inrule

Rules for Simplifying the Internet of Things (IoT)

Rules for Simplifying the Internet of Things (IoT)

Rules for Simplifying the Internet of Things (IoT)

Introduction

According to Computer World, the term "Internet of Things" (IoT) first emerged in 1999[i]. However, by 2013, when this white paper was originally written, IoT had not yet become a staple in our everyday lexicon.

Instead, in 2013, the Internet of Things was a concept that those with a technical background were likely familiar with – but that was pretty much where the familiarity ended. Back then, a search of the term certainly wouldn't have yielded nearly 36 million results (Google search, 20170726: 35,900,000 results) as it does today.

Fast forward to 2017 and IoT has officially caught on. From the consumer space to the enterprise, IoT has created (and even raised) expectations for enabling new initiatives and markets, creating operational efficiencies, reducing costs, and improving customer experience and satisfaction.

In fact, Gartner estimated a 31 percent, year-over-year increase in "connected" devices and systems from 2016-2017[ii]. Consumer applications make up the largest segment of connected device users (63 percent of IoT applications), however "businesses are on pace to employ 3.1 billion connected things in 2017."

Enterprise use of IoT and connected devices is expected to grow in the coming years. According to Forbes, enterprise spending on IoT technologies, apps and solutions is expected to reach $267 billion by 2020[iii]. And McKinsey research predicts that B2B applications will account for nearly 70 percent of the value of IoT in the next 10 years[iv].

The Internet of Things and Complex Event Processing

Before IoT's current pervasiveness and reputation, enterprises turned to Complex Event Processing (CEP), IoT's slightly older cousin, to process notifications and data from multiple events to deliver actionable intelligence in real time or near-real time.

Over the years, the steady increase in devices and business processes created an explosion of real-time data. In 2013, the original draft of this paper was created to share InRule's approach to simplifying complex event processing through the use of rule technology. This paper outlined how InRule can help deliver situational awareness by correlating data to find its meaning and automating decisions to act on that data.

However, in recent years, the surge in demand for and deployment of IoT technologies has paved the way for an even greater need for event processing solutions. This is because event processing solutions can take in all the data from various IoT sensors and systems, swiftly process the data and provide a complete picture of a given situation.

Event processing tools empower organizations with situational awareness, allowing them to react quickly and make key decisions that can have a significant impact on the business.

Event Processing

Definition

An often-cited definition of Event Processing comes from Wikipedia under the definition of Complex Event Processing (CEP). In that definition, Complex Event Processing:

> "…consists in processing many events happening across all the layers of an organization, identifying the most meaningful events within the event cloud, analyzing their impact, and taking subsequent action in real time."

While CEP is probably the most commonly used term for describing these types of systems, its name is misleading. The word "complex" only serves to mystify something that is relatively straightforward. It is used to describe an event processing scenario where different types of events are analyzed **and correlated** at some level to pinpoint a scenario and take action on it. An event cloud refers to all the events that might exist within an organization's "programmatic" reach. For the purposes of this white paper we will use the definition above and simply call it Event Processing.

The many terms used to "define" event processing

Because event processing goes by different names and comes in different flavors, it makes sense to list some of the terms used with it. The following is a brief list of some commonly used terms:

- **Business Event Processing** - To make the CEP term friendlier, some software vendors have started calling it Business Event Processing. Unfortunately, Business Event Processing is perhaps too generic causing us to ask ourselves "What does 'business' mean in this context?"
- **Event Stream Processing (ESP)** – a term used to describe the set of technologies designed to assist the construction of event-driven systems.
- **Event Driven Architecture** – a software architecture pattern promoting the production, detection, consumption of, and reaction to events
- **Event Processing Engine** – place where the analysis and rules are applied to events. It is generally a process that hosts a capability to apply logic. It can come in the form of hard-coded logic or as a Commercial-Off-The-Shelf (COTS) technology. Typically, it is the latter as the latter usually gives the users the ability to manage the logic statements without programming effort. See *Typical architecture for an event processing solution* for a further definition.

Increasing importance of event processing

Event processing engines are not new. Events have been processed via "hard coded" systems for some time now. Financial service firms have used event-based decisioning for some time now to perform aspects of algorithmic trading, fraud detection, and surveillance. But those systems were very specific to the problem they solved and generally required expensive, specialized hardware and software that to both generate and process the events[v]. For those systems, it was worth the investment because of the risk (or reward) involved if these systems were not present. Today organizations from many different industries are starting to realize the risk of not having real-time event decisioning in their own environment.

