

# Software and Services for Business Automation

PV207 BPM

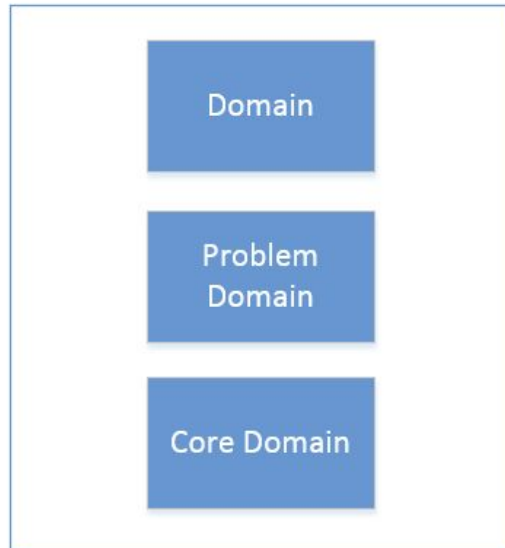
Mgr. Marian Macik  
Principal Software Quality Engineer

April 2024

# DOMAIN-DRIVEN DESIGN (DDD)

Domain-driven design is a concept that the structure and language of software code should match the business domain.

## Problem space

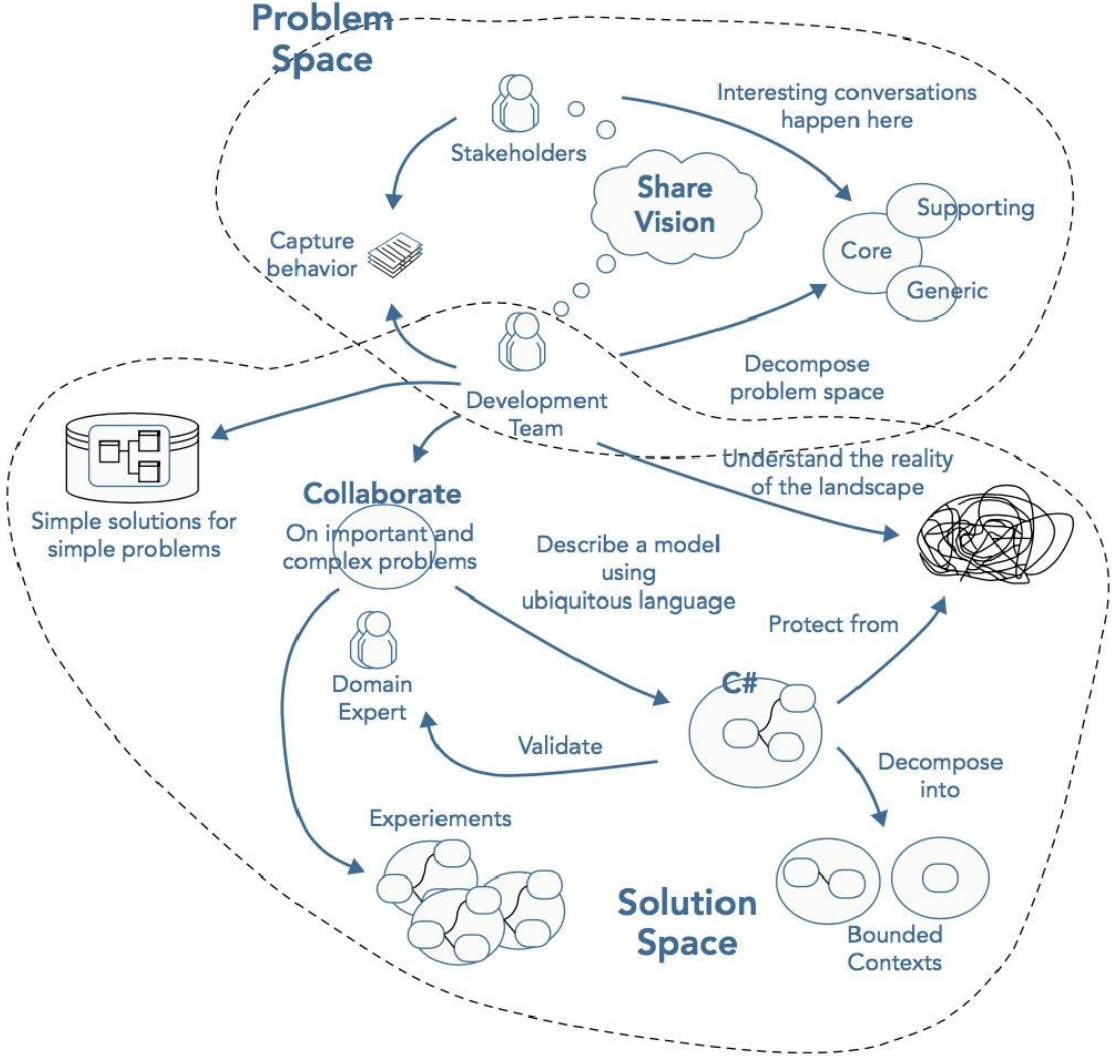


## Solution space

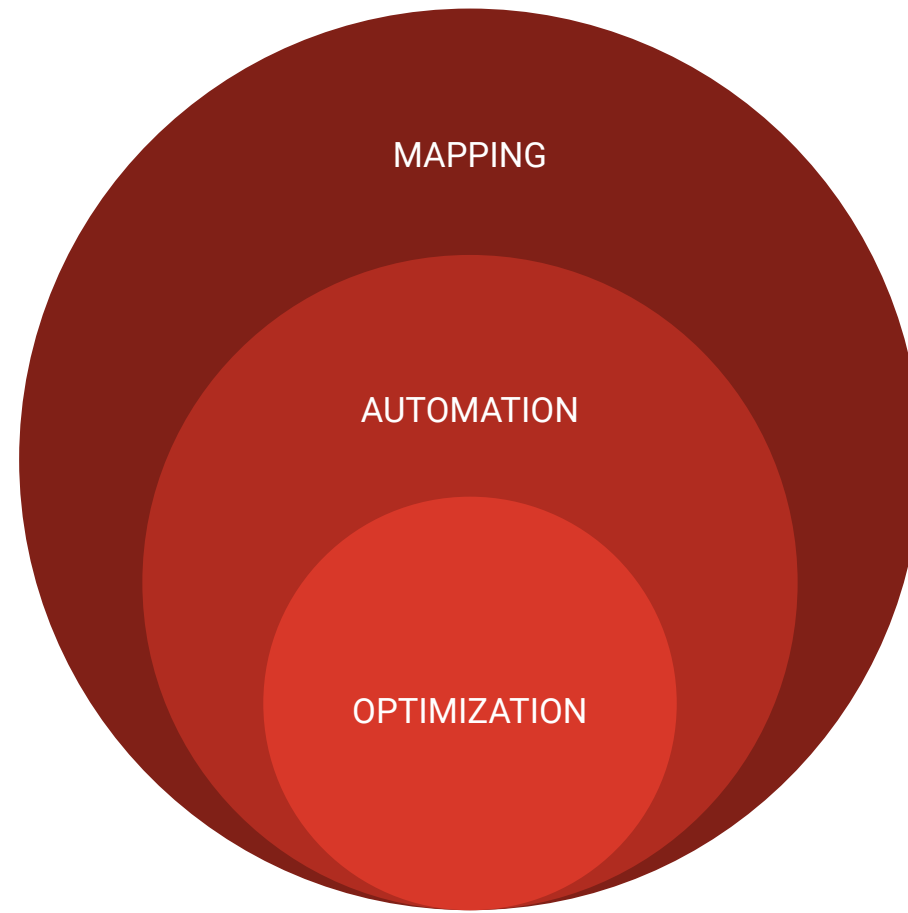


- Eases communication
- Improves flexibility
- Emphasizes domain over interface
- Requires robust domain expertise
- Encourages iterative practices
- Ill-suited for highly technical projects

