Daitan GROUP

Accelerating™

DAITAN WHITE PAPER

Event-Driven Architecture

HOW TO PREPARE, MIGRATE AND SCALE AN EVENT-DRIVEN ARCHITECTURE

White Paper Contents

Introduction

Organizations have been designed around processes since the Industrial Revolution. Optimizing processes to increase productivity is still the main goal of most business organizations. This type of thinking leads organizations to develop IT infrastructure optimized for managing internally focused, enterprise-centric transactional models for implementing business processes.

In the twenty-first century, the transition to Digital Business changes everything. Organizations are switching from inward-facing enterprise-centric models to outward-facing ecosystem-centric models. With new ecosystem focus comes a transition from data-centric thinking to event-centric thinking.

The concept of event-centric thinking emerged in the early 2000's initially around Messaging and eventually became known as Event-Driven Architecture (EDA). This idea would enable companies to leverage events into real-time actionable information. By definition, "*Event-driven architecture (EDA), is a software architecture pattern promoting the production, detection, consumption of, and reaction to events.*"[1]

As examples, Waze and Uber are two prominent companies demonstrating how Event-Driven Architecture is key to delivering disruption in ways that today's Digital Businesses compete in an on-demand economy.

[1] K. Mani Chandy *Event-Driven Applications: Costs, Benefits and Design Approaches*, California Institute of Technology, 2006

"Most CEOs recognize a triangular relationship between technology, product improvement and growth. They recognize that technology is the fundamental enabler of digital transformation and leading digital companies have figured out that EDA is the 'secret sauce' that gives them a competitive edge."

- Ann Thomas, VP & Distinguished Analyst, Gartner

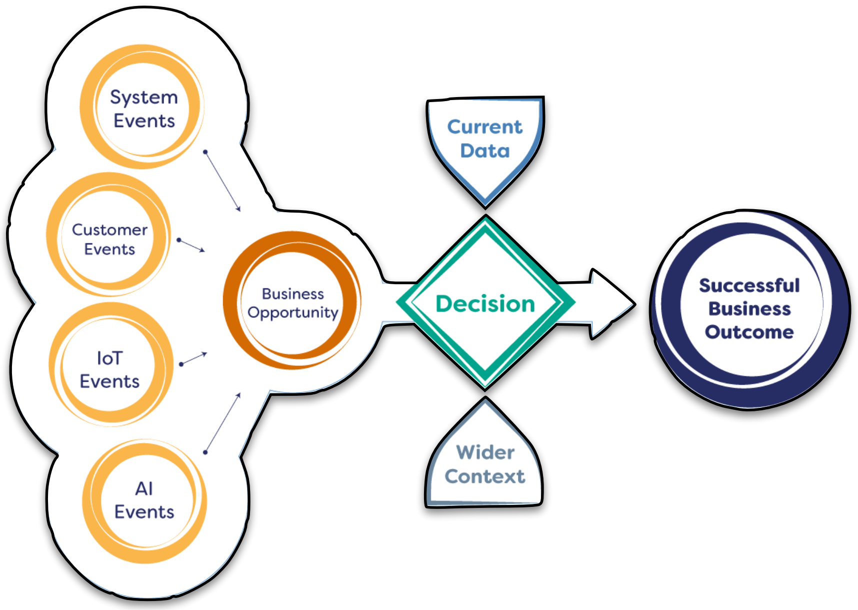
Event-Driven Architecture Enables Digital Transformation

Situational awareness, real-time responsiveness and informed decision making are critical to managing ecosystems that include suppliers, business partners and ultimately customers. External requirements including conformance and environment protection are additional drivers toward event-centric ecosystem thinking.

Event-Driven Architecture (EDA) supports and enables critical trends in the transformation to Digital Business. According to Gartner, some examples include:

- **Responsive Customer Engagement:** Events with added situational awareness enable context enriched customer experiences.
- **Capitalizing on Business Moments:** Coalesced groups of events indicate significant business moments. Companies can respond in (near) real time.
- **Internet of Things:** Events from sensors are the single most important foundation of IoT. Event-Driven Architecture is the best option.
- **Artificial Intelligence and Machine Learning:** Many of the applications involve loops of event collection, analysis and response.

Figure 1: EDA is a Prerequisite for Capitalizing on Business Moments[2]



[2] Articulating the Business Value of Event-Driven Architecture Gartner 2017

“An event is a change in state that merits attention from systems.”

- *Event-Driven Architecture: How SOA Enables the Real-Time Enterprise*, 2009

To be successful, Digital Businesses must respond to Business Events containing multiple information elements from multiple ecosystem sources. In response to Event Streams, Business Moments are recognized and handled by Smart Event Brokers triggering the best beneficial response.

Waze and Uber are two prominent examples in which Event-Driven Architecture is key to delivering a disruptive service within the **On-Demand Economy.** Waze is constantly collecting events from vehicles and traffic monitors. These are used to find the fastest driving routes and balance traffic across them. Uber monitors drivers and rider demand, to ensure drivers are properly located in high-demand areas.

> *“Event-driven IT is an essential enabler of digital business, but pervasive, competent use of event-driven architecture is not achieved simply by implementing certain technologies or adhering to architectural-design guidance. It requires a new way of thinking and cultural change. Specifically, it demands a shift from the familiar model for designing and executing data-centric transactions to a focus on continuous sensing, assessing and responding to streams of business events — ‘event thinking.’”*
>
> “Business Events, Business Moments and Event Thinking in Digital Business”
> Gartner, 4-August 2017

In short, Digital Businesses must adopt Event Thinking to operate in complex ecosystems, provide real-time responsiveness, make context-informed decisions, be agile and scale.