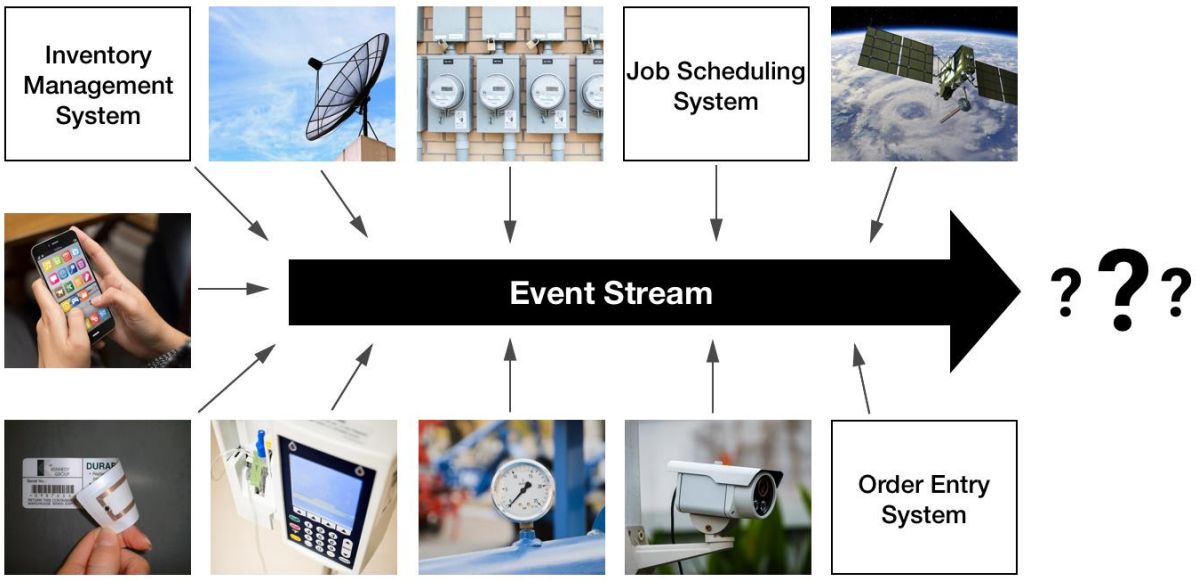

Figure 1 - What do all these events mean?

More, interesting data

Because most important organizational processes are automated or at least tracked via automated systems to some degree, there is a lot more data that can be analyzed and correlated to assess the business situation. Add to that the additional information originating from a myriad of devices - telephone systems, radar, smart meters, Radio-frequency identification (RFID) tags, pipe sensors, medical equipment, and even a highway ice sensor (see figure above) are all examples of devices creating data that can be acted upon. Consider that the modern-day cell phone alone has an accelerometer, a GPS, a proximity sensor, a compass, and a light sensor and you get the idea!

Opportunity in real time

For organizations today, event processing is becoming increasingly important because *they risk not having the operational intelligence and decisioning required to be competitive*. It's no longer acceptable to simply capture data as a transaction, store it, and then report on it to look for important business events. Often this approach can take days, if not weeks, to get data stored and then massaged into a format that can be reported against. At this point the important business event may have lost its relevance or taken a serious turn for the worse.

Time is money. Take, for example, credit card fraud. If I happen to use my credit card at a gas station in Illinois at 3:07 PM and another charge comes in at 3:31 PM for a retail store in California, then we know that there is fraudulent activity. Many credit card companies are monitoring for this type of event and put a hold on your card. If my credit card company was alerted days after the fraudulent events occurred, a lot of money could be lost. So in this case, early detection is best, as long as it does not result in a false positive. The same can be said for many events in your business. The sooner you can identify it and act on it, the better.

Without an event processing capability, there are very few systems that take a cross- functional view of events and monitor them in real time for situations where an organization can either reduce risk or seize upon opportunity. The expansion of available data and new kinds of sensor data creates an opportunity where unique operational intelligence can be derived.

Types of event processing

For the purposes of this document, we will only consider two types of event processing: Single (or Simple) Event Processing and Multi (or Complex) Event Processing. There can be many different derivations of these two types of event processing but for some degree of "simplicity" in this discussion, we will treat them as binary scenarios.

For the most part when doing event processing, you are either investigating a single event at a point in time and determining if it is something of significance on its own accord, or you are looking at a series of like or unlike events and determining if two or more of them have significance together. In either case, the identification of something of significance initiates some kind of downstream processing.

Single Event Processing

Figure 2 - Single Event Processing

With single event processing, a source generates an event and that event is sent to an event processing engine for analysis. The Event Processing Engine is where the "thinking" happens. Here analytics and rules are applied seeking one or more true conditions. If a true condition is discovered, an action is taken. Single event processing is stateless, meaning the Event Processing Engine does not keep information about the event for future event processing. Single event processing only deals with one type of event.

For example, consider a hospital system where RFID tags are used to track all aspects of medical equipment, medicine, drugs, etc. A simple event system might be used to alert hospital staff if penicillin was brought into a room with a patient who is allergic to that antibiotic. The source is the RFID sensor in the patient's room and the event is penicillin entering Room A. The Event Processing Engine contains the logic to check allergies for the patient in Room A (Patient B) and then send a downstream message to take some action (in this case, send a warning).

