# Architecting Tomorrow: a Vision on Event-Driven Architecture

Smals Research

Koen Vanderkimpen

Nov 9 2023



### **Smals Research 2023**





### Agenda

- Start with: WHY?
- EDA 101: What is Event Driven Architecture?
- An Evolving IT Architecture for the Future
- EDA 201: Some Advanced Concepts
- EDA benefits for Business and IT
- How to start using EDA: Approach and Mind-Set Shift
- Conclusions





## Let's imagine our WHY

Some men see things as they are, and say why.

I dream of things that never were, and say why not.

– Robert Kennedy

- My Vision
- eGov 3.0
- Why EDA; Why Now ?

### **A Perfect Storm**

- Debt, Inflation, Stagflation, Financial Crises, ...
- Income Inequality
- Globalization/ Mobility of Workers & Capital
- Deglobalization / Protectionism / Political Extremism
- Aging Populace, Retiring Baby Boomers
- Global Health Emergencies, Pandemics
- Transformative Climate Change
- the AI revolution, Workplace Automation

Ten Dangerous Trends That Imperil Our Future, and How to Survive Them

### T **MEGATHREATS NOURIEL ROUBINI** BESTSELLING AUTHOR OF CRISIS ECONOMICS

(Book for sale in most book stores; Not the exact same threats)



Nov 09, 2023 Event Driven Architecture

### What's it to Us?

What can we do, as the architects and builders of the eGovernment?

- Simply put: support policy, and make sure we are ready for anything!

More specifically:

Let's build a machine, capable of efficiently automating what we know and quickly adapting to what we don't know.



Iov 09, 2023 Event Driven Architecture

### My vision: Imagine... IT as the Agile Automator

In my vision, the administration, and its IT systems, get out of citizens' and companies' ways, as an **invisible force** that handles all the data and acts accordingly, with minimal human intervention. Whenever something changes in the real world, through actual events in people's lives, or by changing the rules by which the administration has to play, this system **easily adapts**, with minimal maintenance.

Key Principles: agility, automation, resiliency, once only, no wrong door Main take-away: EDA will be an important part of the puzzle ;-)

This is the idealistic; now let's get practical



Nov 09, 2023 Event Driven Architecture

## What does that mean, Practically? (1)

- **Resiliency**: (in the strict sense of the word) continuity (and self repair) in the presence of failures, even stronger than High Availability. But one could also include: very agile.
- Agility: Resiliency to required changes at the functional level. The greater the ease with which we can adapt the system's behaviour, and the smaller the impact of required change, the higher the agility.
- Automation: At business level (e.g. getting certain rights automatically) and at IT level (DevOps, CI/CD).



# What does that mean, Practically? (2)

- Once only: you give your data to the government only once. It then travels wherever it is needed. *IMPORTANT*: this does not mean we retain it only once! We are allowed to make copies (for technical or legal reasons).
- No wrong door: it does not matter which channel is used by the citizen/company to give the necessary data to the government.



### Looking ahead to How: *Modularity*





LOW COHESION HIGH COUPLING HIGH COHESION



### The eGov 3.0 Vision

Pure Pragmatism can't imagine a bold future.

Pure idealism can't get anything done.

It's when the two cooperate that magic happens.

– Simon Sinek

eGov 3.0: the Groundwork for the near and far future Created by the leaders of our institutions of Social Security

**Comparable to my own vision** (but more practical)



#### E-GOV 3.0

Naar een toekomstbestendige digitale sociale zekerheid



### The eGov 3.0 Vision

A joint strategy for the future, driven by the joint institutions of Social Security (OISZ/IPSS), together with a number of stakeholders.

- Short Explanation
- Why EDA?
- eGov 3.0 EDA Exploration Workgroup





### What does it entail, in short?

Some statements: ... How to ensure a single business solution based on a common vision, in a **future proof** ecosystem context (adaptable, expandable)? ... Agility and flexibility are a conditio sine qua non. ... ... The main goal is to redesign digital social security; to actualize and future proof its digital systems, ...