# DOMAIN-DRIVEN DESIGN (DDD)



# BUSINESS AUTOMATION



MARKET

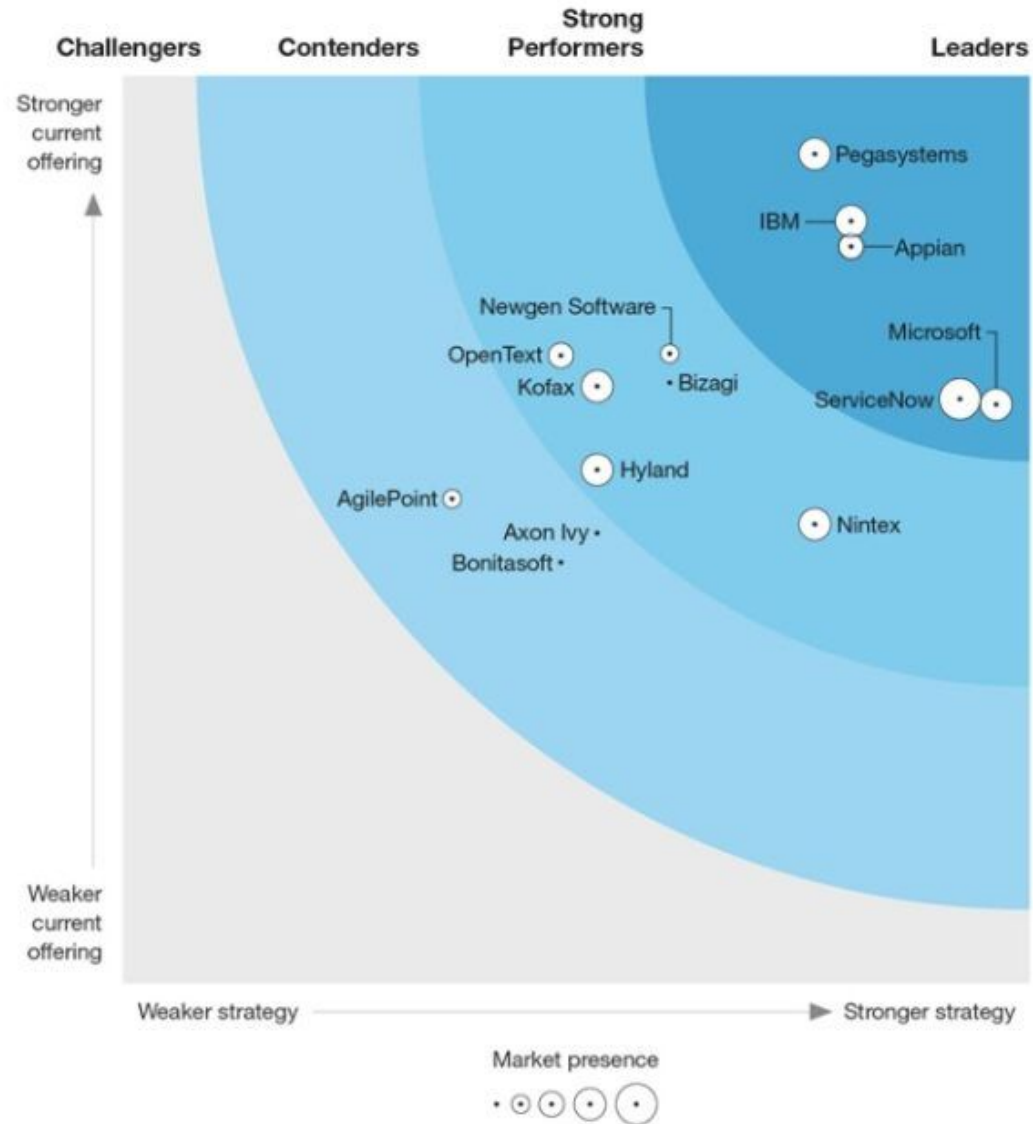
Low Code  
Platforms

DPA  
Platforms

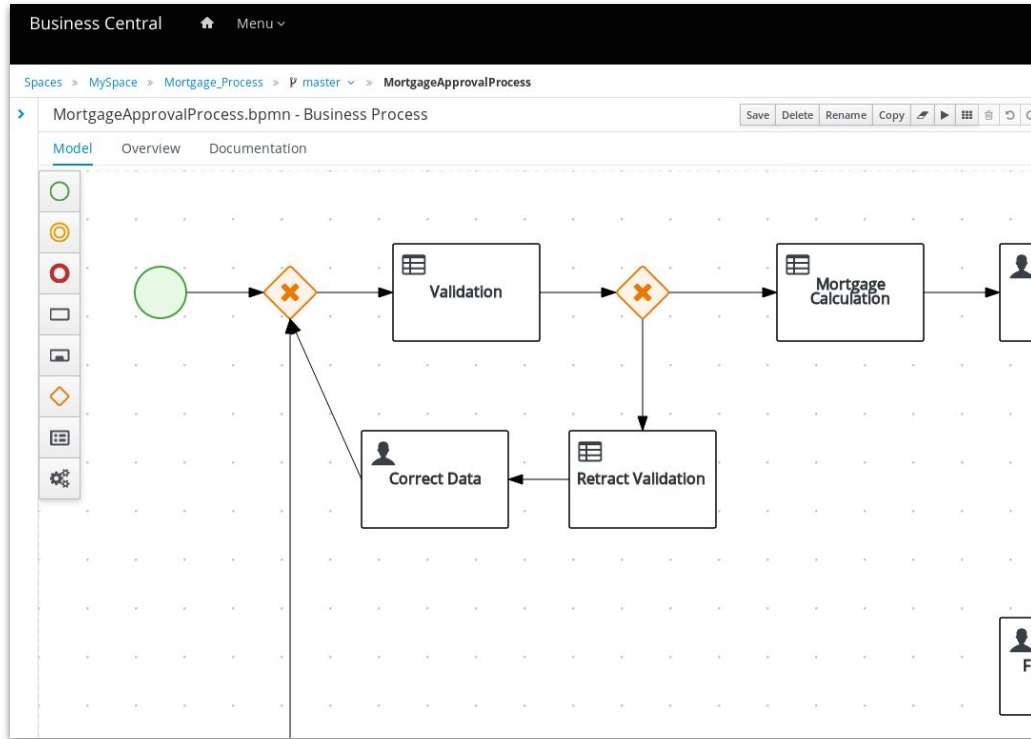
Cloud  
Services

Communities  
(Object Management Group, Cloud Native Computing Foundation, Kubernetes, Knative, KIE, Kogito)