Multi-Event Processing (aka CEP)

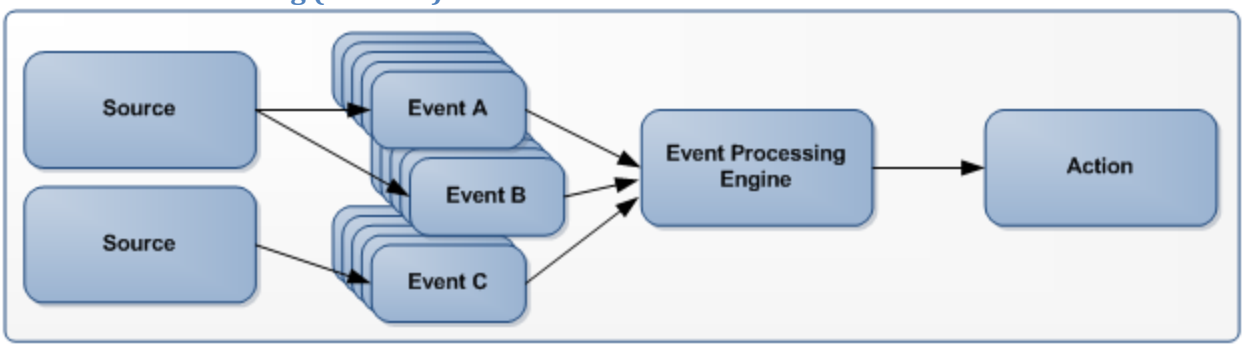

Figure 3 - Multi-Event Processing

There are many times when IT plays a central role in critical, even truly "mission critical" situations. Consider the following:

> *Over the course of an hour, ICU monitors detect a subtle but growing change in a patient's condition. When a tolerance level is exceeded, a number of events take place in rapid succession: a change in medication is "prescribed" via an alert sent to the nurse's station; provisioning of the necessary medication is automated and it arrives within minutes. The attending physician gets an alert, an RFID tag on the med is correlated with the alert and the patient's electronic medical record is updated with the physician's approval and the administering of the medication. Within minutes, the monitors record an improvement and the alert is closed.*

All this took place because of an effective yet complex event processing system.

With multi-event processing, one or more sources generate two or more events to Event Processing Engine. The Event Processing Engine in this case generally seeks to find a pattern in the stream of data by applying analytics, aggregation, and/or correlation algorithms. Because events do not necessarily arrive at the same time, the event processing engine is stateful, meaning it tracks information about previous events so that information can be applied to future events as part of the pattern detection process. Almost always more than one event type is involved although this is not a requirement.

Another distinguishing aspect of multi-event processing is that it often seeks to correlate the events over time and space. For spatial correlation, GPS sensors often used to give an object's location over time.

A number of financial firms use multi-event processing engines for their algorithmic trading systems. A very simple scenario might be to monitor the stock price of two companies who are in the same industry. Since they are in the same industry, movement in the price of one stock should have some correlation to movement in the other. So if the price of IBM goes up $1.00 and the price of MSFT goes up $.30 in a five minute period, then buy Oracle.

Many CEP vendors claim that a "CEP problem" is one where low latency and high-volume processing are a requirement. Unfortunately, low latency and high volume are very relative depending on the industry and problem. In financial markets, this is certainly the case. The ability to process thousands of transactions per second is a requirement. For other industries, this is not the case. One of the goals of this paper is to give readers an understanding of ways in which event processing can be applied to other industry problems. The next section explores single and multi-event processing examples where other industries are taking advantage of event processing.

Solutions using InRule® for event processing today

Chances are if you are reading this white paper, it's because you've recognized you have a need to process events in real time using rules. So far, we have been talking about where events come from and how they can be aggregated. Regardless of how they get there, the decision of what to do is always based on rules. Event processing with InRule is composed of two pieces: event collection and the application of logic. The following section describes how four solutions utilize InRule for event processing.

Use case: Real-time data center monitoring

A large systems integrator is using InRule to monitor customer-hosted systems and applications in a worldwide solution.