8 Primary Directives (krachtlijnen / lignes directrices):



. Better unique data model, transparancy & autonomy

4. Across jurisdictional levels

6. Real Time sharing & First Time Right decisions

3. Enterprise Architecture "going forward"



(paraphrasing to english & emphasis are mine)

Nov 09, 2023 Event Driven Architecture

### Why EDA for eGov 3.0?

- EDA can help with several of the directives: *real-time, automatically granted rights, futureproof architecture, across jurisdictional levels*. Perhaps even more?
- EDA is referred to in the document:

The information should preferably be available in nearly **real-time** to allow for first time right decisions ... Applications are developed to be used **maximally Event-Driven**. Information systems can both use the push and pull principles. ..., we can, in the future, use **a more event-driven approach** instead of an action driven one (e.g. "in situation ... we get the right ..." instead of "if you want the right, you should apply for it") ... Concepts such as **asynchronous communication and APIs, EDA, eventual consistency, reactive** design, ... can all contribute towards a higher availability and more easily adaptable systems.



### eGov 3.0 Exploration Workgroup: EDA

Elementary questions (WHY, WHAT, WHEN, WHO) about using EDA in OISZ

<u>Justification</u> Current architecture paradigms are at their limits for the ambitions of eGov 3.0, so the event-driven architecture becomes necessary	<b><u>Goal</u></b> Map what needs to be done for a shift towards using event-driven architecture
<ul> <li>Scope</li> <li>Map EDA impact on <ul> <li>architecture</li> <li>business/organisation/processes</li> </ul> </li> <li>SWOT analysis EDA</li> <li>Requirements to start using EDA</li> </ul>	<ul> <li>Output</li> <li>Guidelines for using EDA <ul> <li>SWOT EDA</li> <li>When to use</li> <li>When not</li> </ul> </li> <li>Guidelines to analyse projects</li> <li>Approach on improving EDA knowledge @ OISZ</li> </ul>

## Why EDA, in general?

- APIs are really good, but they don't cover all types of **Integration**
- EDA offers increased Agility, Modularity & Resiliency
- **Real-time** Interactions & Insigths: Value of Information decreases with Time!

After Critical Mass attained:

- Cost Savings
- Faster Time to Market

#### Gartner:

"The World is Event Driven

Are your Business Systems Event-Aware?



### Why EDA Now?

EDA is nothing new, really

• Publish/Subscribe mechanism and Message queues have existed for decades

And *yet*: IT industry has really started focusing on EDA the **last 5-7 years** 

 Might be, in part, a reaction to overly connected microservices architecture, and also the increase in coupling between systems caused by RESTful APIs



17

According to Gartner, this year, 2023, "over 50% of business organizations will participate in event-driven digital business ecosystems" and "most leading providers of application platforms will include high-productivity tools for event-driven design"

Recent reply by Gartner's leading EDA analyst: "There is a lot more adoption and the **need for continuous intelligence** is more important. Also the popularity of Kafka." – Yefim Natis



### **EDA 101: What is Event Driven Architecture?**

Life is more than sunglasses and hit movies. Reality - that's the main Event.

– Sylvester Stallone





### EDA 101

- A Different way to Communicate / Integrate
- A Different way to Represent Reality
- Definitions
- Technology, Community



### A different mode of Communication

**Synchronous Communication** 



Asynchronous Communication



Two-way

waiting for immediate reply

One way not waiting for reply



### Add <u>one-to-many</u> and you have... EDA

### **Direct Call**



### direct, synchronous two-directional, *1 receiver*

### **Event Send and Receive**



### indirect, asynchronous unidirectional, *many receivers*



Photo: Jeff Geerling https://www.flickr.com/photos/lifeisaprayer/4592438980: CC BY-NC-SA 2.0

### **APIs: the Golden Hammer**

The **RESTful API standard** remains the standard way of Integrating systems... synchronously

But they are so useful they blind us to other options

"With a good hammer, every problem looks like a nail..."

Often, a screw driver would be better

Let's consider a FULL integration strategy, giving equal importance to EDA



### **A closer way to represent Reality**



### **Business Events: the prime drivers of Data**

Currently, all data is derived from Events in the real world.

Representing that data as the actual Event brings the IT system closer to the real world.

I believe we should keep it in that form as long as possible, and only **transform it to "regular" data when a business process effectively needs to** act on it.

Keep these Events around... you never know when it would be useful to have a look at where your data came from.