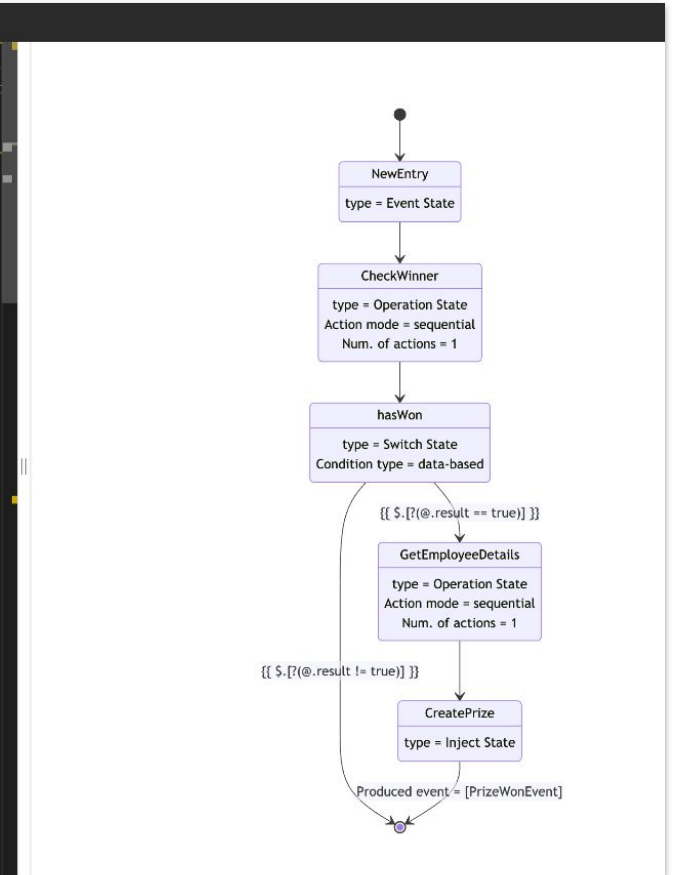
# DPA SOFTWARE FROM FORRESTER ANALYST REPORT '21



# AUTOMATION OF WORKFLOWS



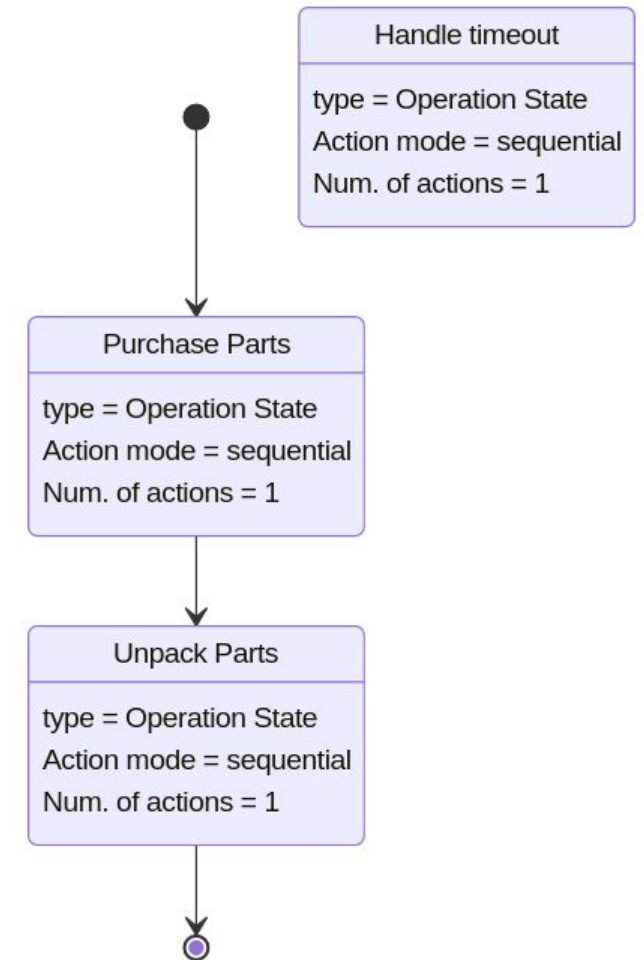
```
playtowin.sw.json
1 {
2   "id": "PlayToWin_ServerlessWorkflow",
3   "version": "1.0",
4   "name": "Play to win",
5   "description": "",
6   "expressionLang": "jsonpath",
7   "start": "NewEntry",
8   "events": [
9     {
10      "name": "NewEntryEvent",
11      "CloudEvent type": "participants",
12      "type": "participants"
13    },
14    {
15      "name": "PrizeWonEvent",
16      "source": "prizes",
17      "type": "prizes"
18    }
19  ],
20  "functions": [
21    {
22      "name": "isWinnerFunction",
23      "operation": "https://raw.githubusercontent.com/k
24    },
25    {
26      "name": "getEmployeeDetailsFunction",
27      "operation": "https://raw.githubusercontent.com/k
28    }
29  ],
30  "states": [
31    {
32      "name": "NewEntry",
33      "type": "Event State"
34    }
35  ]
36 }
```



# SERVERLESS WORKFLOW

- Similar specification to BPMN but aimed more at developers
- More lightweight, no human tasks, targeted at machine interaction
- Workflow is “modelled” directly by writing the code (JSON/YAML), currently doesn’t define graphical notation
- Higher level than BPM, focuses more on orchestration of services or infrastructure in the cloud, self-healing infrastructure
- Background service, end users don’t interact directly with it (as with BPM in loan approval, order processing...)
- <https://serverlessworkflow.io/>
- [BPMN vs. SWF](#)

```
id: executiontimeout
name: Execution Timeout Workflow
version: '1.0.0'
specVersion: '0.8'
start: Purchase Parts
timeouts:
  workflowExecTimeout:
    duration: PT7D
    interrupt: true
    runBefore: Handle timeout
states:
- name: Purchase Parts
  type: operation
  actions:
  - functionRef: purchasePartsFunction
  transition: Unpack Parts
- name: Unpack Parts
  type: operation
  actions:
  - functionRef: unpackPartsFunction
  end: true
- name: Handle timeout
  type: operation
  actions:
  - functionRef: handleTimeoutFunction
functions:
- name: purchasePartsFunction
  operation: file://myservice.json#purchase
- name: unpackPartsFunction
  operation: file://myservice.json#unpack
- name: handleTimeoutFunction
  operation: file://myservice.json#handle
```





# AUTOMATION OF WORKFLOWS

The LogicApp.json interface displays a workflow configuration. The first step is a trigger: "When a feed item is published". It is configured with the RSS feed URL "http://feeds.reuters.com/reuters/topNews", an interval of "1" minute, and a frequency of "Minute". The second step is an action: "Send an email". It is configured with the following details:

- Title: Feed title
- Body: Date published: Feed published...
- Link: Primary feed link
- Subject: New RSS item: Feed title
- To: sophia-owen@fabrikam.com

The interface also shows a "Design" view and a "Code View" tab at the bottom.