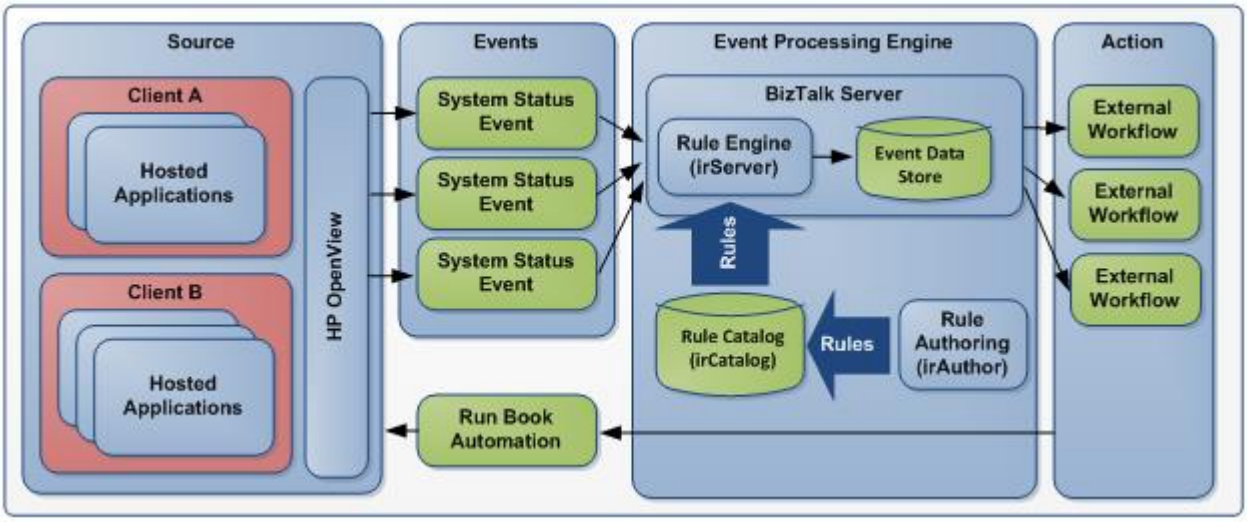

Figure 4 – Data Center Event Monitoring Solution

From the figure above you can see the Client Hosted Apps with HP OpenView (HP BTO Business Technology Optimization software) act as the source of the event data. System events are sent into the Event Processing Engine, in this case InRule, hosted on a set of BizTalk servers. A team of business analysts in Europe manage correlation rules (i.e. the event processing logic) that are both product- and customer-specific. As events come in, rules stored in the rule catalog are executed to correlate the application events over time and identify potential downstream system outages. The system is architected to 1000s of events per second coming from all over the world in real time. A simple example of a rule is listed below:

```
If
    3 or more event messages contains "cpu utilization exceeds threshold" for the same
    machine and location within 300 minutes »
Then
    "More than 3 events for high CPU utilization were generated from same
    customer, datacenter in less than 5 hours." ^ Please investigate.
    [add action]
```

Figure 5 - Example data center situation rule

Using rule technology in this way provides a way to assess the *data center situation* and take proactive measures when an issue occurs.

Use case: Early medical condition identification

A large healthcare coverage and care provider uses InRule for real-time medical condition identification. In this system, events are sourced from doctors' notes, hospital claims, pharmacy records, clinic records, lab results, and elective patient information. The event processing engine then runs rules to correlate over 2.5 million events per day to create a patient's real-time medical situation.

In this approach, early detection indicators point to potential downstream medical conditions. Doctors and patients are notified of candidates as they are moved into higher risk populations. When a patient is included into a risk population the system can also conditionally trigger the scheduling of tests and clinic appointments to address the situation as quickly as possible.

```
If
    Event is "DEXA Screening Test" »
     and   there is no "Osteoporosis Exclusion" in the last 180 days »
Then
    set Target Population to "Osteoporosis Management"
    set TargetStatus to "Active"
    Evaluate population membership reassessing candidates declined more than one year ago
    [add action]
```

Figure 6 - Example medical condition identification rule

Using InRule for event processing across a patient's routine results from physicals and doctor visits, this system detects potential disease risks and automatically notifies the patient and their primary care physician.

Getting a real-time assessment of the *medical situation* in this way has historically demonstrated a dramatic reduction in medical costs and dramatically reduced disruptions in quality of life.

Use case: Security device monitoring

A leading provider of physical security information management (PSIM) solutions implemented InRule in its software for various event processing scenarios. Events originate from a wide variety of sensors (including, but not limited to, fire alarms, cameras, motion detectors, GPS, video analytics, etc.) and data sources. The event processing engine analyzes and correlates the various events to help create "an intelligent, geographically based, shared operating picture and facilitates the management of numerous events of a large city."