On-Demand Economy

“The On-Demand Economy is defined as the economic activity created by technology companies that fulfill consumer demand via the immediate provisioning of goods and services. Supply is driven via an efficient, intuitive digital mesh layered on top of existing infrastructure networks.”

- Business Insider 2014

Daitan Group is supporting our clients in their efforts to adopt Event-Driven Architecture. Event thinking and adopting Event-Driven Architecture (EDA) are compatible with other components of the Digital Business Stack.

In a recent case study Daitan published, ***Use of Big Data, Analytics and Data Science in Telecommunications***[3], we discussed Artificial Intelligence and Machine Learning as input funnels to Smart Event Brokers that ensure events are routed to the best responding Event Handlers. Similarly, in an earlier Daitan white paper title, ***Renovate to Innovate***[4], we discussed architecture renovation and microservices emerged as a good approach to handle elastic demand, as it is often the case with EDA.

Here we will focus on events as the ‘front end’ to Digital Business. It’s part of our ongoing guide for our partners and clients who see Digital Business as a key to near real-time responsiveness and thriving in complex ecosystems—all success factors in twenty-first century digital business.

[3] Use of Big Data, Analytics and Data Science in Telecommunications, Daitan Group 2017 http://www.daitangroup.com/wp-content/uploads/2017/09/Daitan_CaseStudy_Telecom_Advanced_Analytics.pdf

[4] Renovate to Innovate, Daitan Group 2017 http://www.daitangroup.com/wp-content/uploads/2017/01/Renovate_to_Innovate.pdf

“Messaging is a technology that enables high-speed, asynchronous, program-to-program communication with reliable delivery.”

- *Event-Driven Architecture: How SOA Enables the Real-Time Enterprise,* 2009

Messaging as a Precursor to EDA

Without question, Messaging was a key precursor to Event-Driven Architecture. EDA is built on the principles of Messaging, and today it remains a key foundation.

Most Message Oriented Middleware (MOM) systems support asynchronous communication. This reduces time dependencies between applications. Messages are queued until the consumer is ready. This is a more effective way to guarantee system responsiveness. Messaging can also increase reliability using intrinsic “guaranteed delivery” mechanisms.

While EDA is built on the principles of Messaging, it also requires more sophisticated, faster and scalable event analysis not found in traditional MOM. These capabilities are detailed below in the section, “Technical Overview of Event-Driven Architecture.”

The Event-Driven Mindset

Message Oriented Middleware (MOM)

“Message-oriented middleware (MOM) is software or hardware infrastructure supporting sending and receiving messages between distributed systems”

- Wikipedia

All organizations capture and handle events. For example, any customer purchase is a series of events and fulfilling the purchase involves handling the events. Events are typically handled by microservices and APIs that handle requests and generate responses.

Order fulfillment is based on simple events and such event handling is enterprise-focused. The migration to EDA involves thinking of events more broadly. Organizations should adopt a broader ecosystem focus in which events can be received from and/or handled by ecosystem partners. In addition, events should be augmented with situational information to add value.

Applying such event thinking to a simple purchase might raise questions and reveal corresponding opportunities. For example:

- Does the purchase create an unusual impact on parts inventory or production demands? Should the response trigger an order from a supplier or ecosystem partner? Should it trigger a change in production schedule/commitments? Should it cause an alert to channel or distribution partners?
- Is the purchase somehow tied to external events? Can sensing such events be used to predict business changes when such external events recur? Can new event flows be created to anticipate such external events?

- Does analyzing the purchase result in some learning that would benefit a supplier, after-market service provider or other ecosystem partner?

Fan gear is a simple consumer-focused example of a purchase event. When a team does well, fans buy more hats, jerseys and banners. The entire manufacturing chain can respond by adjusting production levels. Similarly, wholesalers, distributers and retailers can adjust their purchasing and inventory to anticipate changing demand. A sports league such as the NFL manages online shops for all teams. Events related to team and player performance can be used to intelligently drive production, distribution and retail flows.

Adopting the event-driven mindset is the first step. Discovering new event flows, adding situational awareness and linking to an ecosystem are all keys. This is the way to find new business opportunities from Event-Driven Architecture.

Microservices Enable Event-Driven Architecture

"Microservices are generally well-suited to Event-Driven Architecture (EDA) migration. Organizations that have not implemented well-defined Microservices interfaces might want to consider that step before focusing on EDA."

- Daitan Group

How Various Industries Leverage EDA

With Digital Business adoption, companies are managing more digital events every year. Near real-time responsiveness and situational awareness are required. Two key drivers are Internet of Things (IoT) sensors collecting events and the emergence of companies participating in the On-Demand Economy.

Our case studies start with an example of Daitan's experience implementing EDA. Followed by Uber and Deliveroo, two companies disrupting their markets with on-demand models. Then, Monzo Bank, an emerging Financial Technology or "FinTech" company that has recently transitioned into full consumer banking 100% online. Finally, Centrica is the parent company of British Gas and other energy providers. They give IoT sensors to homeowners, enabling energy cost savings, improved safety and higher quality customer service.

Telecommunications Industry Implements High Capacity Data Pipelines

For a telecommunications operator, continuous service and delivery of the right service, for the right user, at an appropriate cost is the lynchpin to business. Fast event handling and intelligent event analysis are two critical capabilities for supporting these business goals.

