

# Is Amazon Kinesis Data Analytics Suitable as Core for an Event Processing Network Model?

Arne Koschel<sup>1</sup> <sup>a</sup>, Irina Astrova<sup>2</sup>, Anna Pakosch<sup>1</sup>, Christian Gerner<sup>1</sup>, Christin Schulze<sup>1</sup>  
and Matthias Tyca<sup>1</sup>

<sup>1</sup>*Hochschule Hannover, DataH, University of Applied Sciences and Arts, Hannover, Germany*

<sup>2</sup>*Department of Software Science, School of IT, Tallinn University of Technology, Tallinn, Estonia*

**Keywords:** Event Processing Network (EPN), Event Processing Network Model, Amazon Kinesis Data Analytics.

**Abstract:** This article looks at a proposed list of generalized requirements for a unified modelling of event processing networks (EPNs) and its application to Amazon Kinesis Data Analytics. It enhances our previous work in this area, in which we recently analyzed Apache Storm and earlier also the EPiA model, the BEMN model, and the RuleCore model. Our proposed EPN requirements look at both: The logical model of EPNs and the concrete technical implementation of them. Therefore, our article provides requirements for EPN models based on attributes derived from event processing in general as well as existing models. Moreover, as its core contribution, our article applies those requirements by an in depth analysis of Amazon Kinesis Data Analytics as a concrete implementation foundation of an EPN model.

## 1 INTRODUCTION

Intelligent data management and processing has changed: Collecting large amounts of data from various sources happens in every company today, called 'Big Data'. It's not longer sufficient to store data in relational DBs, log files or events separately. Information from data combined from different sources, is important for the competitiveness of enterprises.

Batch Processing (Shaikh, 2019) is an established approach for processing 'Big Data'. At its core, data is collected and processed 'in batches'. Therefore, data is collected for a certain period of time before being processed. The drawback is, that no real-time processing is possible. First, data is collected for some time before processing takes place.

Recently, (Event) Stream Processing (Shaikh, 2019) joined the field. An approach to process data directly after generation. Through this near real-time processing, an action can happen immediately after processing. Enterprises can react faster to changes.


For the implementation of Stream Processing, a modeling technique called (complex) EPNs found its way into practice. This approach gives a guideline, included components and also requirements, how such a Stream Processing should occur.

Along with the rise of Stream Processing, tools were developed to model and implement EPNs. Therefore, we contribute here an evaluation of different recent tools, which support EPN realization as automatically as possible. The evaluation also addresses the following questions: What are EPNs and which requirements are important to provide them? A detailed look is taken at Apache Storm, Amazon Kinesis Data Analytics and Microsoft Azure Stream Analytics.

In this article, which enhances our work from (Schulze et al., 2023), we provide the following contributions: First, we briefly look at our – compared to our work from (Koschel et al., 2017) – more formally structured list of generalized EPN model requirements (as shown in (Schulze et al., 2023) in detail). Second, we provide – as the key contribution of the present article – an in-depth evaluation of Amazon Kinesis Data (AKD) Analytics with respect to our EPN model requirements. Moreover, we also briefly compare AKD to Microsoft Azure Stream Analytics.

Future work of ours will provide an in-depth evaluation of Microsoft Azure Stream Analytics regarding our EPN model requirements as well.

The remainder of this article is structured as follows: After discussing related work in Section 2, we place a brief introduction to the topic and provide our EPN model requirements in Section 3. Next, we take an in-depth look at Amazon Kinesis Data Analytics

<sup>a</sup>  <https://orcid.org/0000-0001-5695-2893>

in Section 4 followed by a brief comparison to Microsoft Azure Stream Analytics in Section 5. Eventually, Section 6 summarizes our results and concludes.

## 2 RELATED WORK

The basis of our project builds on authors in the scope of EPN and Complex Event Processing (CEP), for example, Dunkel and Bruns (Dunkel and Bruns, 2015) and (Bruns and Dunkel, 2010). We also used foundations from our earlier work on EPNs, namely (Koschel et al., 2017), (Koschel et al., 2018) and (Astrova et al., 2019). There, we more informally establish the requirements for EPNs and apply them to different EPN modeling approaches and tools (EPiA, BEMN, RuleCore). With the present paper, we extend our work with slightly refined and more formally structured requirements as well as a deep look at more recent tools, here in particular at Apache Storm.