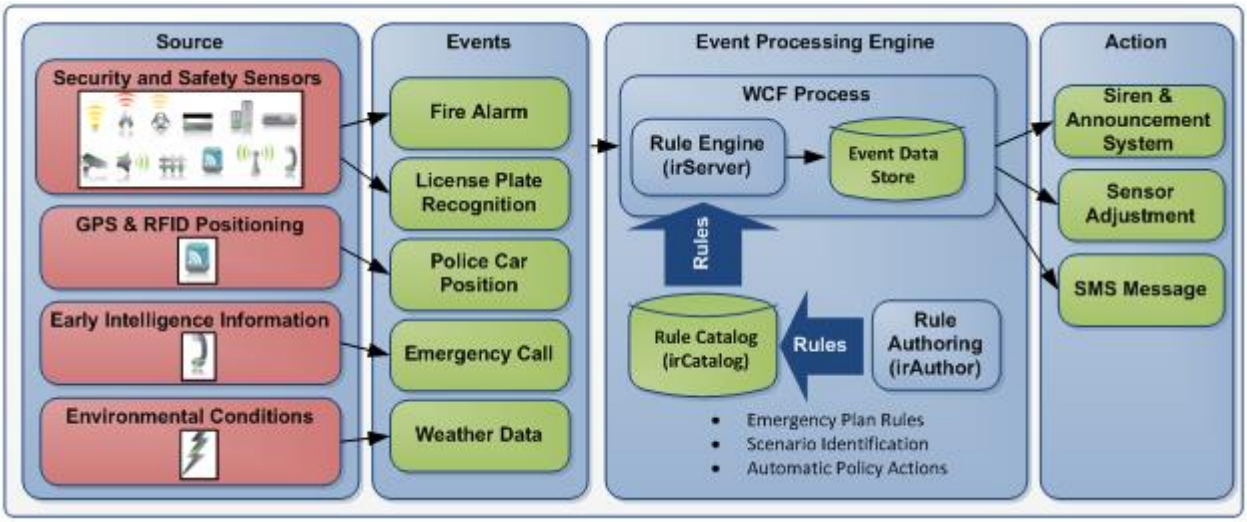

Figure 7 - City monitoring solution

Automatically identifying the correlation between different events, the rules engine enables operators to solve an event in the most efficient manner, while giving managers the flexibility to pre-define responses according to the organization's policy.

Depending on the *security situation*, the engine initiates context-based automated and semi-automated predefined responses to a range of routine and emergency situations.

Use case: PBX monitoring

In telecommunication systems, there is a growing need to analyze call records and detect PBX hacking and misuse. Increasingly companies want to identify when:

- the same employee regularly calls the same number within the same time range
- an outsider regularly calls within the same time range
- a sales person makes fewer phone calls than expected
- excessive overseas call are made

In a paper published for the Advanced International Conference on Telecommunications[vi], a solution was presented for the monitoring of telecommunications systems using an event processing approach with InRule. Events are sourced from an open-source PBX system called Asterisk and sent to an event processing engine running InRule. Call events are correlated and evaluated against rules to determine if a call should be allowed to pass through. In this case, the event type is the same; however, temporal reasoning is applied to correlate the call events over time.

A data-mining package or any other analytical tool would not be as efficient because the *telecommunication situation* would be reported when it is too late to take action.

High Level Architecture

Anatomy of an event processing solution

As we've seen from the definition and solutions described above, event processing solutions generally share the following characteristics:

- Event information is processed by the rule engine in real time
- Decision logic can be based on a single event or on combinations of multiple events and event types over time
- Rules are used to interpret events and identify situations
- Once a situation is identified, action is taken in real time

It should be noted that a subset of solutions require high-volume, low-latency processing (i.e. process 1000s events/sec) like those found in capital markets. However, many event processing solutions do not require this level of throughput and responses need not be measured in milliseconds. Often responses measuring in seconds are adequate.

Typical architecture for an event processing solution system

The standard architecture for event processing is described below. It often refers to an Event-Driven Architecture (EDA) and is the basis for most event processing solutions.

The four logical layers are as follows:

- **Event Generators** - Every event is generated from a source. The source might be an application, data store, service, business process, transmitter, sensor, or collaboration tool (IM, email). Many "platform" applications like SQL Server, Salesforce, Oracle, Microsoft Dynamics, Microsoft Azure service bus and other enterprise service bus technologies (ESBs), etc. can be used to source events. The power of IoT can also be seen when events cause a change in state over a period of time that are of interest, such as when a temperature, measurement or other value rises or drops below a specified amount.
- **Event Channel** - The event channel, typically a messaging backbone, transports standard formatted events between event generators, event processing engines, and downstream subscribers.
- **Event Processing Engine** - In the event processing layer, upon receipt, events are evaluated against event processing rules, and actions are initiated. The event processing rules and actions are defined in accordance to the needs of the interested parties, not of the event generators.
- **Downstream Event Activity** – A single event, or event correlation, may initiate numerous downstream activities. The invocation of the activity might be a push by the event processing engine (service invocation, business process initiation, notification) or a pull by subscribers of event publications.

Plugging InRule into your event processing system

Depending on your environment, you probably already have an Event Driven Architecture in place. Enterprise Service Bus (ESBs) products like Mulesoft, Neudesic's Neuron ESB, or nServiceBus may already contain the events you are planning to run through an Event Processing Engine. If that is the case, InRule can easily be plugged in the Event Processing Engine layer of the EDA to add operational intelligence to your processing. If not, your own EDA can be easily constructed with a homegrown set of listeners, complimentary processes, and InRule.

The figure below shows how an Event Driven Architecture can be constructed using a combination of third-party products, your applications, and InRule. Filtering, aggregation, correlation, and other rules are authored using InRule's out-of-the-box authoring tool and stored in the Rule Catalog. Events are then sourced from a variety of places and channeled into a host that invokes InRule. InRule runs the rules authored by developer, analysts or subject matter experts and determines which actions to take.

