certain event constellations and how to react to such situations by generating new events. Thus an ESM model can connect several EPCs by collecting relevant events from different EPCs and by producing new events triggering other EPCs.

In order to describe ESM we use OMG's Meta Object Facility (MOF) [9]. MOF uses UML class diagrams that can be enhanced with OCL statements. Figure 1 shows the meta model of the ESM language.

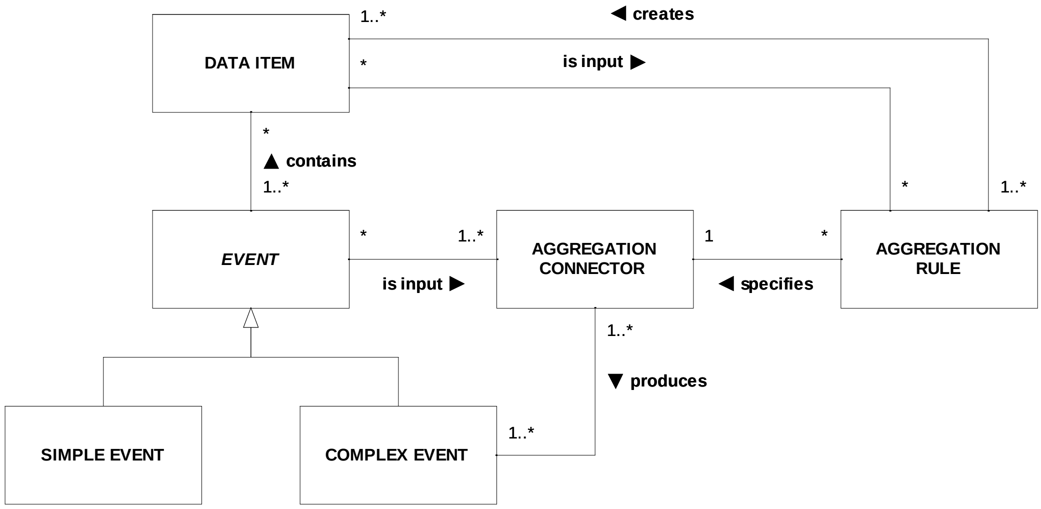

Figure 1: Meta model of the Event Structure Model (ESM)

The entry point the of the meta model is the abstract class "Event". Events can be divided in simple events and complex events. Complex events can be constructed from an arbitrary number of simple or other complex events. The aggregation connector creates the relationship between complex events and events that it is deduced from. The way how events are aggregated to a complex event is described as an aggregation rule, which is attached to each aggregation connector. The aggregation rule is based on data items that are provided by an event. The data items are used in the aggregation rule in two ways. First, they are used in order to specify whether the complex event occurs. Second, data items of the complex event can be created based on data items of its ingoing events. The graphical notation is presented in the scenario below.

To visualize the functionality of ESM, the application scenario introduced above is further detailed by an exemplary ESM. On the left-hand side of figure 2, a simplified process model of the application scenario is depicted as an EPC. For each vehicle to be assembled, parts are installed at different assembly stations. Once the installation of a specific part succeeded, the vehicle is transported to the next station until the end of production line is reached. A final quality check is conducted, ensuring the overall quality of the product.

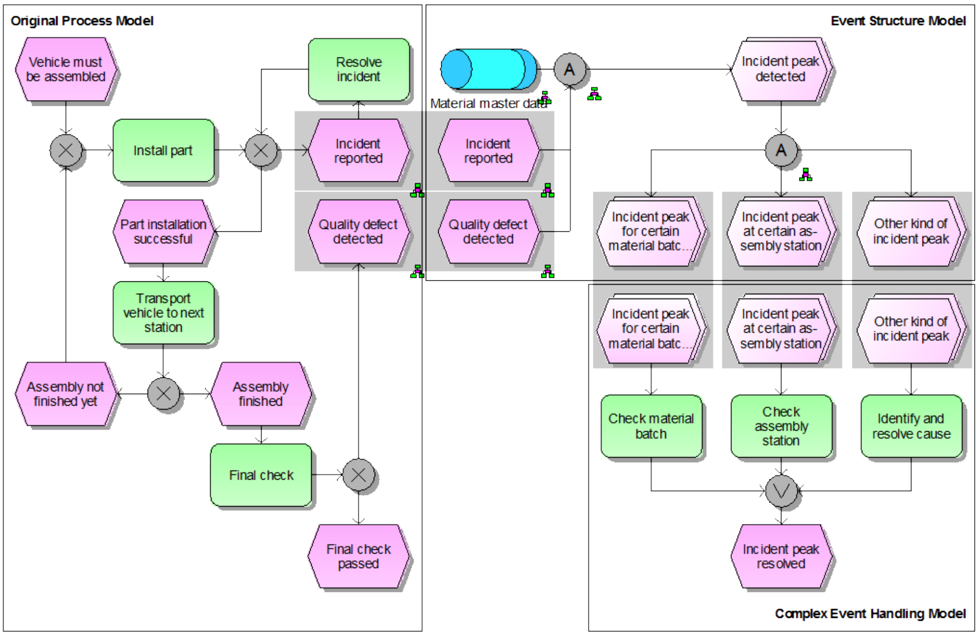

Figure 2: An Event Structure Model (ESM, top right) mediates between the standard production process (left) and the separate process for handling exceptional incident peaks (bottom right)