Daitan Group has direct experience implementing Event-Driven Architecture for Telecom.[5] Our client's product is capable of 15Bn messages per day comprising 13TB of data. Messages are collected from

[5] Use of Big Data, Analytics and Data Science in Telecommunications, Daitan Group 2017 http://www.daitangroup.com/wp-content/uploads/2017/09/Daitan_CaseStudy_Telecom_Advanced_Analytics.pdf

an ecosystem including multiple vendors, each with their own data format. Data includes call events, signal strength measures, data link usage, geolocation and logs. The data originates from various points in the fixed infrastructure as well as mobile devices.

The system is architected to scale to large volumes of data. Multiple clusters of tools are provisioned to handle data flow from collection to business analysis. Event analysis and Machine Learning are used to improve on key business objectives. Quality of service can be measured and improved. Customer usage patterns are leveraged to recommend and up sell new services. Optimizing infrastructure utilization reduces costs. Detecting a typical usage patterns reveals and helps eliminate fraud.

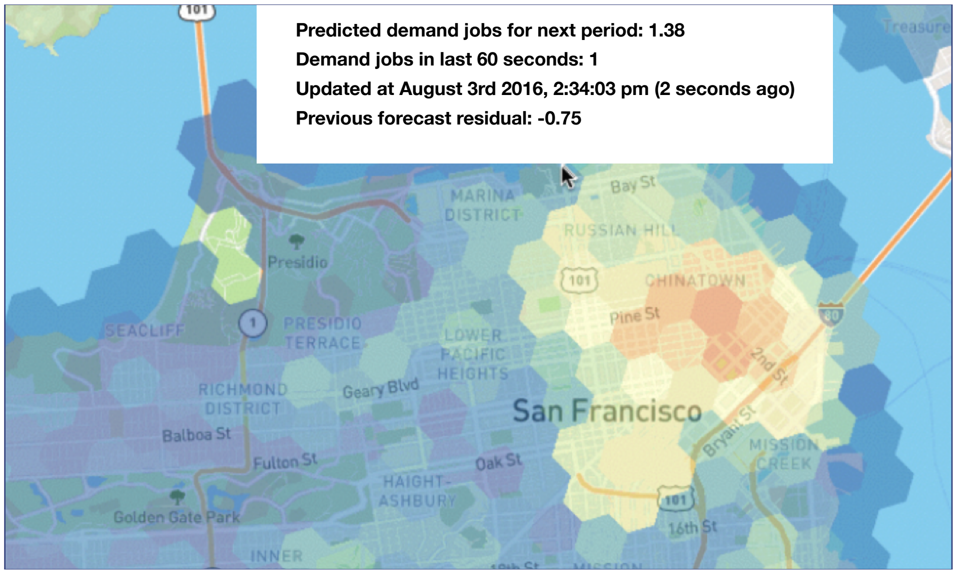
Uber Changed Ride Hailing Forever

UBER

Uber has a global footprint providing on-demand transportation for people in hundreds of cities. Invisible to consumers, Uber's backend infrastructure runs a continuous, real-time balancing system to ensure drivers with cars are positioned correctly for rider demand.

Uber's system responds to—and tries to predict—changes in driver position and rider demand. Every city is divided into a grid of small hexagonal regions. For each small region, recent activity is sampled and historical activity is measured. When the system detects a supply-demand mismatch, drivers nearby are instructed to move and consequently resolve the mismatch.

Figure 2: Stream Processing & Analytics with Flink: Uber Technology Presentation


Predicted demand jobs for next period: 1.38
Demand jobs in last 60 seconds: 1
Updated at August 3rd 2016, 2:34:03 pm (2 seconds ago)
Previous forecast residual: -0.75

Uber's open-source streaming analytics platform, AthenaX, runs on top of Apache Flink Event Stream Processing (ESP)[6].

Today Uber's system monitors roughly one million small regions worldwide, each with an activity event stream. The scaled system is able to capture these events, aggregate them with context and analyze them. This triggers driver instructions that optimize Uber's system performance.

Deliveroo Delivers High-End Dining-In Experience

deliveroo

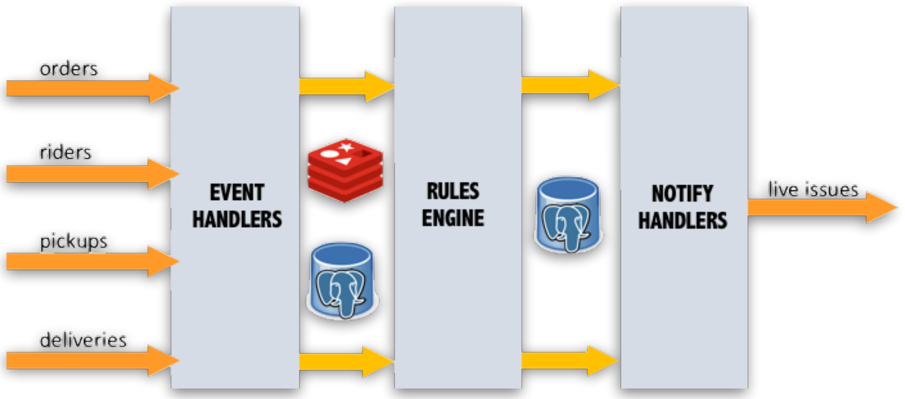
Deliveroo provides on-demand transportation for restaurant meals. The company specializes in delivery, focusing on high-quality restaurants that do not provide delivery and customers in areas beyond delivery range.