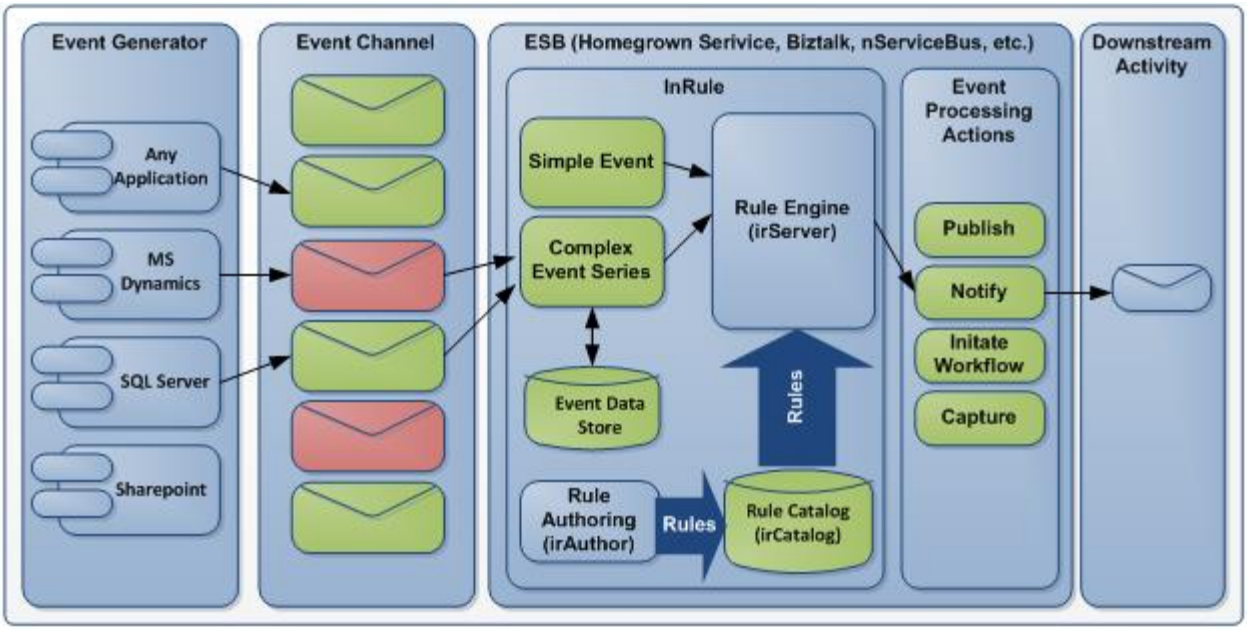

Figure 8 - Event Driven Architecture with InRule

While there are a number of event processing (aka CEP) products on the market today, they are often highly specialized for a particular solution (like algorithmic trading) and have weak authoring capabilities for business-oriented users. The following section explores some of the advantages of using InRule for event processing.

Sophisticated and intuitive authoring (analyst-driven) capability

It's important that rule authors can clearly express their intent and that they are insulated from the technical underpinnings of the implementation. Often event processing products have specialized event processing languages and tools that are difficult to use for business analysts. In some cases, vendors have implemented graphical modeling capabilities, but these too are based on arbitrary approaches. Implementation of a specialized event processing language is closer to SQL than it is to a more familiar logic statement.

Authoring for the desktop

For most users, InRule's out-of-the-box, authoring tool called irAuthor is used for the development and management of event logic. For those users who want to author from the context of their own application or from the web, InRule authoring can be embedded in third-party products and provide a way to surface its complete set of authoring controls. Like the security device monitoring solution described above, users write rules directly from their software.

Proven authoring environment

InRule has a proven authoring environment consistently used by business analysts to express business logic. In addition to over 100 built-in functions, analysts can choose to write logic using an Excel-like syntax language for calculations, Business Language for conditional statements, or a decision table for matrix-oriented logic.

Robust vocabulary and business language template development

InRule also features the ability for users to create their own vocabulary, classifications, and templates to provide a more domain-specific authoring experience. Poorly named entities and fields that were

imported from external schemas can be aliased to provide better authoring clarity. Classifications can be used to label a business condition with a meaningful name for use in other rule conditions. Finally, templates can be created to abstract repeated logic patterns so they can be reused across rulesets.