Compared to our earlier work, we here cast the requirements into a template from (Rupp and Pohl, 2021), that means, we somewhat formalize them. We use a template to define the requirements in a standardized form and to show their importance. To ensure the quality of the requirements, we validated them against the quality criteria from (IEE, 1998).

Furthermore, we use the *IEEE830-1998* standard (IEE, 1998) for quality criteria for requirements. There exists a newer standard, the *IEEE/ISO/IEC29148-2011*, which describes the quality criteria from IEEE830-1998 in more summary form. We still meet all the quality criteria from both versions, except for the singularity. That one is new in IEEE/ISO/IEC29148-2011.

For the description and evaluation of the different tools, we used the documentation of the publishers. Apache offers large documentation in (Str, 2021a) for Apache Storm, Amazon Web Services provides information in (Str, 2021b) for Amazon Kinesis Data Analytics and Microsoft introduces Microsoft Azure Stream Analytics in (Microsoft, 2021).

We distinguish ourselves from other publications by standardizing and validating the requirements of EPNs and by evaluating various tools with different open or closed source characteristic, effort and costs. With this variety, we aim to give an overview of different tools and support the decision for a suitable tool.

## 3 EPNs AND REQUIREMENTS

This section curtly presents the basics of EPNs and the requirements for this kind of systems.

An event is 'a significant change of state' (Luckham, 2002). Event Processing Networks (EPNs) can be seen as generalized software systems that allow for the processing of events. However, EPN models lack standardization, which is where our work aims to contribute.

### 3.1 Basics of Event Processing Networks

EPNs are built on the basis of the Event-Driven Architectures (EDA) and Complex Event Processing (CEP). These both approaches

*'[...] represent a new style of enterprise applications that places events at the center of the software architecture - event orientation as an architectural style.'* — Dunkel and Bruns (Bruns and Dunkel, 2010, p. 4)

In this context, EDA is more about the design of event-driven architectures as a design style. CEP describes a technology for dynamic processing of large datasets (Bruns and Dunkel, 2010). Thus, CEP is a part of an EDA, which can be used for processing data within it. In detail, CEP describes the dynamic processing of large data streams (also called *event streams*) in real-time. An event is any happening in the system. Here, the change of state of a fact or an object is represented (Bruns and Dunkel, 2010).

The processing of events within a CEP is realized using rules. These rules contain knowledge about handling events or event sequences (Dunkel and Bruns, 2015). For the realization of these rules and the processing of the data, the CEP contains Event Processing Agents (EPA).

An EPN is a set of EPAs which are interconnected and exchange information during and about processing of the data (Dunkel and Bruns, 2015). An EPN can be interpreted as a graphical tool for modeling the flow of events for event processing systems (Koschel et al., 2017). Thus, the main components of EPN are EPAs in order to be able to perform CEP. EPAs contain various components, like Event Model, (Event) Rules and Event Procession Engine (Dunkel and Bruns, 2015).

Other components, such as producers, are also further elements of EPNs and can be taken from (Dunkel and Bruns, 2015) and (Koschel et al., 2017). The next part explains the requirements for EPNs.

### 3.2 Requirements

We evaluate the selected tools following an identical set of requirements, which we have put into a standardized form as show next.

### 3.2.1 Handling the Requirements

The set of requirements originates from our work in (Koschel et al., 2017). We have standardized the form of these requirements in (Schulze et al., 2023) by applying (Rupp and Pohl, 2021) and (IEE, 1998). The reason behind that was, that the original requirements were just described as bullet points, had no formal structure and were partially a little ambiguous.

To address these issues, we evaluated various requirement templates how they address issues such as writeability, readability and learnability and are commonly used (Robertson and Robertson, 2012). We have chosen (Rupp and Pohl, 2021) because it provides a straight-forward structure for requirements.