Incidents in the assembly process cause the assembly worker to stop the production line in order to resolve the issue. At the point of detection, the main priority is to resolve the error as fast as possible to minimize the delay in production, providing very little room for thorough root cause analysis and correlation of incident reports. Similarly, if a defective part is identified in the final quality check, it is communicated by the event “quality defect detected” and repaired with high priority in order to deliver the final product.

Since there may be multiple lines of production within a factory, peaks within the overall number of similar incidents may not be obvious, especially if the incidents happen during different working shifts or at different stations. However, the identification of peaks in related incidents (e.g. caused by defective parts of the same lot) is vital in order to optimize the production process and to prevent incidents from reoccurring due to the same cause. By correlating all incident events occurring in the production process, the complex event “incident peak occurred” can be refined.

In our approach, the structural aggregation of the complex event is specified as an ESM. As depicted in the top right corner of figure 2, complex events are aggregated by connecting member events from process models as well as external information carriers using a complex event aggregation connector (“A”). Each of the ingoing objects is typed by an assigned data type model, representing the “payload” that is communicated by those events or information carriers. The aggregated complex event itself can be used to define further business relevant complex events. In the application scenario, the two events “incident reported” and “quality defect detected” as well as the part’s material master data are used to form the complex event “incident peak occurred”. The latter is used to deduce three business relevant events which serve as trigger for optimization processes as shown in the lower right corner of figure 2.

Once the structural aggregation is defined, a mapping of ingoing event data to the complex event’s data structure is to be specified. Within that mapping, arithmetic and logical operators are available in order to define the desired payload of the resulting complex event.

By defining complex events in the ESM using simple events from process models, the conceptual aggregation aspects are covered. For the deployment of complex event definitions, technical aspects such as data mapping and filtering of data streams need to be covered. These technical modeling aspects are assigned to the complex event aggregation connectors and are described in detail in the next section.

Technical Modeling

When creating ESMs it must be taken into account that the majority of business relevant events defined in EPC models are of complex nature (such as “goods received”) and are not directly measurable in a company’s runtime environment. From a business user’s view, the information provided is sufficient in terms of business process management; however, such “narrative” declarations lack execution-relevant information on the actual process implementation within the company.

To allow for an implementation of the ESM described above it is therefore necessary to amend it with technical details that specify the event aggregation operationally. For that purpose technical models are added to the conceptual model. Actually every A-connector in the conceptual ESM is assigned with such a technical aggregation rule. As an example, figure 3 shows the technical model for the upper left A-connector in the ESM in figure 2.

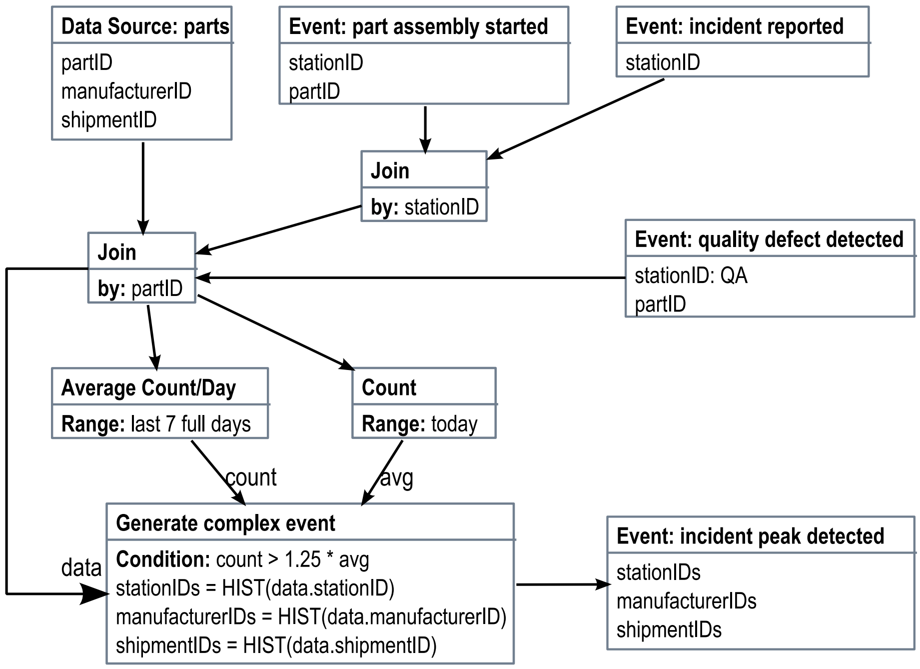

Figure 3: Technical CEP model describing the generation of the complex event “Incident Peak detected”