The Google Cloud Platform Cloud Workflows Demo interface shows a workflow definition in the "Define workflow" step. The workflow is defined in JSON:

```
1 - getCurrentTime:  
2   call: http.get  
3   args:  
4     url: https://us-central1-workflowsample.cloudfunctions.net/datetime  
5     result: CurrentDateTime  
6 - readWikipedia:  
7   call: http.get  
8   args:  
9     url: https://en.wikipedia.org/w/api.php  
10  args:
```

The "Visualization" pane on the right shows a flowchart of the workflow:

```
graph TD  
  Start((START)) --> GetCurrentTime[getCurrentTime call]  
  GetCurrentTime --> ReadWikipedia[readWikipedia call]  
  ReadWikipedia --> ReturnOutput[returnOutput return]  
  ReturnOutput --> End((END))
```

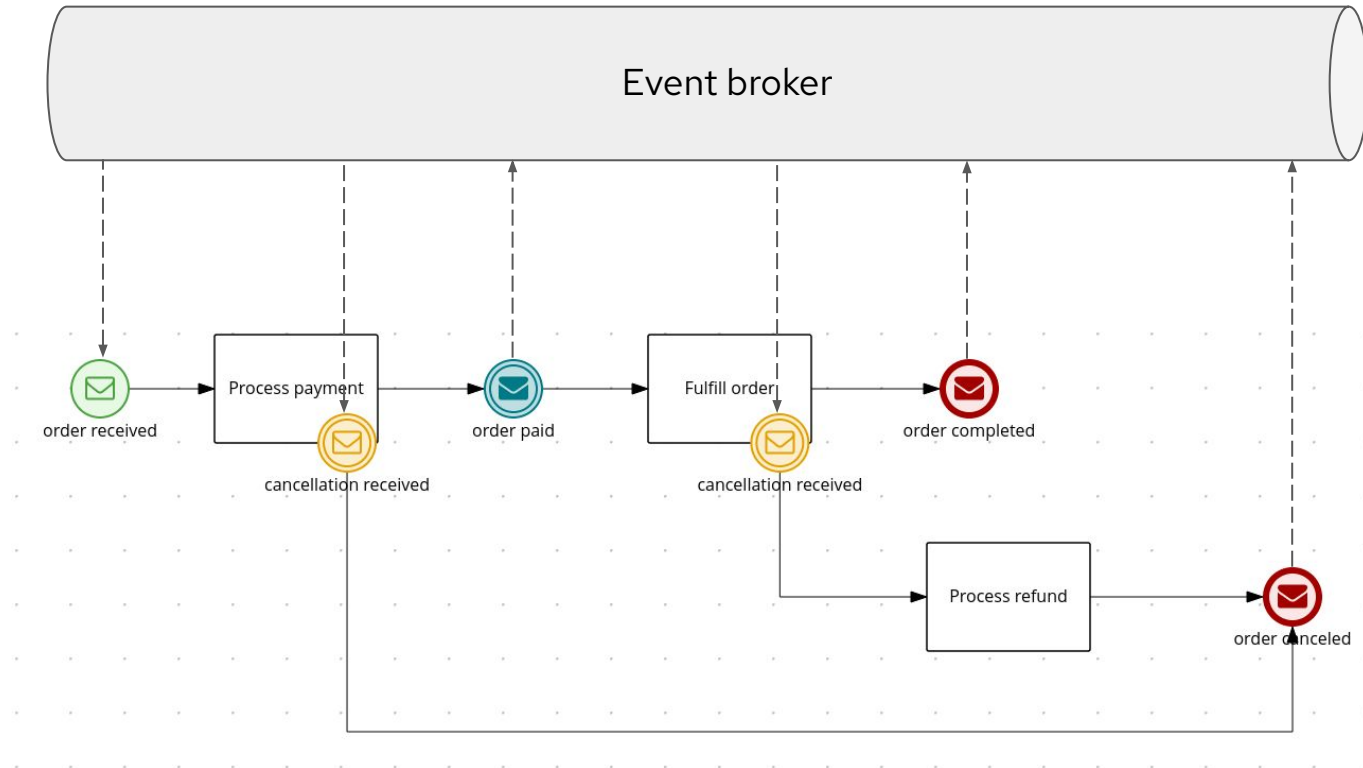
The AWS Step Functions console interface shows a workflow design in "Step 2: Design workflow". The workflow is a linear sequence of tasks:

- Start
- Lambda: Invoke Prepare Data
- Glue: StartJobRun Process Data
- Parallel state Data Insights Parallel Step
  - ECS: RunTask Data Insights
  - Drop state here
- End

The "Prepare Data" task configuration is shown on the right:

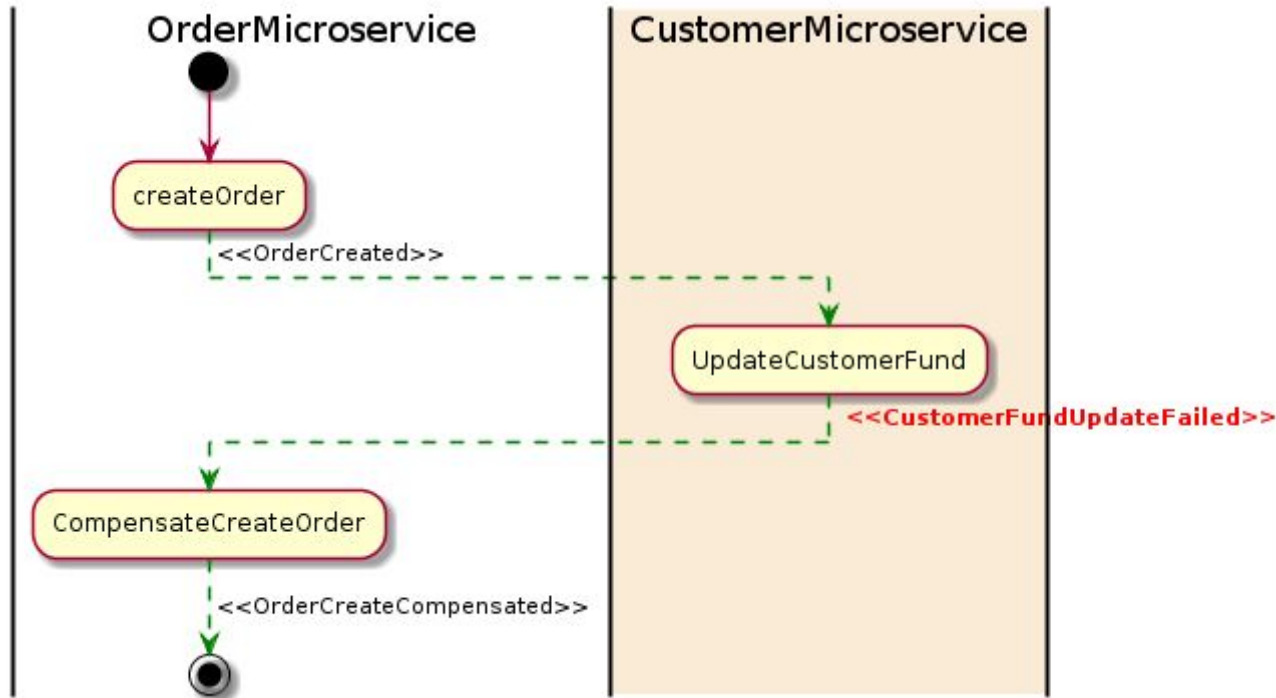
- State name: Prepare Data
- API: AWS Lambda: Invoke
- Integration type: Optimized
- API Parameters: Edit as JSON
- Function name: Choose an option
- Payload: Use state input as payload

# AUTOMATION OF EVENT-DRIVEN WORKFLOWS



# AUTOMATION OF LONG-LIVED TRANSACTIONS

Saga pattern for microservice architectures



If any microservice fails to complete its local transaction, the other microservices will run **compensation** transactions to rollback the changes.