In addition, we apply the quality criteria of IEEE 830-1998 (IEE, 1998) to achieve high quality requirements in structure and content. Specification of the Quality criteria according to (IEE, 1998) are requirements, that are correct, unambiguous, complete, consistent, verifiable, modifiable, traceable and ranked for importance and/ or stability.

The requirements are formulated according to a template and fulfill all quality criteria. The requirements templates achieve writeability, readability and learnability and are therefore efficient. This also satisfies the modifiable criteria from IEEE 830-1993.

Correctness, unambiguousness and completeness are achieved by splitting, expanding and substituting specialist words. Requirements are checked to be consistent, verifiable, traceable and they are ranked by importance (see more details in (Schulze et al., 2023)).

### 3.2.2 The Requirements

Our standardized requirements are as follows:

- **EPNR1:** The tool shall offer the developer to model events with their inherent attributes as the central component of the engine.
- **EPNR2:** The tool shall map real world descriptions to events as scenarios.
- **EPNR3:** The tool shall offer event structures as simple, complex or aggregated. Simple events can be created and used independently. In addition, complex events have dependencies and references to other events. Also, aggregated events can be grouped logically.
- **EPNR4:** The tool shall offer possibilities to express the relativity of events and their temporal and causal relationships, e.g., sequence, preconditions and postconditions.
- **EPNR5:** The tool shall process and show the flow of events through the system.

- **EPNR6:** The tool shall offer the modeling of EPN by components, their properties and used patterns.
- **EPNR7:** The tool shall offer the modeling of components outside the system boundary and the behavior between inside and outside components.
- **EPNR8:** The tool should be expressive in usage, about readability, writability, learnability and efficiency.
- **EPNR9:** The tool should offer the developer further possibilities to create the model, e.g., IDE, graphical event programming.

In (Schulze et al., 2023) we evaluated Apache Storm (Str, 2021a). As the major contribution of the present paper, we will evaluate Amazon Kinesis Data Analytics against our requirements.

## 4 AMAZON KINESIS DATA ANALYTICS

In this section, the Amazon Kinesis Data (AKD) Analytics tool is examined and evaluated. It was chosen by the authors for its wide and modular use in Amazon Web Services (AWS).

### 4.1 Overview of AKD Analytics

For a consideration of AKD Analytics, the authors consider useful to first clarify the relationship between AKD Analytics and Apache Flink, since AKD Analytics is built on the Apache Flink framework.

#### 4.1.1 Apache Flink

Apache Flink is an open source framework for distributed event processing. The application can be used for batch or stream processing and offers stateful operators as a special feature (Fli, 2022d). In addition, it is designed to operate in all common cluster environments, however, it can also act as a standalone system and performs computations at in-memory speed at any scale.

Apache Flink Users report benchmarks of processing multiple trillions of events per day or maintaining multiple terabytes of states while running on thousands of cores (Fli, 2022a). In addition to AWS companies like Uber, Alibaba or Capital One use Apache Flink for streaming analytics, search ranking optimization or real-time activity monitoring and alerting (Fli, 2022e).

For CEP Apache Flink uses distributed applications that use a set of abstractions and technical realizations in the following way (Fli, 2022b): Operator