Deliveroo launched its business in Central London, UK. As they have expanded their geographic reach, the company has also developed their own satellite kitchens known as "Deliveroo Editions." These kitchens are positioned near customer demand. Participating restaurants share occupancy in these kitchens. Food is prepared and dispatched for faster delivery close by. Deliveroo has six satellite kitchens in Greater London with more planned as part of their rapid expansion.

Deliveroo's original challenge was matching delivery driver supply with meal delivery demand. With satellite kitchens, the problem is more complex involving balanced distribution of ingredients, kitchen supplies and kitchen staff. Deliveroo's daily challenge is delivering meals quickly and accurately. Over weeks and months, the Deliveroo system must gain awareness of demand changes and trends. The network of satellite kitchens must be agile and elastic in response to changes in customer demand.

To meet service challenges Deliveroo has built a large-scale architecture that is 100% event driven. An event bus is used to manage the lifecycle of all events. The following diagram from Deliveroo illustrates the flow:

Figure 3: "The Distributed Pit of Success: Deliveroo Technology Presentation"



[6] AthenaX http://athenax.readthedocs.io/en/latest/#technical-specs

Event handlers, also known as event collectors, receive events. The rules engine analyzes events, adds situational information and notifies event handlers, also known as event responders. Communication is managed by publishing and subscribing to events. All event handlers and responders are implemented as microservices. Small engineering teams each manage 1-4 microservices.

Monzo the Mobile-First Bank Disrupts Traditional Banking

UK startup Monzo Bank has proven that it's possible to build a cloud-based, mobile-first bank in compliance with international banking regulations. In April 2017, Monzo Bank was granted a full banking license in the UK. This allows the bank to begin managing customer deposits on a wide and fully insured scale. For example, Monzo Bank expects to offer current accounts to all customers by early 2018.

Monzo Bank was founded in February 2015. The bank is committed to cloud and mobile. Core banking applications are hosted on Amazon Web Services (AWS). This enables elastic provisioning to meet all changes in capacity requirements. Infrastructure availability, fault tolerance and maintenance are all managed by AWS.

Applications are all based on Microservices architecture running across virtual servers using container tools including Docker and Kubernetes. The Microservices architecture comprises a collection of event-driven components that can scale independently and communicate asynchronously.

Monzo Bank exploits machine learning to improve customer satisfaction. All customer interactions are stored as sequences of actions known as "event time series". These series are analyzed to uncover patterns. Such patterns can be used to streamline common customer flows, anticipate customer needs and provide predictive solutions. This represents the confluence of two important digital business trends: The move to Event-Driven Architecture and the use of Machine Learning to improve customer satisfaction.

Centrica is Making the Connected Home a Reality

Centrica is an energy and services company. Centrica supplies energy to residential and business customers. Based on the 2015 acquisition of AlertMe, Centrica has started providing Internet of Things devices including smart thermostats and vehicle charging points to its Connected Home customers. Centrica has built a system that collects and analyzes events from devices to improve customer service, provide increased safety and reduce energy consumption.

In a Centrica Connected Home various sensors generate events which are enhanced, aggregated and dispatched to Centrica or the customer.

Figure 4: Sensors in a Centrica Connected Home

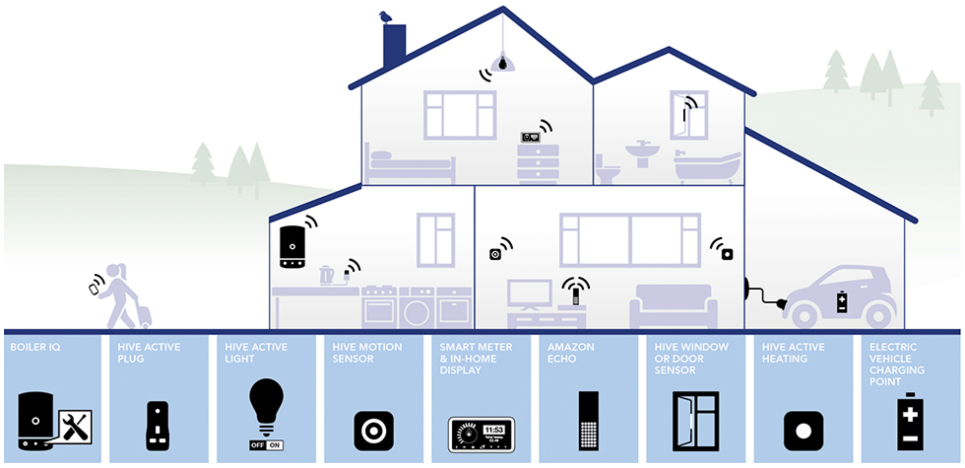


Figure 5: Device Categories in a Centrica Connected Home

	Safety	Better Service	Reduce Consumption
Boiler IQ - Detect faults	✓		
Active Plug & Light	✓		✓
Motion Sensor	✓		✓
Smart Meter & Thermostat			✓
Window/Door Sensor	✓		
Vehicle Charging Point		✓	✓
Amazon Echo Voice command control		✓	
			Source: Centrica

Technical Overview of Event-Driven Architecture

We have discussed the business motivations for EDA, and responding in near real time to changes in the business context to improve customer experience and leverage business opportunities is the top priority.

So from the technical perspective, why this used to be a difficult task? The answer lies into the most common software architecture organizations have been using for decades to collect, analyze and make decisions based on data - the batch-oriented Data Warehouse (DW).

On a typical DW architecture, data is periodically (usually daily or every few hours) fetch from the organization's applications by a specialized ETL (Extract-Transform-Load) tool and stored in an analytical database after possibly some data normalization and transformation. Once data has been imported into the DW, other tools can start analyzing it and deriving some insight, generating reports or dashboards.