### **Event: a Definition**

- An Event in the real world (technical events also exist)
- ... captured by an IT system

("payload")

- Given a type and a timestamp
- Often with extra data about the Event

begov.rsz.work.salary.SalaryPayed
Time:2022.04.01-15:43:21:321456
Employer: 0406.798.006
Employee: 881122.555.77
PaymentDate: 2022.03.31
Bruto: 8000
Netto: 4000
WorkPeriod: {...}
...



### **Event Driven Architecture**

Software Architecture Paradigm, centering on the

creation, transmission, and handling of business Events



# **Basics: Publish/Subscribe mechanism**



### **EDA Technology**

- Event Broker (Apache Kafka, Azure Event Grid, ... )
- Event Definition Standards (AsyncAPI, CloudEvents)
- Libraries & Frameworks (AxonIQ, Dapr, ...)
   AxonIQ
- Overarching Integration Suites (Solace PubSub+ "event mesh")
- API Gateways with EDA support: Event Driven APIs (Gravitee, Axway Amplify Streams, ...)
- Auxiliary concepts (webhooks, SSE, ...)

...



gravitee.io

Solace PubSub+



### **EDA Community**

https://www.eda.community/

The Event Driven Manifesto

- <u>https://cymo.eu/blog/eda-manifesto</u>
  - 1. Think Business First
  - 4. Topics and Event schemas define the contracts
  - 8. Stay Focused on the end-to-end Process





### Zooming out again... Let's talk About IT Architecture!

There's a way to do it better.

Find it.

– Thomas Edison

### How can EDA help to modularize

and make IT more future-proof?

Smals



# IT architecture: The "HOW"

- 3 Layer Architecture and the introduction of APIs
- Distributed Systems and the CAP theorem
- Layered Ecosystems
- EDA in the Service Layer, Eventual Consistency
- Hexagonal Design, Domain Driven Design and Event Storming
- What about microservices?



### Are you building with 3 tiers?



- Increasing **modularity**
- The traditional architecture that came to be when people were sick of spaghetti code
- A good start: these layers have very different responsabilities (concerns)

- "Separation of Concerns"
- Higher cohesion, coupling a bit lower



### Some problems with 3 tiers...



- Overreaching through the layers
- Uncontrolled External Access ("wild integrations")

?

Integration by Database (that includes batch !)



### **Enter the API**



- Rule: applications can only be accessed by their
   API
- Only higher layers can initiate the interaction
- Only an application can use its own database
- They are now called "services"
- APIs become **reusable** entities (if cataloged)

34

•  $\rightarrow$  Service Oriented Architecture



# What are the Hardest Things in Software Design?

There are only two hard things in Computer Science: cache invalidation and naming things – Phil Karlton

*There are only two hard things in Computer Science: cache invalidation and naming things and one-off errors and cache invalidation* – K. Alan Bates





### And in IT EcoSystems?

*EcoSystems are distributed systems*... Caveat Emptor!

Lots of things are hard, especially distributed systems... And we have a giant ecosystem...

→Integration is often the biggest problem









### A Layered Ecosystem!



#### **Client Layer: Frontend & External Systems**

Web, mobile, speech, IoT, services, partners

#### **Service Layer**

(quite similar to the application)

Business Services: added value !

#### **Infrastructure Layer**

Generic Services (e.g. archival, pdf creation, ...), Databases, Middleware, Integration Solutions, ...



### Let's focus on the Service Layer



- Modularization → many small services
- High cohesion
- (Business) need for integration  $\rightarrow$  many interlinks
- High coupling
- Possible cascading failures
- Need for *orchestration* (central control)

This is where EDA can take things a step further...



### **EDA for Lower Coupling**



**Event Broker** 

- Start replacing synchronous, direct calls, with asynchronous, indirect Events
- Replace interdependencies with the use of a new infrastructure: the Event Broker
- lower coupling
- Events become **reusable** entities (if cataloged)

40

replace orchestration with *choreography* (decentralized control)

#### **Infrastructure Layer**



### **Revisit CAP: "Eventual" Consistency**

- EDA reduces coupling; increasing independency of services and reducing cascading failures
- $\rightarrow$  increased availability (and other advantages)
- We don't wait for an answer when we send an event: we **no longer perform a transaction**
- → consistency is *delayed* (but *eventually* reached)

Typically, a lot of business processes can handle a small delay in consistency. Data is **propagated via Events behind the scenes to ALL** interested systems/parties, potentially triggering numerous other automations, and reaching *consistency across the ecosystem* in potentially minutes (if not mere seconds)