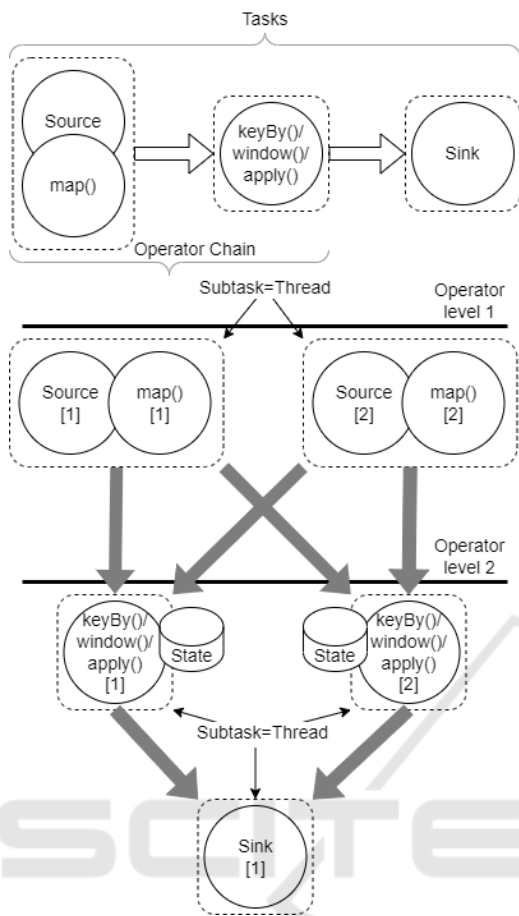


Figure 1: Stream processing in Apache Flink (based on (Fli, 2022c)).

subtasks are chained together into tasks. Each task is executed by one thread, this reduces the overhead of thread-to-thread handover and buffering and increases overall throughput while decreasing latency. The basic concept of the approach is *MapReduce*. As a special feature there are stateful operators (see Figure 1).

Apache Flink requires effective allocation and management of component resources in order to execute streaming applications. This can be achieved by integrating it with common cluster resources or Apache Flink can be set up to run as a standalone cluster or even as a library. We will only briefly mention the anatomy of a Apache Flink Cluster since the AKD Analytics context will handle the clustering and distribution in the system.

The Apache Flink runtime consists of two types of processes, JobManager and TaskManager (Fli, 2022b).

**JobManager.** The JobManager decides when to schedule the next task, reacts to finished tasks or execution failures, coordinates checkpoints and coordi-

nates recovery on failures. This process consists of three different components:

- The **ResourceManager** is responsible for resource de- or allocation and provisioning in a Apache Flink Cluster and manages the task slots.
- The **Dispatcher** provides a REST interface to submit Apache Flink applications for execution and starts a new JobMaster for each submitted job. It also runs the Apache Flink WebUI to provide information about job executions.
- A **JobMaster** is responsible for managing the execution of a single JobGraph.

**TaskManager.** The TaskManagers (or *workers*) execute tasks of a dataflow. Also, they are responsible for buffering and exchange the data streams. There must always be at least one TaskManager. The smallest unit of resource scheduling in a TaskManager is a task slot. Multiple operators may execute in a task slot (see Figure 1).

Each TaskManager is a JVM process and may execute one or more subtasks in separate threads.

An Apache Flink Application consists of multiple Apache Flink jobs. The execution of these jobs can happen in a local JVM (*LocalEnvironment*) or on a remote setup of clusters with multiple machines (*RemoteEnvironment*).

Jobs of an Apache Flink Application can either be submitted to a long-running Apache Flink *Session Cluster*, a dedicated Apache Flink *Job Cluster*, or an Apache Flink *Application Cluster*. The difference between these options is mainly related to the clusters lifecycle and to resource isolation guarantees.

#### Apache Flink Session Cluster.

- **Cluster Lifecycle:** In an Apache Flink Session Cluster, the client connects to a pre-existing, long-running Cluster that can accept multiple job submissions. After all jobs are finished the session is manually stopped.
- **Resource Isolation:** TaskManager slots are allocated by the ResourceManager on job submission and released once the job is finished. Because all jobs are sharing the same cluster, there is some competition for cluster resources. If some fatal error occurs on the JobManager, it will affect all jobs running in the cluster.
- **Other Considerations:** An existing cluster saves a lot of time when requesting resources and starting TaskManagers. This is important in scenarios where the execution time of jobs is very short and a high startup time would negatively impact



the end-to-end user experience as is the case with interactive analysis of short queries, where it is desirable that jobs can quickly perform computations using existing resources.