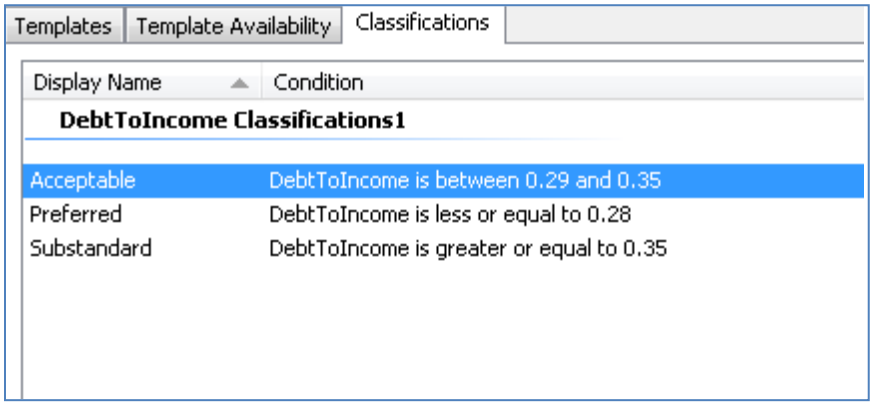

Figure 9 - Defining Classifications

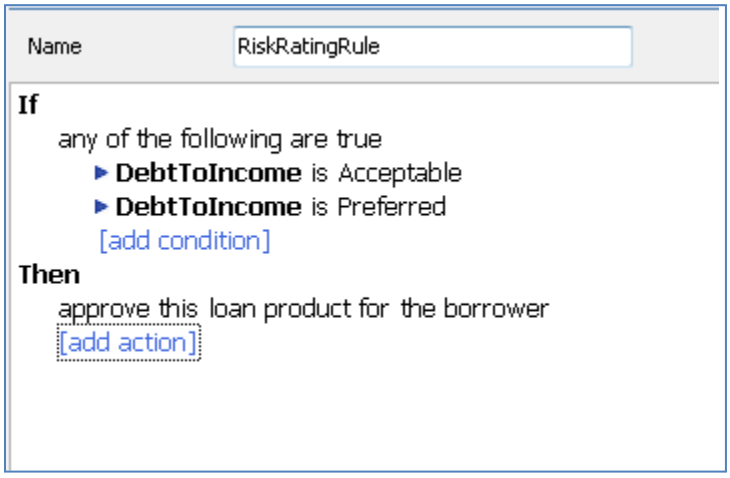

Figure 10 - Using Classifications in a Business Language Rule

Flexible schema definition

In event processing, sourcing and structuring the event schema can be a challenge. Because event processing often involves executing logic over disparate data sources and event types, it is important to have the ability to both to define schemas in an ad-hoc manner and leverage external schema definitions.

InRule allows you to define or source schema four different ways:

- Ad-Hoc
- From an XSD
- From a .NET Assembly
- From a database

Events that are sourced maintain the hierarchical structure of the schema definition. Data points are not "flattened out" or two-dimensional. Collections and other complex types maintain their relationship to

their parents, peers, and children giving authors access to data points in a context that is a closer representation of the real-world event they are evaluating.

Finally, additional, temporary data points can be added to the data structure for information derived from calculations or rules.

Ability to call external data stores for additional lookup information

It is not reasonable to expect that a given event type will contain all the information necessary to filter, analyze, aggregate, and correlate an event. Unfortunately, most event processing platforms don't provide an easy way to access external data to support the event decisioning process. Programming code needs to be written in the form of an adapter to access information contained in one or more external systems or processes.

With InRule, reference data is accessible from any number of sources. Authors can access functionality and data via external .NET Assemblies, database lookups, web service calls, and file lookups.

Rich set of functions

In the real-time application event monitoring solution above many of the rules searched unstructured text for meaningful phrases in the system event. Because there is no standard for error messages generated by a device in a data center environment, the message that says a disk drive is failing can be written any number of ways. Having the appropriate, built-in functions perform a search across this unstructured data was crucial for business analysts to properly investigate the data center situation.

InRule has over 100 built-in functions for text searching, time comparisons, finance calculations, etc.

Implementation Approaches

There are two main approaches for structuring an event processing engine using InRule:

- Keeping all relevant events in memory;
- Storing events in a database and selecting and processing those needed for analysis.

Both have their advantages and disadvantages, which are detailed below.

Keeping relevant transactions in memory

When processing a single event for filtering, analysis, routing, transformation, etc., this is the only approach you need to consider. The event comes in, it is processed against the authored business logic, an action may or may not be taken, and the event is discarded. The event only remains in memory during the process of evaluating the event. There are no real disadvantages using this approach when processing a single event at a time.

When multiple events are needed for decisions that require aggregation or correlated decisions, the event and a rolling history of events are kept in memory throughout the life of the entire process. This means that as each new event is presented to the engine, it is stored in memory and kept there even after the event instance is processed.

The following is a high-level sequence of steps the Event Processing Engine would take using this approach:

1. Event is received from source or listener.
2. Event is transformed, if necessary.

3. Event is filtered, if necessary. In some cases, it can be determined that the event isn't relevant and can be discarded early to avoid unnecessary processing.
4. Event is added to the event history collection in memory.
5. Aggregated values are calculated using specified filter criteria.
6. Authored rules are executed against stream of events looking for correlations.
7. Actions are taken when specified conditions are true (e.g. send notification message, execute SQL query, etc.).
8. Periodically remove items from history that don't fall into windows of processing.

An advantage of this approach is that it is less CPU-intensive because events remain loaded in memory and therefore require less time to load and unload the collection of event history for each event instance. A disadvantage of this approach is that it's memory intensive – so there is a limit to the number of events and events streams that can be managed. In addition, events are not necessarily persisted so the process may not be able to rebuild the "situation" if an outage occurs. If the number of events that you need to correlate over time or some fixed constant is relatively low, you should choose the in-memory approach.