## Advantages

- support for long-lived transactions
- other microservices are not blocked if a microservice is running for a long time
- there is no lock on any object

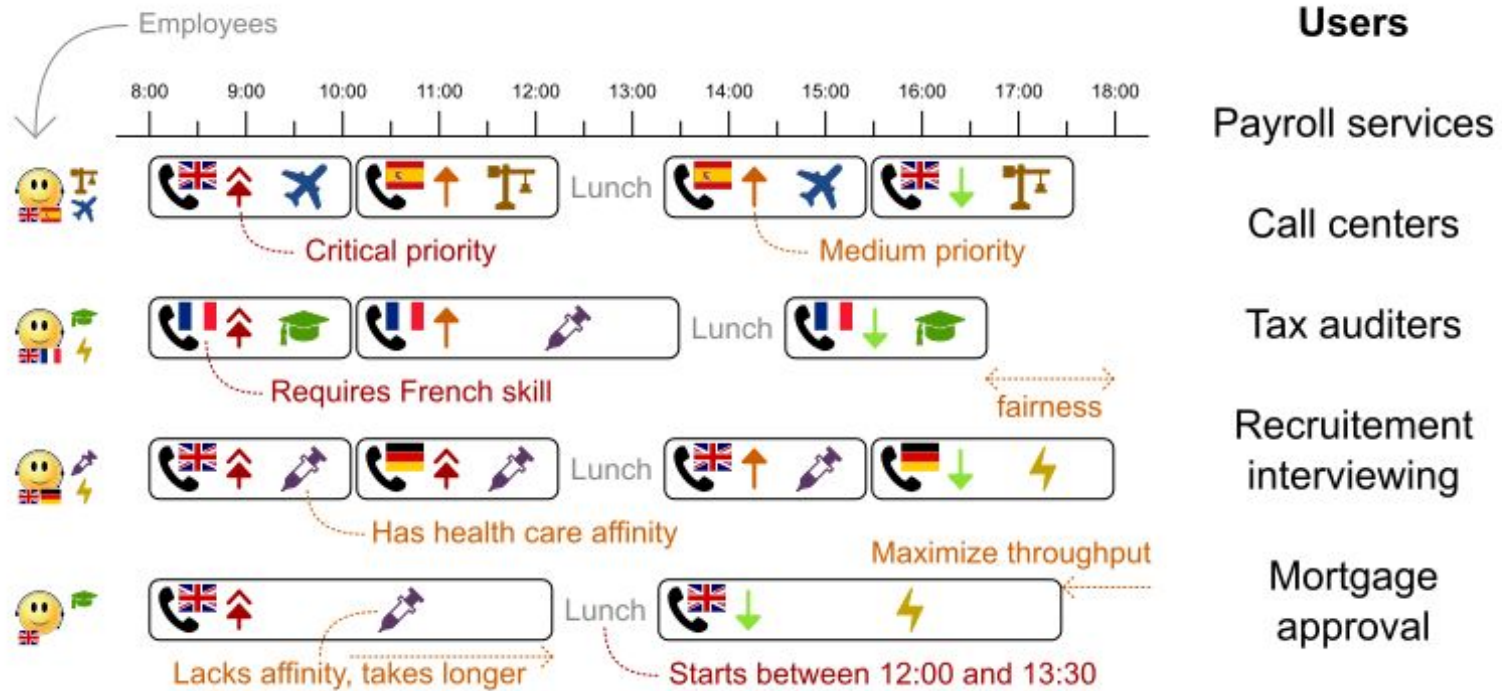
## Disadvantages

- difficult to debug
- difficult to maintain if the system gets complex
- does not have read isolation


Adding a **process manager** addresses the complexity issue of the Saga pattern when it becomes responsible for listening to events and triggering endpoints.

# AUTOMATION OF TASK ASSIGNMENTS

Optimize the task queue of every employee by reassigning and reordering tasks.

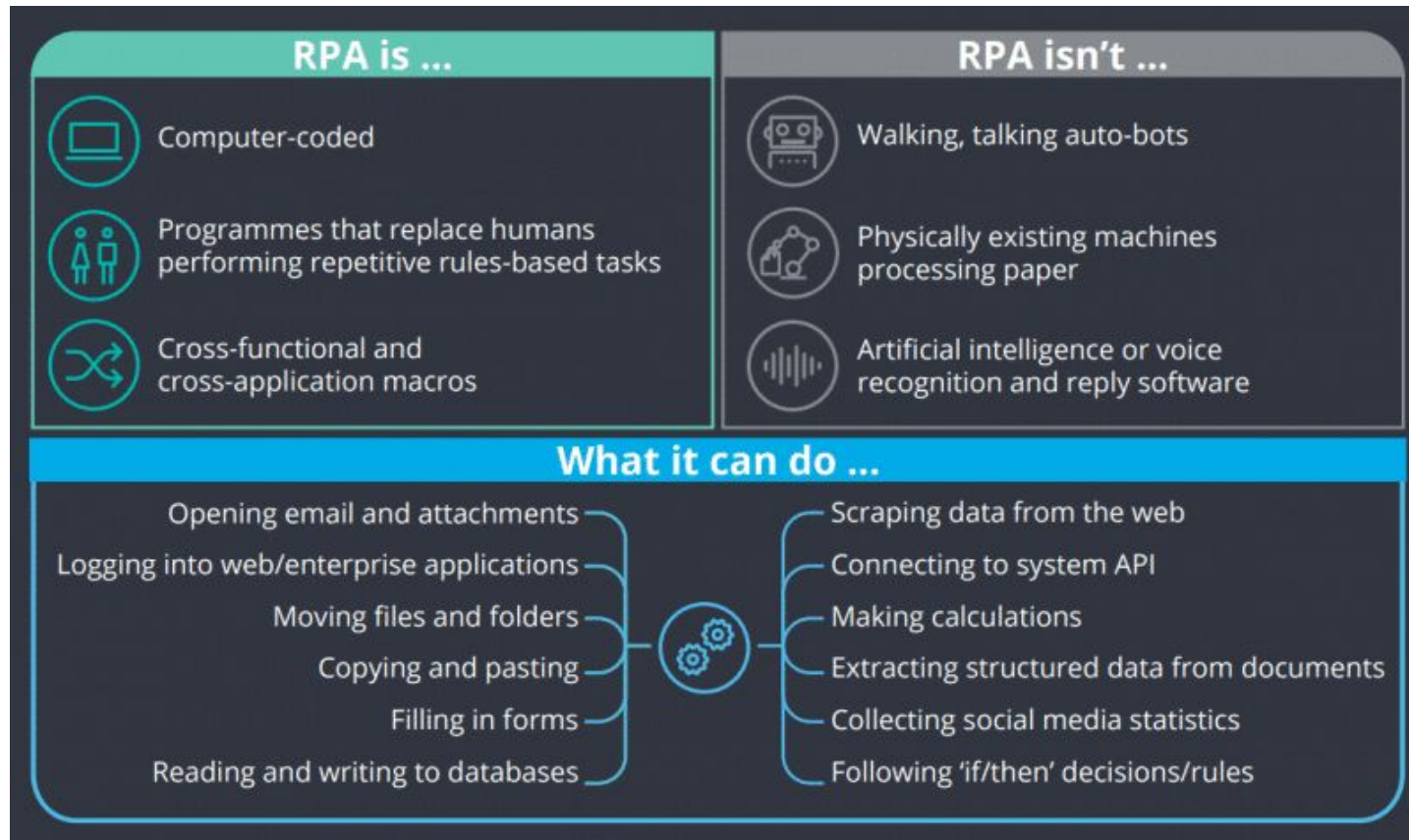


[\[KIELive#45\] Optimizing user tasks assignment in business processes with Kogito and OptaPlanner\\*](#)

