Involving Business Users in the Design of Complex Event Processing Systems

Jens SCHIMMELPFENNIG, Dirk MAYER, Philipp WALTER, Christian SEEL

Research Projects Dept.
IDS Scheer AG
Altenkesseler Str. 17
66115 Saarbrücken, Germany
{jens.schimmelpfennig | dirk.mayer | philipp.walter | christian.seel}@ids-scheer.com

Abstract: Complex Event Processing (CEP) gained more and more attention in research and practical usage during the last years. Several engines and languages have been created and adapted in order to handle complex events. To describe the aggregation of simple events to more complex ones, these engines use technical and formal languages which are difficult to handle for business experts. Hence, business experts are often not properly involved in the definition of complex events. This lack in the area of requirement engineering worsens the result of many CEP projects. Therefore this paper proposes a conceptual modeling language that allows business experts to specify complex events according to actual 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.

1 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.
1 Introduction

Within the last years, complex event processing technology continuously matured and is going to leave its technical niche. The founding of the Event Processing Technical Society (EPTS), their aim to provide a foundation for analysis and communication [1] and the fact that nearly all big business software vendors extend their portfolios with a complex event processing (CEP) engine indicate that CEP is becoming more important for daily business. Apart from that, the possibility to fit more and more prefabricated parts or products with RFID tags in an affordable manner will also steadily increase the amount of real-time information that must be processed in an enterprise. Since these isolated pieces of information about one component are usually not relevant for business decisions, they must be aggregated and put into a larger context to allow for supporting business-level decisions. Hence the aggregation of single events to complex and business relevant ones with the help of CEP engines [2], [3] is coming to the fore. CEP engines allow aggregating several basic events to complex ones and provide a history of both types of events. Therefore complex events can be generated that trigger specific business processes or even invoke changes in a running process instance. For example the complex event “15 percent of parts from the same charge assembled within one hour have a different weight than expected” can trigger both a business process for removing the entire charge from production and a supply chain process to order new charges as reimbursement. Despite their relevance for business processes, complex events are commonly represented by query languages or XPath expressions [4].

These languages are well comprehensible for technical staff, so implementing a precisely specified CEP query is normally not much of a problem. However, the necessity for CEP usually originates from a business scenario that mostly non-technical domain experts with an economic background are familiar with, such as the stock manager in a manufacturing company. I.e., successful CEP applications require the involvement of business domain experts who usually lack of profound technical knowledge in complex event processing languages required for the specification of complex events. So from a technical point of view, the challenge is to allow business people to specify their CEP requirements in a standardized and technically sound way.

To bridge this gap between business and IT view and to involve business users and process owners in CEP implementations, we propose a conceptual semi-formal and graphical modeling language that enables users to specify complex events in a business-oriented manner. The language allows for using various data sources in order to describe complex events and the aggregation of basic and complex events to (higher) complex events in any number of aggregation levels. A technician can transform such conceptual CEP models more precisely and easily into technical CEP specifications than, requirements written down in textual form. Furthermore, complex events are required for business process modeling in order to describe reactions to complex event occurrences as business processes. After setting the basic principles, we demonstrate our approach using a production process from an automotive scenario as an example. The paper closes with a summary and provides an outlook on further research requirements.
2 Theoretical Foundations

2.1 Methodology Used

In an endeavour 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. [5]. 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 the research project ADiWa, funded by the German Ministry of Research and Education (BMBF), and in several industry consulting projects. The resulting artefact of the research process is a modeling tool and its underlying concept. Moreover, Weick’s sense making paradigm [6] is taken into account which ensures the consistency of the developed modeling language and its integration into business process modeling approaches.

2.2 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 exceed 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 an abstracted data representation of a real-world happening. Thus it is associated with that point in time when the underlying real-world incident happened. The event’s data structure highly depends on the scenario and real-world happening that is to be represented. Usually, 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, broadcast 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).
A popular example for the application of CEP is credit card fraud detection [7]. 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 which can be distinguished in two major paradigms. Stream-based approaches usually implement an SQL-like language to inspect and modify event streams [8], e.g., Esper [9], SASE [10] and StreamSQL [11]. Rule-based approaches allow specifying ECA rules for event processing, e.g., JBoss Drools [12].

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

3.1 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 car 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. For 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 the correct variant for a certain vehicle is assembled.

Today, following Kaizen [13] and total quality management principles, each employee working on the assembly line can stop the line by signalling 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 vehicle produced which may also reveal quality defects unnoticed during the assembly.
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 incidents 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 supplied 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.

3.2 Conceptual Modeling

In current business process management approaches, processes are documented using semi-formal modeling methods. The Event-driven Process Chain (EPC) [14] 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 behaviour 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 such as a slow increase in incidents exemplified above. It is not necessarily impossible to express such conditions in an EPC, but the model complexity would be immense and reduce the expressiveness and readability of the model. Therefore we propose to introduce a new modeling language called Event Structure Model (ESM) which supplements the EPC. It allows 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.

![Figure 1: Meta model of the Event Structure Model (ESM)](image-url)
In order to describe ESM we use OMG’s Meta Object Facility (MOF) [15]. MOF uses UML class diagrams that can be enhanced with Object Constraint Language (OCL) statements. Figure 1 shows the meta model of the ESM language.

The entry point of the meta model is the abstract class “Event”. Events can be distinguished 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. These 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 incoming 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.

![Diagram of the application scenario](image)

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.

### 3.3 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 necessary to amend it with technical details that specify the event aggregation operationally. For that purpose, platform independent 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 a technical 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 Conclusion 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 still poses a challenge. 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.
References