#### Apache Flink Job Cluster.

- **Cluster Lifecycle:** In an Apache Flink Job Cluster, the available cluster manager (like YARN) is used to spin up a cluster for each submitted job. The client first requests resources from the cluster manager to start the JobManager and submits the job to the Dispatcher running inside this process. Once the job is finished, the Apache Flink Job Cluster is torn down.
- **Resource Isolation:** A fatal error in the JobManager only affects that one job running in the Apache Flink Job Cluster.
- **Other Considerations:** The ResourceManager has to apply and wait for external resource management components to start the TaskManager processes and allocate resources. Apache Flink Job Clusters are more suited to large jobs that are long-running, have high-stability requirements and are not sensitive to longer startup times.

#### Apache Flink Application Cluster.

- **Cluster Lifecycle:** An Apache Flink Application Cluster is a dedicated Apache Flink Cluster that only executes jobs from one Apache Flink Application and where the main() method runs on the cluster rather than the client.  
The job submission is a one-step process: There is no need to start a Apache Flink Cluster first and then submit a job to the existing Cluster session, instead the application logic and dependencies are packaged into a executable job JAR and the Cluster entry point (ApplicationClusterEntry-Point) is responsible for calling the main() method to extract the JobGraph. This allows to deploy an Apache Flink Application like any other application on Kubernetes, for example. The lifetime of an Apache Flink Application Cluster is therefore bound to the lifetime of the Apache Flink Application.
- **Resource Isolation:** In an Apache Flink Application Cluster, the ResourceManager and Dispatcher are scoped to a single Apache Flink Application, which provides a better separation of concerns.

After the short introduction to Apache Flink as the basis of AKD Analytics we look at the actual tool next.

#### 4.1.2 Analytics in Amazon Kinesis

AKD Analytics is part of the AWS Kinesis portfolio, which still consists of AKD Firehouse for data persistence. As well as AKD Streams and Amazon Kinesis Video Streams for scalable continuous Streaming of data. Video Streams is specialized in video data.

In this portfolio, AKD Analytics is designed for process data from streams or batches up to real-time processing. Like mentioned before, this processing is done by as an Apache Flink Application and AKD Analytics is a framework for this use. AKD Analytics also offers SQL processing, but this is flagged legacy or deprecated. AKD Analytics does not necessarily require Amazon Streams for processing, but can basically be used with other tuple-based streams as well.

#### 4.1.3 Amazon Kinesis Data Analytics Framework

To use AKD Analytics an AWS Identity and Management Account (IAM) and the AWS Command Line Interface (CLI) are needed. This paper will not discuss these steps further, as they are necessary in general to use AWS services.

An AKD Analytics Application has the following components (see Figure 1 (Com, 2022)):

- **Runtime properties:** Runtime properties are for runtime configuration and independent of the application code.
- **Source:** Sources are the input for an application and can be any kind of tuple-based formats, e.g., Kinesis Data Stream or Apache Kafka (DA-, 2022a).
- **Operators:** The application processes the data by means of tasks that are implemented in operator chains. An operator can transform, enrich, or aggregate data. Operators can have states.
- **Sink:** The results of the calculation are directed into sinks, e.g., Kinesis data stream, a Kinesis Data Firehose delivery stream, an Amazon S3 bucket.

To use a CEP application in AKD Analytics the following steps are needed: There must be at least one stream that the application can process. There must be at least one sink for the results. Both must be created or connected to the AWS Kinesis environment in order to be used by the Analytics application.

In order to implement the desired business-logic, AWS provides Java example code that can be cloned from a Git repository (DA-, 2022b).

The core features of this example are: A *Project Object Model* file, containing information about the

applications configuration and dependencies, including the AKD Analytics libraries.

The *BasicStreamingJob.java* file containing the main method that defines the application's functionality. The example uses a Kinesis source to read from the source stream, that is where the source has been connected before. It creates source and sink connectors to access external resources using a *Stream-Execution-Environment* object, in default these are created using static properties. To use dynamic application properties, *createSourceFromApplicationProperties* and *createSinkFromApplicationProperties* methods to create the connectors can be used. These methods read the applications properties to configure the connectors (DA-, 2022c).