\* OptaPlanner has been succeeded by  timefold

# AUTOMATION OF HUMAN ACTIONS INTERACTING WITH SW

## Robotic Process Automation (RPA)



# AUTOMATION OF BUSINESS RULES

[Kogito DRL Rules language](#)

The screenshot shows the LogicDrop web interface. The top navigation bar includes the LogicDrop logo, a project dropdown set to 'Insurance Sample', and user information. The main content area displays a breadcrumb trail: 'Projects > Insurance Sample > Rules > Approve or Deny Policy'. Below this, the rule name 'Driver is a single male' is shown with an edit icon. The rule is defined in DRL (Decision Rule Language) as follows:

```
1 when
2   $driver : Driver(
3     gender == "MALE",
4     age < 25,
5     maritalState == "SINGLE",
6     inf : insuranceFactor)
7 then
8   $driver.setInsuranceFactor(1.6 * inf);
9   sparks.print("Driver Single Young Male Driver: " + $driver.getInsuranceFactor());
10
```

On the right side of the interface, there is a 'Ruleset Source' panel with a 'Freeform Rules' section. It lists several rules, with 'Driver is a single male' highlighted in blue.



IBM Action rules express business policy statements using a predefined business vocabulary that can be interpreted by a computer.

## Policy

"change customers in the Gold category to the Platinum category when they spend more than \$1,500 in a single transaction"

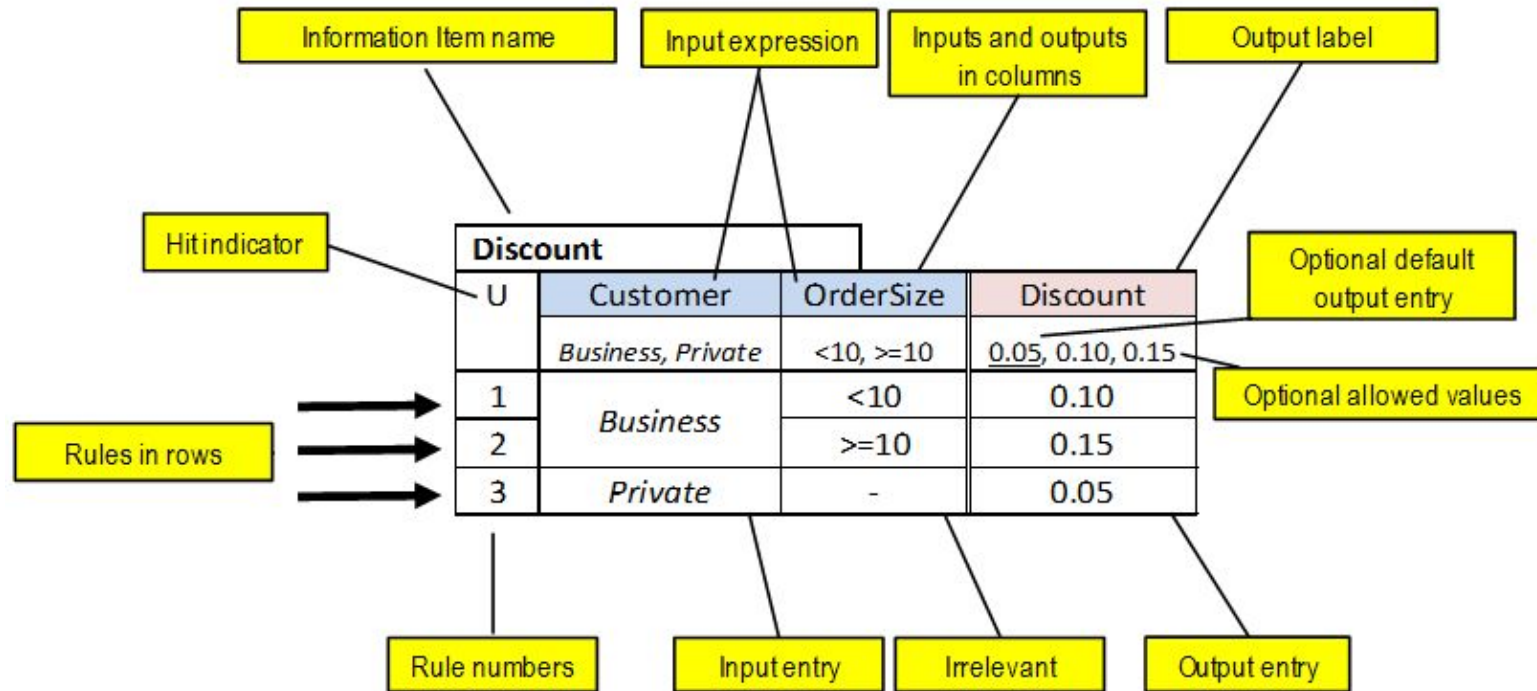
## Action Rule

If All of the following conditions are true:

- the customer category is Gold
- the value of the shopping cart is more than \$ 1,500