Storing relevant transactions in a database

As the number of events being considered by an Event Processing Engine increases, it passes a threshold where a dynamic persistence mechanism needs to be used. In fact, many CEP engines use relational databases "under the hood" as their backend for storing events and then running specialized database queries against this data to do their analysis and correlation. Other CEP implementations use proprietary data stores to ensure reliable recovery in the case of a system outage.

Customers with larger volumes of events have implemented InRule in much the same way. By storing the event data in a database, the process hosting InRule can select a window of events to consider based on time or an arbitrary number. Much like a CEP engine, the time window or arbitrary number of events can be derived from the correlation logic. These are loaded in real time into the event window at which point the analysis and correlation rules run.

The advantage of this approach is a larger number of events can be handled because the various event channels are stored in the database and retrieved as needed. In addition, the event data is persisted so it can easily be restored or replayed at a point in time. The disadvantage is it is more CPU intensive. Not only does it need to retrieve and assimilate the desired event window "on the fly", it also cannot reuse existing derived data as was the case in the in-memory option. Previous instances of events are thrown away with each new event.

Looking to the event processing horizon

It would appear we just now understand the full potential of Decision Event Processing along with many of the requirements, if not challenges As we've seen from the InRule use case examples, there are a number of different industries and solutions where InRule can used as part of an Event-Driven Architecture. InRule can be plugged in as the Event Processing Engine to provide a robust, flexible solution for the analysis and correlation of events. Moving forward, InRule intends on furthering its event processing capabilities. The following are some of those features and functions:

- **More flexible engine architecture** – currently InRule can run rules against event data in XML streams, JSON and other data formats. While these alone provide many options when architecting a system, it's clear there is a need to be able to operate against data stored in other formats including persisted data. Future InRule engines will make it easier to add new kinds of state including events stored in a database or on the file system.
- **New ways to source events** – Many products on the market are now capable of configuration through their APIs and these can be great sources for events. Being able to reach into these products and source event types and automatically build workflows and triggers that will send events to a centralized event processing engine at run-time would be a huge gain in productivity. InRule would be used to configure sources for various event channels and orchestrate the execution of the analysis and correlation rules on the event streams.
- **Non-technical setup** – Many of the products mentioned above are increasingly getting configured by non-technical personnel, especially in small-to-medium sized enterprises. Giving users the ability to dynamically setup an event processing scenario without programmer involvement would put the business in control of assessing any number operational intelligence situations.

Summary

There is a growing need with all sizes of organizations and at all levels within the organization to understand "the situation." More data is coming online from many different sources and provided it is assimilated, it can provide businesses with important situational awareness. Event processing enables organizations to simulate various events real-time to gain a level of operational intelligence not attainable through previous methods like BI and reporting tools.

While financial services could be considered the gold standard for the application of complex event processing, many people incorrectly assume that all event processing problems require an extreme low-latency, high volume capability. This is not the case and many event processing solutions compromise simplicity and ease of use in favor of achieving millisecond timeframes. Certainly this is valid for many financial service applications like algorithmic trading, risk assessment, and fraud detection where 1000s of events must be processed in a short amount of time. But these tools have been criticized by analyst firms for their lack of business-friendly tools.

InRule provides the capability to build an event-driven architecture that simplifies the authoring process. InRule delivers a set of business-friendly tools where non-technical people can author the rules that analyze and correlate business events over time. Many event processing problems don't require extreme low-latency or extreme high volume. It is relative to the industry. In the case of the data center event processing solution, responses within a few seconds are perfectly acceptable. In many cases where InRule is used for event processing, response times are still sub-second. In addition, InRule provides other advantages such as the ability to handle complex data structures, comprehensive policy and rule management, access to external data, etc. Finally, InRule can be used in other traditional business rule engine applications like pricing, eligibility, and scoring, thereby letting organizations centralize their event processing and transaction application logic in one tool.

Using InRule for event processing can help organizations sift out meaningful information from background noise and give "in the moment" meaning to what matters.

Imagine the upside when subject matter experts in scenarios such as those highlighted here could author and update the rules that allow effective processing of complex events and automate decisions, without hard coding. **Complex becomes simplified.**

---

[i] Article: http://www.computerworld.com/article/3076213/internet-of-things/the-iot-past-present-and-future-an-interview-with-professor-sanjay-sarma.html

[ii] Announcement: http://www.gartner.com/newsroom/id/3598917

[iii] Article: https://www.forbes.com/forbes/welcome/?toURL=https://www.forbes.com/sites/louiscolumbus/2017/01/29/internet-of-things-market-to-reach-267b-by-2020

[iv] Article: http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/an-executives-guide-to-the-internet-of-things

[v] It should be noted that many of these systems have since migrated to a commercial off-the-shelf product for processing events.

[vi] Article: http://www.computer.org/portal/web/csdl/doi/10.1109/AICT.2010.93