Needless to say, this whole process can take several hours in the best scenario, which is a long time if the organization's objective is to respond quickly to an interaction with a customer that is happening just now, for example.

Additionally, the stack of tools and skills required to build an end-to-end DW solution involves significant investment and maintenance costs.

ESP tools enable EDA because they continuously analyze data as it is generated and allow for low latency responses.

The relationship between Messaging and Event-Driven Architecture was explained earlier, in the section Messaging as a Precursor to Event-Driven Architecture. Today it remains a key foundation. Messaging provides the foundation for collecting, queueing, routing and consuming messages.

Technically, a number of popular technologies can be related to Event-Driven Architecture:

- Event capturing can be facilitated by an Integration PaaS such as Dell Boomi, SnapLogic, and MuleSoft or Stream Data Integration Platforms such as Hortonworks DataFlow, Apache Apex or Apache Beam;
- Routing can be performed with a number of Message-Queueing Middleware or a Publish-Subscribe Platform such as RabbitMQ, ActiveMQ, Apache Kafka and various Java Message Service (JMS) implementations;
- Processing is the scope of Distributed Stream Computing tools such as Apache Flink, Apache Storm, and Apache Spark Streaming or Event Processing Platforms, with higher-level

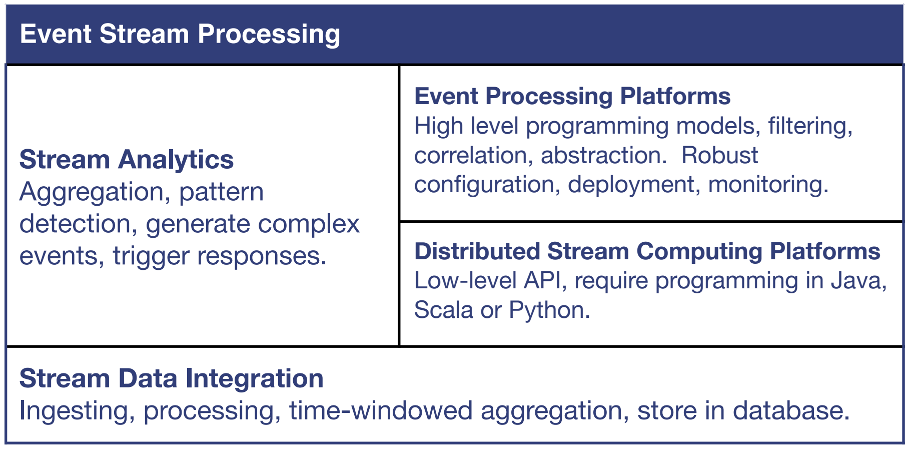
programming models such as Amazon Kinesis, Azure Stream Analytics, TIBCO Business Events, among others.

- Responding to events is the broadest category and accommodates any Event-Driven Application, or even a set of Function-as-a-Service (FaaS, or Serverless) event handlers built on platforms such as AWS Lambda, IBM OpenWhisk, or Microsoft Azure Functions.

Some specialized applications can also claim to be "event-driven," including Business Activity Monitoring (BAM) and Operational Intelligence Platforms. Although they really handle events similarly to EDA, they were built to fulfill specific requirements and are not that useful as general-purpose tools.

Event Stream Processing (ESP)

Event Stream Processing (ESP) can be considered an umbrella concept that contains both categories of stream analytics plus the stream data integration.



Event Stream Processing

Stream Analytics
Aggregation, pattern detection, generate complex events, trigger responses.

Event Processing Platforms
High level programming models, filtering, correlation, abstraction. Robust configuration, deployment, monitoring.

Distributed Stream Computing Platforms
Low-level API, require programming in Java, Scala or Python.

Stream Data Integration
Ingesting, processing, time-windowed aggregation, store in database.

Within ESP, Stream Analytics products are primarily used to trigger responses. Stream Data Integration is typically used to accumulate database repositories of events to support analytics queries. We are inherently more interested in Stream Analytics, triggering responses as an enabler for real-time responsiveness and situational awareness since these are key components of Digital Business.

Within Stream Analytics, Event Processing Platforms (EPP) tend to be more high-level and enterprise-ready. They are mostly offered as proprietary packages from commercial vendors who charge for licensing. Distributed Stream Computing Platforms (DSCP) mostly offer low-level API that are programmed from Java, Scala or Python. These products are typically open source and are better-suited to ISVs or software-savvy organizations willing to keep tighter control on their platforms.

In these architectures, event flows are implemented. Flows consist of a set of steps through which the events are collected from event sources, enhanced, analyzed and dispatched to event handlers.