Then Change the customer category to Platinum

# AUTOMATION OF DECISION TABLES

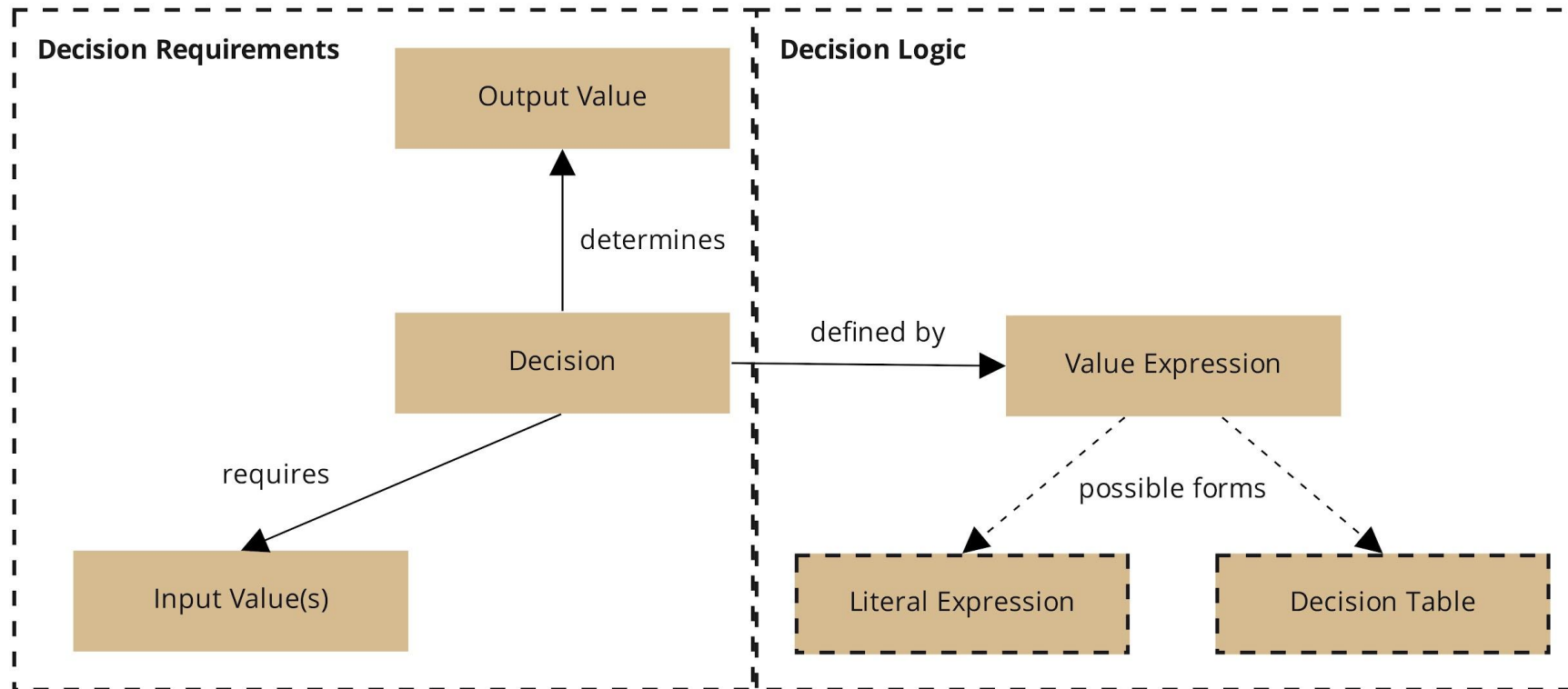


Types of decisions

- Selection (routing)
- Scoring
- Categorizing

The decision doesn't take an action (no side effect), just determines a data value

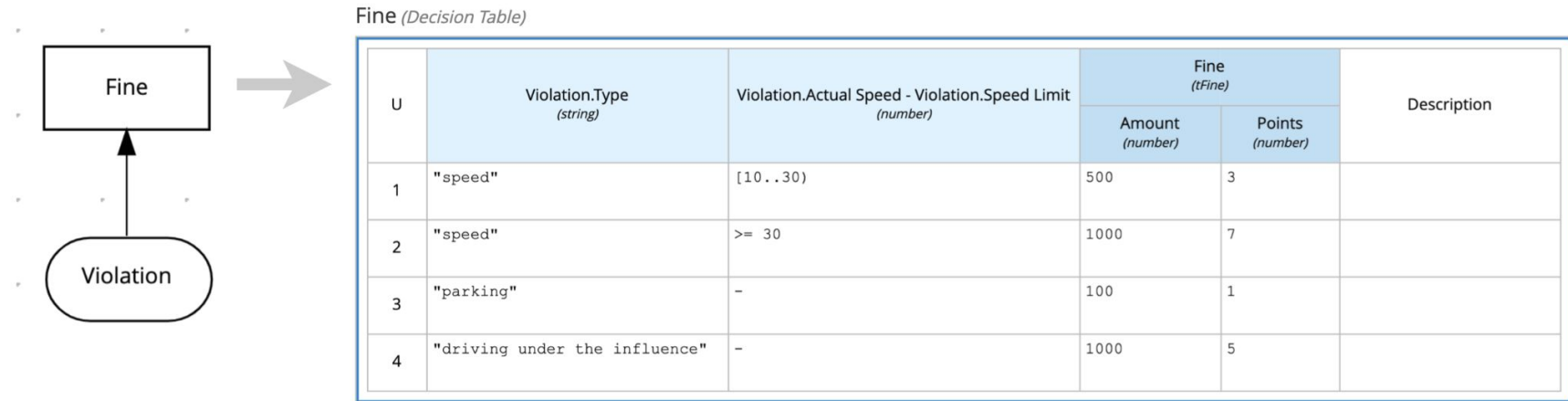
# AUTOMATION OF DECISION MODELS





# AUTOMATION OF DECISION MODELS

Executing decision logic in DMN models



```
/* = Actual input data = */  
"Violation": {  
  "Type": "speed",  
  "Speed Limit": 60,  
  "Actual Speed": 100  
}
```

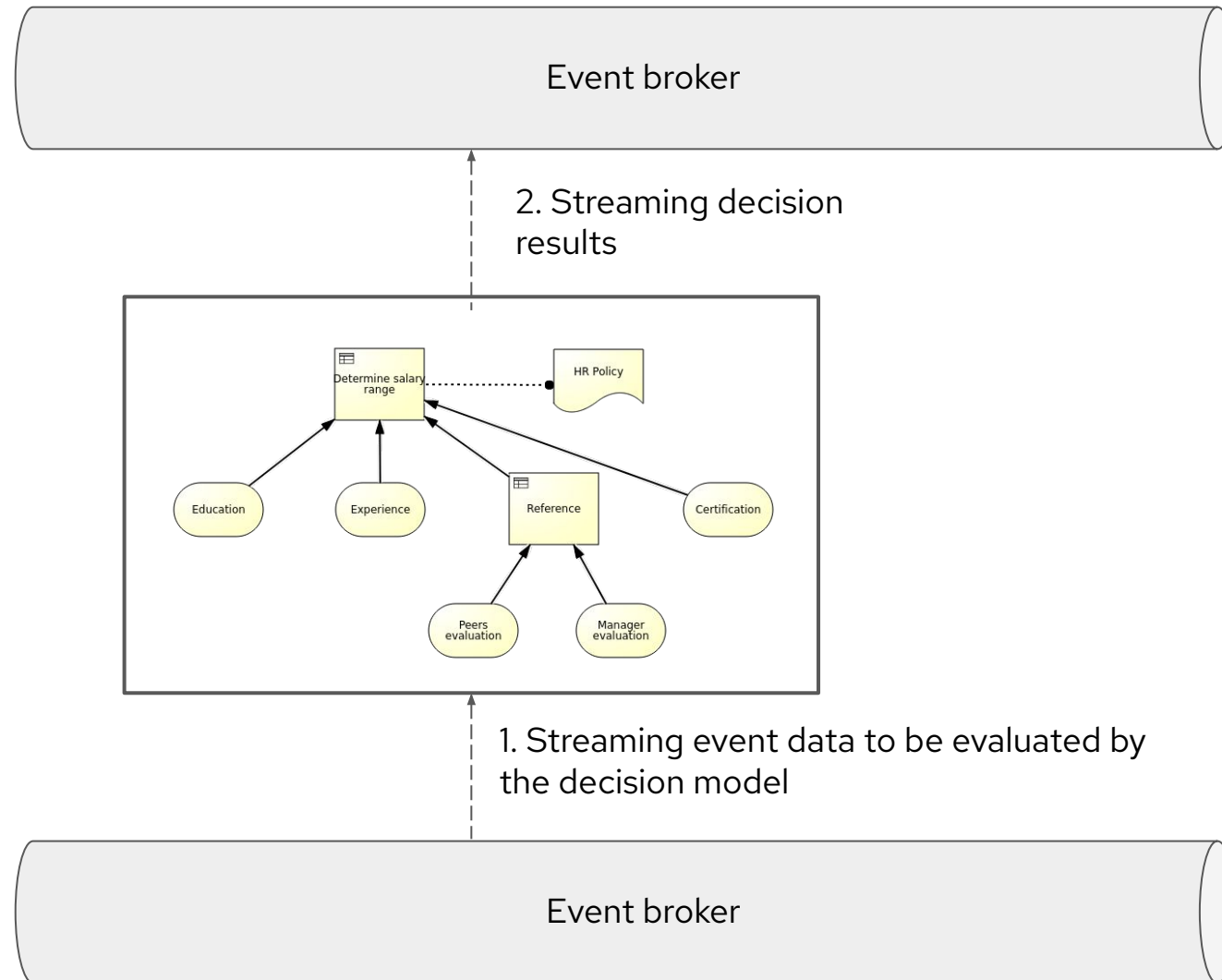


```
/* = Expected output data = */  
"Fine": {  
  "Points": 7,  
  "Amount": 1000  
}
```