After the desired logic is implemented, the code is compiled to a JAR and uploaded to AWS. The JAR is then used to create and run a AKD Analytics Application that connects to the source and sinks that has been set in the AWS before. Finally the AKD Analytics Application is started and the data is processed (DA-, 2022c).

In practice, it boils down to the following: connect a stream source, connect a sink, implement the required business logic, i.e. using the Java example, upload and start it in the AWS. Amazon now will handle everything else corresponding to your AWS plan.

Thus, we examined Apache Flink and how it works as a foundation for AKD Analytics, considering the flow and distribution of processing. Next, we considered the commissioning of an AKD Analytics application. This demonstrated that AKD Analytics is a framework. Finally, it was explained which components can be controlled in the framework and which components can be implemented exclusively as logic within the Apache Flink foundation.

In the next part, we evaluate Amazon Kinesis Data Analytics against our EPNM requirements.

## 4.2 Evaluation of Amazon Kinesis Data Analytics for Modeling EPN

This part will argue, which requirements are fulfilled by AKD Analytics.

- **EPNR1** - fulfilled:  
Events are abstracted by tuples. Tuples contain structured lists of attributes which can take any primitive or complex value. This can be done static or dynamic, hence no restriction.
- **EPNR2** - fulfilled:  
Tuples from the source stream can be dynamically adapted to match and describe any real world scenario.

- **EPNR3** - fulfilled:  
Basically, AKD Analytics abstracts the concrete processing of events by Apache Flink. Nevertheless, the tool offers the required functions for the runnable AKD Analytics Application by means of operators.
- **EPNR4** - fulfilled:  
Through the processing steps within the Apache Flink operator chain, causal relationships and dependencies can be mapped. Furthermore, the tool offers the direct possibility to look at temporal and causal relations within or between several stateful operators.
- **EPNR5** - not fulfilled:  
Although the event flow can be monitored and evaluated using AWS metrics, there is no concrete representation in AKD Analytics in the sense of this requirement, since this is precisely part of the abstraction that AKD Analytics performs.
- **EPNR6** - fulfilled:  
The tool is based precisely on the fact that components, their properties and patterns to be applied just model and do not have to be concretized. This is done by the framework.
- **EPNR7** - fulfilled:  
By default the components, which lie outside of the system borders, are represented statically by abstract objects. However, these can also be adapted dynamically, so that the behavior of these components can be modeled.
- **EPNR8** - fulfilled:  
In AWS, there are many tutorials in written and video form on how to build a AKD Analytics Application. It exists an IDE. For Java, there is a prepared example that can be cloned from Git and then only needs to be extended by the functional code. Developer basically only need to familiarize themselves with the functionalities of the library provided.
- **EPNR9** - fulfilled:  
AWS offers AKD Analytics Studio, a specialized IDE with an interface for Scala, Python, or SQL. In addition to the standard features of an IDE, Studio also offers visualization capabilities.

In conclusion, AKD Analytics is a suitable tool for CEP, can represent things of the real world and guarantees the processing of data at any scale. To use it, the AWS environment is required. In return, the developer can fully focus on the domain, as AWS takes care of and guarantees scaling and deployment at the desired quality.

Table 1: Tools – Criteria, Characteristics, Comparison.

|                                | <b>Amazon Kinesis Data Analytics</b> | <b>Microsoft Azure Stream Analytics</b>            |
|--------------------------------|--------------------------------------|--|
| <b>Basis</b>                   | MapReduce                            | Trill  |
| <b>Support</b>                 | Platform-as-a-Service (PaaS)         | Platform-as-a-Service (PaaS)                       |
| <b>Costs</b>                   | costs based on usage                 | costs based on usage                               |
| <b>Effort</b>                  | low                                  | average  |
| <b>Event format</b>            | Tuple                                | JSON, AVRO, csv                                    |
| <b>Environment</b>             | AWS                                  | Azure  |
| <b>Language</b>                | Java, Scala, Python, SQL             | SQL based, JavaScript or C# user-defined functions |
| <b>Distribution</b>            | Thread-based in Apache Flink         | no details   |
| <b>Maximum Processing Rate</b> | real-time                            | real-time  |
| <b>Reliability</b>             | guaranteed by AWS                    | 99.9%promised                                      |
| <b>Data protection</b>         | Server location can be set           | Server location can be set                         |
| <b>Security</b>                | provided by AWS                      | provided by Azure                                  |