### Let's get back to the design of ONE service

#### "HEXAGONAL" design

- A.k.a. *onion architecture*
- Goal: keep business logic independent from all externalities / other concerns
- Async Comms The hexagon can act as an "anticorruption" layer between all the involved systems
- Ultimately, everything should be able to evolve separately, and plugged in/out easily → Increased Modularity

**Provided APIs** 

15-25 days into the study; +-20 slides: deeper

Data

**Business** 

42

Logic

Infrastructures

Services

mals

ICT for society



### Tackling Complexity in the Heart of Software



## **Domain Driven Design**

- Design method; book published in 2003
- Focus on *Strategic Design*: how to organize large business
  domains into separate subdomains, called Bounded Contexts;
  these are typically quite small; not the entire business logic of an application
- **Bounded Context**: Logical, cohesive subset of the business domain that should be considered a single module
- Also about the ubiquitous language: a common language between business and IT, embedded into everything from analysis to the programming code (can be different in each bounded context)



### **Event Storming**

- Method for finding the domain Events in your business, as a Lingua Franca for describing the business and implementing its supporting software
- Focus on *behaviour* first! → Not data, which risks your design being centered around an ER (database) scheme
- Eventually, it also helps you to identify commands, users, ... and altogether the **bounded contexts** within the business domain
- Starting with this method @analysis can lay a lot of the groundwork for a good, modular, future-proof architecture, combining the power of API and EDA





### **Does Complete Modularization = Microservices?**

### **NO** (we should be practical)

- We can design *a good monolith*: organizing our code and dependencies in a modular way and respecting all principles (high cohesion, low coupling, ...)
- The choice between monolith and microservice is not made @design level, but @deployment level,
   i.e. where do we effectively run which part of our application ?
- Separating a part of the system out of the monolith and deploying it independently, can be done as needed for scalability and performance reasons. If the part we take out is really small (and very independent), we call it a microservice
- Too many microservices can make an ecosystem harder to manage





### EDA 201

Everything should be made as simple as possible, but not simpler

– Albert Einstein

#### 4 Different styles of EDA usage



### **1. Event Notification**



- Subscribers are being made aware of certain Events and state changes by publishers
- Very useful to stop polling style of communication
- Event only has type and time-stamp (and perhaps publisher ID)
- No payload with other information: subscriber still has to ask (less decoupling)





### **2. Event Carried State Transfer**

begov.podmi.ocmw.LivingWageGranted Time:2023.10.21-15:43:21:321456

- **OCMW: 3300**
- Grantee: 881122.555.77
- **StartDate: 2023.10.31**
- Amount: {EUR; 809,42}
- ...

- Lots of relevant information added to **Event Payload**
- **Distribution** of timely, useful data, across many interested stakeholders in the **Ecosystem**
- **Decoupling** between publishers and subscribers
- *Optionally*: allow all stakeholders to construct their own view of reality

This is where a lot of **untapped potential** lies if you've

48

only been using Event Notification !!



### **3. Event Sourcing**



- Application storing all Events as its main source of truth (instead of only its current state to a database)
- Current state is "calculated" from the Event history
- Strictly implemented, the **Event store only grows**: corrective events may be necessary
- Useful for history reconstruction, audit logging, testing new versions, analytics, ...



# 4. Command & Query Responsibility Segregation



- CQRS
- Receiving Commands (causing state changes) is done by a different module than handling queries (only asking for information)
- Modules for answering queries can use their own internal state
- If deployed independently, query services can scale separately as needed





### **EDA Advantages, Examples**

Don't be afraid to give up the good to go for the great.

– John D. Rockefeller

What are the advantages of EDA for **resilient**, **agile** automation ? How does EDA get the data everywhere in real-time, so we have **once only** and **no wrong door** ?



51

Event Driven Architecture



### **EDA advantages & examples**

- Real-time eGovernment
- Agility & ReUse
- Resilience & other IT advantages

#### Disclaimer – high impact examples

The examples are based mostly on how we could do things if we had an **ecosystem with a lot of EDA technology** and existing events in place. I realize the effort it would take to first create the necessary preconditions of such an eventdriven ecosystem.