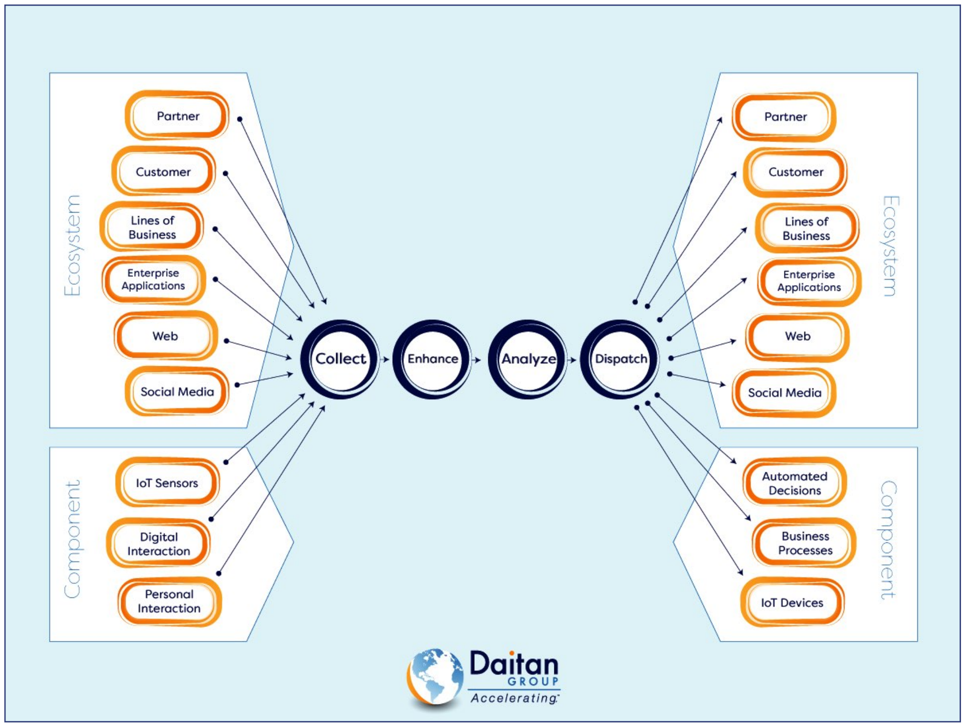
- **Collect:** From an ecosystem perspective, sources of events include partners, customers, lines of business, enterprise applications, web and social media. From a component perspective, sources include Internet of Things (IoT) sensors, digital interactions and customer interactions. All these sources generate events which are the inputs to ESP.

  REST web services using HTTP protocol are the de-facto standard for communicating synchronous events between applications. However, specialized message queueing protocols can be better alternatives for large scale Messaging or EDA. Examples include Advanced Message Queueing Protocol (AMQP) and Message Queue Telemetry Transport (MQTT). Latency can also be minimized by avoiding slow connections between message handling components.
- **Enhance**: A Stream Data Integration platform or other variant captures events. Events can be enhanced with situational information. Most obvious contributors are time and location, but other relevant measures include weather, traffic, economic indicators and calendar events.
- **Analyze:** Enhanced events are then analyzed. This includes event classification, recognizing patterns and gaining additional insight by aggregating groups of events. In the final step, enhanced, analyzed events trigger responses which are dispatched to event handlers.
- **Dispatch:** From an ecosystem perspective, the event handlers are similar to event sources listed above. From a component perspective, handlers are typically automated decisions, business processes and IoT devices.

Overall, event flows should be elastic because the rate of events is often highly variable and unpredictable. Serverless components and microservices are two approaches that support improved elasticity to respond to events.

When rates increase, more instances of these components can be rapidly launched and kept alive for periods of high demand. As demand abates, excess capacity is retired avoiding excess cost of under-utilized capacity.

Figure 6: Event Stream Processing From Event Sources to Event Handlers



Business and Revenue Models for ESP Products

The market for Event Stream Processing (ESP) is emerging, complex and experiencing rapid growth. ESP is growing because companies have more streaming data every year. Across the functional landscape of analytics and data integration, products are offered with four distinct licensing and revenue models.

Licensing and Revenue Models	
Proprietary	IBM, SAP, Software AG, TIBCO own almost 50% combined market share.
Pure open source	Numerous Apache projects based on Kafka, Samza and others.
Hybrid open source	Open Source with proprietary enhancements for specific functionality, performance or security.
Cloud	New entries from Amazon, Google, Microsoft.

Each model has it’s advantages. At this stage only the proprietary products are designed to facilitate use by Enterprises, at the cost of some flexibility. Pure open source tools work better for companies that have software engineering core competency and want maximum flexibility. Hybrid open source has advantages when a specific offering precisely matches company needs. Nascent cloud offerings could be an option for pilot projects where simplicity is key and cloud latency is not an immediate concern.

Considerations When Choosing an ESP

Before selecting the core of you EDA, we recommend carefully thinking about the following decision points. We are always available to discuss with our clients in more details.

Do You Really Need EDA?

Although EDA is widely applicable and can become a truly business differentiator in many cases, it is not recommended for all types of problems:

- In some situations, REST-based, synchronous communication is faster to implement and less expensive to maintain - this usually applies to simpler environments, with very few producers and consumers, not expected to change much or grow
- When on-the-fly data processing is not needed, but message routing can be complex, traditional Message Queue Middleware (MQM) tools can do the job of facilitating communication between multiple producers and consumers without the infrastructure needed for a full EDA implementation.
- When data needs to be quickly distributed to multiple consumers, but without complex routing rules, a scalable pub-sub tool such as Kafka alone can do the job.

Consider EDA when you need to do some processing on events, at high speed, and distribute it to a large or even unanticipated list of consumers.

Streaming Versus Micro-Batching

Some tools such as Apache Spark do not process single events individually. Instead, they group them in micro-batches which makes some operations more complicated, such as dealing with events received out of order, while restricting some time-window operations and increasing latency. It improves throughput, though.

Latency

Many factors influence event latency in EDA, but keep in mind that the tool of choice can easily become the most important of them. Apache

Storm, Apache Ignite and Apache Apex are known to have very low latency (to the order of a few dozen milliseconds in some scenarios).

However such a low latency is not the most common use case, and even Apache Spark (one of the highest latencies among Streaming tools) is suitable for many business use cases.