The technical model is laid out from top to bottom. At the top there are the information sources, i.e. the material database and three types of events. They are used to construct a single event stream consisting of incident and quality defect reports exclusively. Construction starts with an “incident report” event only carrying the number of the station reporting the incident. However, this information is too scarce to analyze the cause of a peak later, so it is combined with the previous event “part assembly started“. Each part is equipped with an RFID tag and whenever an employee starts assembling a part, he first scans its tag. Every such RFID scan generates a single event “part assembly started” which carries the scanned part number. If an incident is reported afterwards, this part is assumed to cause the incident and its number is added to the incident report event by joining it with the previous assembly start event. In the conceptual model, this connection to the assembly start event is not expressed since it is more of a technical workaround than a conceptual necessity.

In a similar manner, quality defects detected during the final quality assurance (QA) check are broadcasted as events. The quality defect report names the objected parts, so their numbers are included in the QA events. The station ID of the QA events is set to the QA station number, so the QA event structure is exactly the same as the incident report event structure: both have a station and a part number. Thus the origin of such an event is no longer of concern; both types are furthermore treated as “incident report events“.

Similarly to joining event streams it is possible to include static data in events. The join on the left adds additional material data (manufacturer and shipment ID) from the material database to the events. Finally there is an incident report event identifying station, part, manufacturer and shipment related to the incident.

In the next step the stream of these events is analyzed: the average incident frequency within the last full seven days is determined and is compared to the number of today’s incidents. If the number of today’s incidents exceeds 125% of last week’s average, the complex event “incident peak detected” is generated. It is amended with information about the statistical distribution of stations, manufacturers and shipments involved in these incidents.

The technical model of the second A-connector isn’t shown here since it is quite simple: if the station or the shipment distribution shows a significant accumulation of a single ID, the incident peak is likely to be caused there and must be investigated accordingly.

4. CONCLUSIONS AND OUTLOOK

In this paper we propose an extension to the Event-driven Process Chain (EPC) which is called Event Structure Model (ESM). On a conceptual level, ESM allows to associate different EPC models by specifying conditional transitions from one set of events to another one, thus defining processes which react to certain situations. On a technical level, ESM models can be amended with precise aggregation rules which allow their realization using Complex Event Processing (CEP) engines. However, the distinction between conceptual and technical layer allows separating the “business logic” from its realization so that CEP technology becomes accessible to non-technical business experts. Besides specifying ESM formally, we exemplified its usage in an application scenario in the automotive sector.

There are mainly two points of contact for subsequent research. First, on a conceptual level, other application scenarios that allowed for refining ESM, from industrial as well as non-industrial branches, would be a valuable addition to this work. Second, from a technical point of view, ensuring compatibility of different CEP engine concepts with ESM would significantly increase its practical relevance. Although both points will be addressed in the context of our further research efforts, a lively scientific discussion with diverse contributions could show up new and important aspects of conceptual event modeling.

5. REFERENCES

[1] Luckham D. C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Reading, 2006

[2] Greiner T., Düster W., Pouatcha F., von Ammon R., Brandl H.-M., Guschakowski D.: Business activity monitoring of norisbank taking the example of the application easyCredit and the future adoption of complex event processing (CEP). In Proc. Int. Symp. on Principles and Practice of Programming in Java. ACM, New York,2006, pp. 237–242

[3] Bry F., Eckert M.: Rule-Based Composite Event Queries: The Language XChangeEQ and its Semantics. In Proc. Int. Conf. on Web Reasoning and Rule Systems, Number

[4] A. Hevner, S. T. March, J. Park, S. Ram: “Design Science in Information Systems Research.“ MIS Quarterly 28, Nr. 1 (03 2004): 75-105.

[5] K. E. Weick: Sensemaking in Organizations, Thousand Oaks, CA, USA, 1995

[6] Widder, A.; v. Ammon, R.; Schaeffer, P.; Wolff, C.: Identification of suspicious, unknown event patterns in an event cloud. In: ACM International Conference Proceeding Series; Vol. 233. 2007.

[7] Imai, M.: Kaizen. The Key to Japan's Competitive Success. 1986.

[8] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik. A.-W. Scheer (Hrsg.). No. 89, Saarbrücken 1992.

[9] Object Management Group (OMG): Meta Object Facility (MOF) 2.0. http://www.omg.org/spec/MOF/2.0/ accessed: 03.05.2010.