## 5 COMPARISON

In this article, we analyzed in Section 4 Amazon Kinesis Data Analytics (AKD) in-depth with respect to our standardized set of requirements from Section 3.2.2. We examined that Amazon Kinesis Data Analytics mostly fulfills our requirements. An exception is **EPNR5**, however, this is more due to 'the nature' of AKD. Generally speaking, with the respect of our requirements, AKD is well suited for the realization of EPNs.

To provide some more distinctive criteria to other tools, we took in particular a more developer-oriented perspective. The result is summarized in Table 1, which briefly compares two of our tools under evaluation, namely AKD Analytics and Microsoft Azure Stream Analytics (ASA). Developers may use this table to identify the most important criteria that argue for or against a tool. In particular an open source nature (cf. our analysis of Apache Storm in (Schulze et al., 2023)), price, convenience, and potential vendor lock in some distinctive factors.

Mainly, the collected information for the presentation of AKD Analytics (as well as other tools) was taken from the documentation of the publishers or developers of it. Due to this, some information may be presented subjectively, as companies would like to widely distribute their tool in any case. Moreover, AKD and Microsoft ASA are costly tools, so an advertising factor within the documentation cannot be ruled out. Even more, information may be incomplete

because companies want to keep their implementations private.

Both, AKD Analytics and ASA are commercial products and thus not free of charge. Maintenance is supported by the vendors themselves, both are nicely hosted and maintained by Amazon respectively Microsoft and thus possibly easier to be used compared to self-hosted systems, such as Apache Storm (cf. (Schulze et al., 2023)). Thus, there is no clear winner between AKD Analytics and ASA, but more a question of individual developer taste and skills as well as company preferences. For example, if a company is an AWS shop anyway, wants likely less maintenance effort and is able to pay the costs for AKD Analytics, then it could be more favorable. Similar arguments hold for Microsoft ASA respectively.

## 6 CONCLUSION

In this article, we had a deep look at Event Processing Network Models, as a foundation of Event Stream Processing tools (cf. (Schulze et al., 2023)) and presented our enhanced (compared to our earlier work) standardized set of requirements for EPN models in Section 3.2.2.

As the key contribution of this article, we apply those requirements for an in-depth look at Amazon Kinesis Data Analytics in Section 4. It turns out, that Amazon Kinesis Data Analytics is a well suited tool for modeling and implementation of EPNs. Addition-

ally we briefly compared Amazon Kinesis Data Analytics with Microsoft Azure Stream Analytics in Section 5. Already in (Schulze et al., 2023) we evaluated Apache Storm.

Since all those tools mostly fulfilled our requirements, comparisons may need other criteria as well. The suitability of a tool depends on more individual circumstances, such as which kind of 'shop' you are – for example, Amazon vs. Microsoft –, but also how high the own development and administration effort should be.

In future work of ours, we will also provide in-depth evaluations of Microsoft Azure Stream Analytics regarding our EPN model requirements.

Therefore, the decision for a suitable tool is based on the effort, the control and the costs involved. For these reasons, no absolute recommendation can be made. Rather the authors recommend examining each individual use case or at least a set of typical ones, in order to select the ideal tool.