Delivery Guarantee

In order to improve throughput, many ESPs may deliver the same message to the consumer more than once, or “at least once”. Tools that only work like this include Apache Storm (without the Trident extension), Apache Ignite and Apache Samza.

Other tools can guarantee “exactly once” delivery such as Apache Beam, Apache Flink, Apache Apex and Apache Spark Streaming.

Some alternative communication methods, including a few MQMs, do not guarantee message delivery by default, but most ESPs do.

Out-of-Order Processing

Another important consideration is the ability to process events out-of-order, which Apache Apex and Apache Spark Streaming cannot do, for example.

Maturity and Community Size

When the tool of choice will be open source software (OSS), we always recommend taking a look at the community size, its growth rate and product version number. This is important for many reasons, including:

- Many good tools are still in versions smaller than 1.0. This means they are expected to change a lot in the short term, so avoid relying too much on their APIs and operations.
- A larger community usually means quicker bug fixes, more platforms supported, and guaranteed product evolution.

ESP-as-a-Service or OSS Tool?

Instead of installing an OSS tool into a cluster, some organizations may benefit from using an ESP as a Service, such as Amazon Kinesis. Although such services still cannot reach the performance of the major OSS ESP tools, they can be an option for organizations with more modest throughput that falls within the service limitations.[7]

This also applies to MQaaS, such as Amazon SQS. Our clients report good-enough results for applications that do not require low latency, and are especially happy with the absence of maintenance and cluster administration tasks.

[7] Amazon http://docs.aws.amazon.com/streams/latest/dev/service-sizes-and-limits.html

Popular Open Source ESPs

ESP is definitely a class of tools with many good OSS players available with different approaches to a processing unit (event or micro-batch), latency, throughput, delivery guarantee and so on. The most popular are:

Spark Streaming

samza

Apache Flink

APACHE STORM™

apache Ignite

beam

APEX

Preparing for EDA: How to Get Started

Migrating to Event-Driven Architecture (EDA) is a critical part of migrating to a future-ready Digital Business. Businesses should be mindful of these objectives and challenges:

- Assessment starts with identifying top business opportunities.
- Planning requires a multi-functional team that spans business, architecture and engineering management.
- Required architecture changes should be implemented first while minimizing disruption and regression testing to preserve stability.
- EDA can create new scaling and performance demands on existing consumer applications. Robustness should be proven before EDA goes live.

Assessing the Business Opportunities

As explained above, the first step is adopting the event-driven mindset and examining current flows of events. Organizations should look to expand event flow by incorporating new event flows and adding situational information to events. In addition, adopting an ecosystem focus will expose opportunities to publish events to partners and subscribe to events from partners.

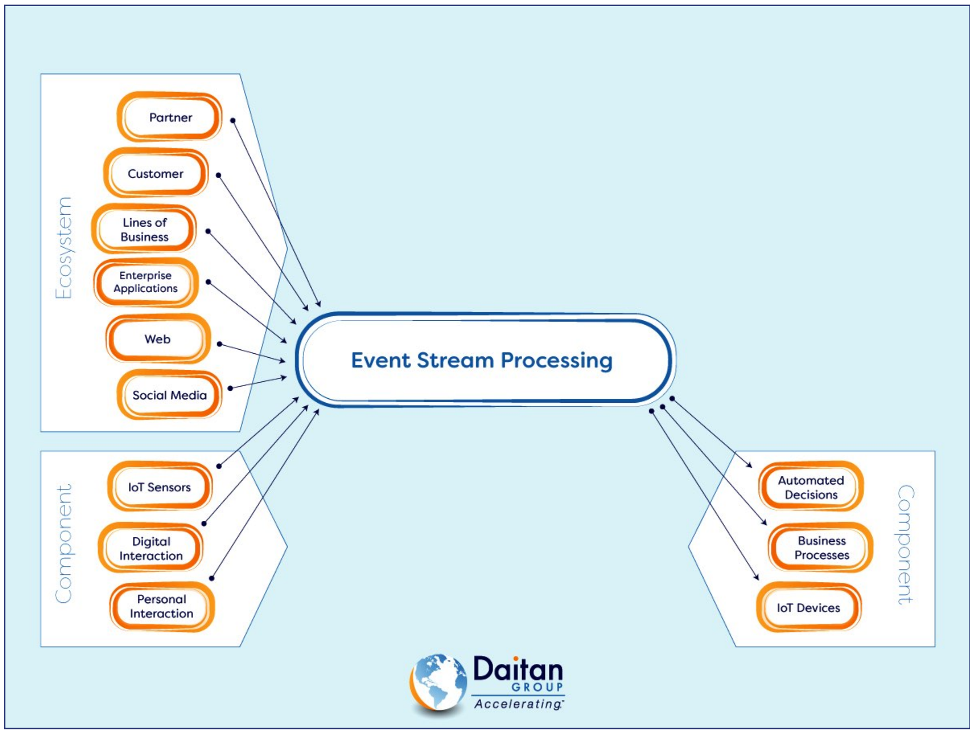
It's helpful to draw a map or diagram showing the sources of events, the contributors of situational information and the array of consumers or event handlers. External sources from partners (publications) and external handlers by partners (subscriptions) should be clearly identified. The results of event handling should also be shown. These are the tangible business benefits or opportunities from handling new and complex events.

The following diagram (Figure 7) represents an inventory of flows, situational enhancements, handlers and tangible business results. The final step of assessment is analyzing the inventory. The business results should be ranked and grouped by two factors:

- Importance should be based on various factors such as, new revenue potential, improving partner relationships, cementing customer loyalty and reducing costs.
- Complexity is based on impact or change required to architecture and Microservices.