### **The Real-time eGovernment**

- Real-time Ecosystem, with events coming in from everywhere, representing all important business events
- Events going everywhere a.s.a.p., to all systems requiring them, making high-speed decisions and automations feasible
- Real-time insights in the data, using **on-the-fly analytics** on the stream of events
- Pushing our own events to 3<sup>rd</sup> parties, delivering immediate feedback to companies, citizens and social benefit organizations



# Example in Automatic Benefits: the Protected (Energy) Consumer (hypothetical, not exhaustive)



(Currently: Trimestrial Updates by FPS Economy in an incredibly shredded social landscape)



# **Agility & ReUse**

- Minimize the impact of changing requirements! Event Consumers can be added or removed without impacting Event producers. Decoupled systems can evolve and adapt independently to meet business requirements.
- Use Events to communicate between teams/domains, creating natural boundaries based on separate business contexts. (Think about Conway's Law.)
- Store your events, and retain all information with **potential future use**, "replay" Events to aid in simulating/testing.
- **ReUse existing Events** in new applications. If an Event is valuable to the ecosystem, publish it; even if nothing will consume it now.



# **Events are ReUsable Building Blocks (like APIs)**

- Systems/Applications can send a number of Events of certain types. Many other systems can subscribe to get some of these types of Events.
- Standards exist to describe Events (e.g. AsyncAPI, same organization as OpenAPI for REST)
- API Management Software now often supports Event Driven APIs

Nov 09, 2023

• Events ARE data. APIs can offer data, but also functionality; their data is usually at least already partially processed. Events can be closer to the uninterpreted truth. So they fill a different niche.

→ Many-to-Many model of reusable, timely Events to subscribe to with your new shiny application

We should add these to the ReUse Catalog !





# Example: knowing about Banished Foreigners (hypothetical, many assumptions)





### **EDA: IT Drivers**

- Increased **Resiliency** from decoupling and eventual consistency (CAP theorem)
- Stop "dis-architectures" using polling, or full batch transfer to update minor details:
   performance improvement and green IT
- EDA decoupling: Escape Service Interdependence, avoid cascading failures
- EDA decoupling: increase **modularity, scalability, maintainability**
- Flexibility, Composability (agility): easily add and remove Event Consumers without breaking producers → Increase future-proofing
- Align IT with the business using the Event as **representation of reality**



### Let's make it Happen!!

If I had asked people what they wanted, they would have said faster horses.

- Henry Ford

This is not business as usual. The vision (including certain examples) is very ambitious, and will take a lot of work to come true. How can we take some first steps towards it?





# Where do we start? How to evolve EDA usage

- What are some practical options to do first?
  - Let's discuss together in the workgroup!
- But Most Importantly: The Mind-Set Shift



### What first? Some options...

- PUBLISH your business events! Use Event Carried State Transfer instead of notifications
- Acquire a **full Event broker (HA, with persistence)** ?
- Get Event-Driven API support
- Get serious about Events in the entire software lifecycle (e.g. Event Storming)

 $\rightarrow$  Let's talk (find out who in your organization is involved in the workgroup ?)



### **Most Importantly: the Mindset Shift**

I have been impressed with the urgency of doing. Knowing is not enough; we must apply. Being willing is not enough; we must do.

– Leonardo da Vinci

- Using EDA makes a lot of sense, but we're not used to thinking about our business, *nor* our IT systems in terms of Events
- Traditionally: government gathers and manages data, and then uses it to implement policy
- Historically, eGov IT has evolved to simply make the above workflow digital, with minor automations

Can we evolve past being mere "data custodians" ?



### Mindset & Role Shift



This culture shift is THE greatest challenge



# **Mindset & Priority Shift**



#### **Continuous State; Eventual Intelligence**

**Continuous Intelligence; Eventual State** 



### Conclusion

The best time to plant a tree was 20 years ago. The second best

time is now

– Chinese Proverb

To eventually achieve the vision of IT as the agile

automator, using EDA, we should start with three things...





# Main Take-Aways:

- Keep learning about EDA (ask me!) and Change your Mindset!
   Don't be a *Reactive* custodian of data, be a Proactive engine of progress!
- Prioritize EDA at the same level as API for a Complete Integration Strategy
- Get a sense of urgency. Most of us are behind in our adoption of EDA. Start Now!



# Thank you !

### If you have any questions, do not hesitate to contact me

koen.vanderkimpen@smals.be