## REFERENCES

- (1998). IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std 830-1998*, pages 1–40.
- (2021a). Apache Storm. *Apache Software Foundation*. Online: <https://storm.apache.org/> [retrieved: 04, 2022].
- (2021b). Streaming Data Solutions on AWS. *Amazon Web Services Inc.* Online: <https://docs.aws.amazon.com/whitepapers/latest/streaming-data-solutions-amazon-kinesis/welcome.html> [retrieved: 04, 2022].
- (2022a). Adding Streaming Data Sources to Kinesis Data Analytics for Apache Flink. *Amazon Web Services Inc.* Online: <https://docs.aws.amazon.com/kinesisanalytics/latest/java/how-sources.html> [retrieved: 04, 2022].
- (2022a). Apache Flink. *Apache Software Foundation*. Online: <https://flink.apache.org/flink-architecture.html> [retrieved: 04, 2022].
- (2022b). Apache Flink Stream Concepts. *Apache Software Foundation*. Online: <https://nightlies.apache.org/flink/flink-docs-release-1.14/docs/concepts/flink-architecture/> [retrieved: 04, 2022].
- (2022c). Apache Flink Stream Processing. *Apache Software Foundation*. Online: <https://nightlies.apache.org/flink/flink-docs-release-1.14/docs/learn-flink/overview/#stream-processing> [retrieved: 04, 2022].
- (2022d). Apache Flink Usecases. *Apache Software Foundation*. Online: <https://flink.apache.org/usecases.html> [retrieved: 04, 2022].
- (2022e). Apache Flink Users. *Apache Software Foundation*. Online: <https://flink.apache.org/poweredy.html> [retrieved: 04, 2022].
- (2022). Components of a Kinesis Data Analytics for Flink Application. *Amazon Web Services Inc.* Online: <https://docs.aws.amazon.com/kinesisanalytics/latest/java/getting-started.html?pg=ln&cp=bn#getting-started-components> [retrieved: 04, 2022].
- (2022b). Data Analytics Java Exmaple. *Amazon Web Services Inc.* Online: <https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples> [retrieved: 04, 2022].
- (2022c). Data Analytics Java Exmaple Usage. *Amazon Web Services Inc.* Online: <https://docs.aws.amazon.com/kinesisanalytics/latest/java/get-started-exercise.html> [retrieved: 04, 2022].
- Astrova, I., Koschel, A., Kobert, S., Naumann, J., Ruhe, T., and Starodubtsev, O. (2019). Evaluating RuleCore as Event Processing Network Model. *Proc. 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*, pages 297–300.
- Bruns, R. and Dunkel, J. (2010). *Event-Driven Architecture - Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse (Software architecture for event-driven business processes)*. Springer.
- Dunkel, J. and Bruns, R. (2015). *Complex Event Processing - Komplexe Analyse von massiven Datenströmen mit CEP (Complex analysis of massive data streams with CEP)*. Springer Vieweg.
- Koschel, A., Astrova, I., Kobert, S., Naumann, J., Ruhe, T., and Starodubtsev, O. (2017). Towards Requirements for Event Processing Network Models. *Proc. 8th International Conference on Information, Intelligence, Systems, Applications (IISA 2017)*, pages 27–30.
- Koschel, A., Astrova, I., Kobert, S., Naumann, J., Ruhe, T., and Starodubtsev, O. (2018). On Requirements for Event Processing Network Models Using Business Event Modeling Notation. *Proc. 2018 Conf. Intelligent Computing. Advances in Intelligent Systems and Computing (SAI 2018)*, pages 756–762.
- Luckham, D. (2002). *The Power of Events*. Addison Wesley, USA.
- Microsoft (2021). Introduction to Azure Stream Analytics. *Microsoft Documentation*. Online: <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction> [retrieved: 04, 2022].
- Robertson, S. and Robertson, J. (2012). *Mastering the Requirements Process: Getting Requirements Right*. Addison-Wesley Professional.
- Rupp, C. and Pohl, R. (2021). *Basiswissen Requirements Engineering (Basic knowledge Requirements Engineering)*. dpunkt.verlag.
- Schulze, C., Gerner, C., Tyca, M., Koschel, A., Pakosch, A., and Astrova, I. (2023). Analyzing Apache Storm as Core for an Event Processing Network Model. *Proc. International Conference Intelligent Systems Conference (IntelliSys 2023)*.
- Shaikh, T. (2019). Batch Processing — Hadoop Ecosystem. *K2 Data Science and Engineering*. Online: <https://blog.k2datascience.com/batch-processing-hadoop-ecosystem-f6da88f11cae> [retrieved: 04, 2022].