The result of analysis is a pilot group of new flows and enhancements that enables new business opportunities with relatively low investment and lower risk of disrupting existing systems.

Figure 7: Event Stream Processing: Events from Event Sources are Directed to Consumer Components. This diagram can be used to identify potential contributors to Event-Driven Architecture.



Selective EDA-Enabling with a Pilot Project

As will be explained below, full-scale migration to EDA requires adopting new system-wide architecture and components. It also requires low

“Application leaders are finding that, in a growing number of cases, conventional application infrastructure architecture and technologies (such as DBMSs, business intelligence [BI] tools, message-oriented middleware and application servers) cannot effectively support the real-time responsiveness required for digital business moments.”

- ***How to Leverage Events to Empower Decisions in Business Real-Time*, Gartner 2017**

latency, high scalability, fault tolerance, new external gateways and asynchronous event handlers.

A selective pilot project is a good way to test the benefits of EDA and better understand the resources required for full adoption. A pilot project for selective EDA enabling could be limited as follows:

- A small set of event flows with high importance and low complexity.
- Limiting or throttling number of events to avoid system stress.
- Test runs of limited duration conducted during low traffic time periods.
- Events flow through existing interfaces despite being synchronous.
- Make allowances for higher latency as compared to scaled solution.

Adhering to these limits would minimize the cost of the pilot, reduce risk to infrastructure and accelerate progress to tangible results. It also keeps the overall effort simple. This allows the pilot team to focus on the business aspect of the pilot, measuring business benefits and extrapolating to scaled value over longer terms.

Software choices for the pilot can be simplified. Consider using cloud-based Event Processing Platforms (EPP). These can be launched quickly with a smaller investment. Keep in mind that latency can be higher and specific application functionality is limited. However, for a pilot, such tradeoffs can be acceptable.

Executing a pilot should provide a detailed business case for the selected Event-Driven Architecture. Once completed the cross-functional pilot team should have more expertise for evaluating broader benefit from more flows without the necessity of implementing additional pilots.

If the newly qualified opportunities warrant, the organization can embark on planning broader EDA adoption.

Full-Scale Migration to EDA: Planning to Avoid Risk

Organizations that have not implemented a well-defined Microservices architecture might want to consider that step before moving to full-scale EDA. Easy-to-scale Event Handlers are an important foundation for Event-Driven Architecture (EDA) migration.

A selective pilot will drive initial analysis identifying business benefits and engineering requirements. The pilot team will gain valuable experience adopting event thinking and valuable exposure to tools for implementing EDA.

More planning is required before starting full-scale migration to EDA. This should start with revisiting the full set of supported event flows (see Figure xx) developed when evaluating the business case.

These flows should have been ranked in importance versus complexity and grouped by area of system impact. If possible the ranking and grouping should be used to define multiple stages of EDA migration. The first such phase should prioritize enabling the most significant new business opportunities with relatively low investment and lower risk of disruption to existing systems.

The scope of the first phase should be evaluated. This includes the full collection of events and Microservices-based event responders invoked. With the scope known, the evaluation would be comprised of the following high-level steps:

1. Volumes and performance requirements
2. Microservices capacity planning
3. Event pipeline aggregation and analysis
4. End-to-end architecture

The scope of the first phase might be similar to a pilot project. The key difference is scale and robustness. In the pilot, various shortcuts and performance waivers are allowed. In this first phase, a robust, scaled, high-performing system is required — for limited scope.

Volumes and performance: This is focused on estimating quantitative measurements including:

- Total event volume
- Event volume per event flow
- Peak event rates
- Required success rate
- Maximum allowed responder latency
- Maximum allowed completion time

These requirements will influence the choice of Event Stream Processing and other components for collecting and dispatching events.

Microservices capacity planning: Responder performance requirements should be compared to current capacity for Microservices. This could require new infrastructure and methods for expanding baseline capacity and implementing elastic capacity for peak volumes.

Event pipeline aggregation and analysis: Event Stream Processing provides a pipeline for aggregating and analyzing events before dispatch to responders. This includes collecting outside information and aggregating to generate complex events with situational awareness.

Information sources should be identified and infrastructure requirements measured.

Dispatching events often requires complex analysis. This can include analyzing collections of time series events. Analysis requirements should be detailed and infrastructure requirements measured.

End-to-end architecture: Earlier diagrams in this document show the end-to-end EDA architecture from event collection to response. These diagrams can be augmented to show tools used for each step. Performance and capacity requirements can be labeled on each step of the diagrammed architecture.

Conclusion

Full-scale adoption of EDA requires a significant effort. The planning steps discussed in this paper should expose time and resource requirements. As planning progresses, requirements should be measured against benefits to decide priorities in migrating to Event-Driven Architecture.

Acknowledgements

We would like to acknowledge some of our engineers and managers for their contribution to this white paper: Claudio Oliveira, Leonardo Pereira, Rodrigo Zenji, Thiago Cangussu, Cleosson Souza, Walter Mesquita, João Pedro Perondini and Francisco Nascimento Ros.

About Daitan Group

Daitan Group provides high quality software development services to significantly accelerate time to market for global technology companies. The company's expert agile teams deliver full lifecycle software product development, maintenance and quality assurance services across today's leading technologies, including: cloud and virtualization; communications, collaboration, artificial intelligence, big data and analytics. For more information: http://www.daitangroup.com

DAITAN GROUP HEADQUARTERS
2410 CAMINO RAMON, SUITE 285 | SAN RAMON, CALIFORNIA, 94583 | USA