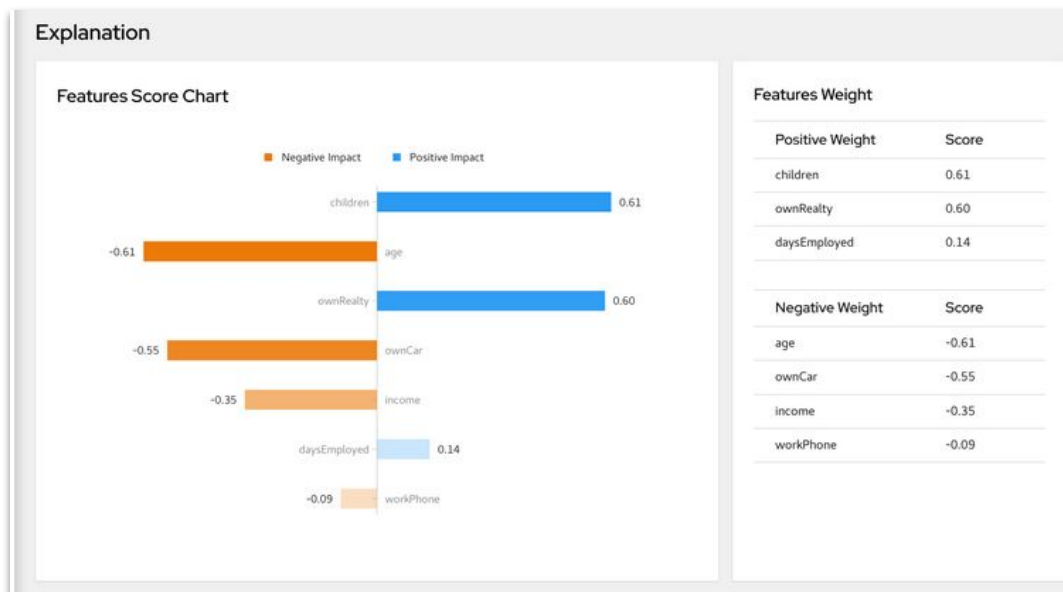
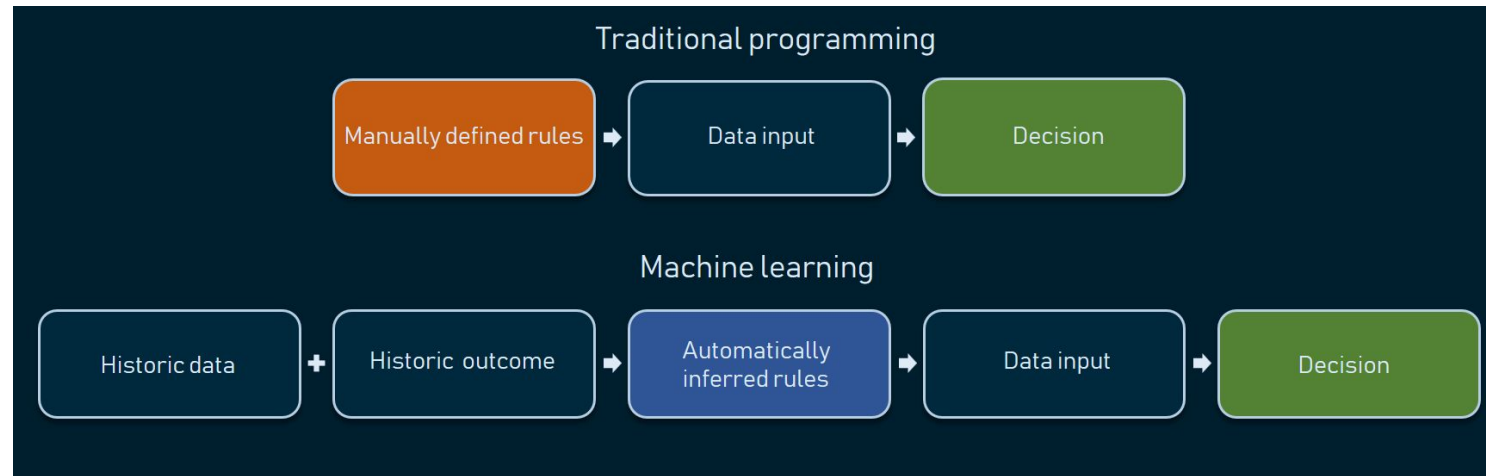
[DMN with Kogito on Quarkus](#)

[DMN FEEL Cheatsheet](#)

# AUTOMATION OF DECISION MAKING IN STREAMS



# AUTOMATION OF DECISION MAKING WITH AI



[TrustyAI](#)

[Can you trust AI? - presentation](#)

# AUTOMATION OF SCORING

**Scorecard** is a risk management tool used mostly by banks to calculate the risk they take by selling you one of their products.

How risky is a customer?

What are the changes the customer might default on payment?

Based on the customer score we may adapt the product offer - a better interest rate, a higher credit limit, etc.

[PMML Scorecard Editor in VS Code](#)

The screenshot displays the 'SampleScorecard' interface. At the top, there are three buttons: 'Set Data Dictionary', 'Set Mining Schema', and 'Set Outputs'. Below this is the 'Model Setup' section, which includes several configuration options: 'Is Scorable: Yes', 'Function: regression', 'Initial Score: 0', 'Use Reason Codes: Yes', 'Reason Code Algorithm: pointsBelow', and 'Baseline Method: other'. The main area is titled 'Characteristics' and features a search bar labeled 'Filter by name' and an 'Add Characteristic' button. The characteristics are listed in a scrollable container:

- departmentScore** (Reason code: RC1, Baseline score: 19)
  - department isMissing (Partial score: -9)
  - department = "marketing" (Partial score: 19)
  - department = "engineering" (Partial score: 3)
  - department = "business" (Partial score: 6)
- ageScore** (Reason code: RC2, Baseline score: 18)

# AUTOMATION OF HUMAN-LIKE CONVERSATIONS

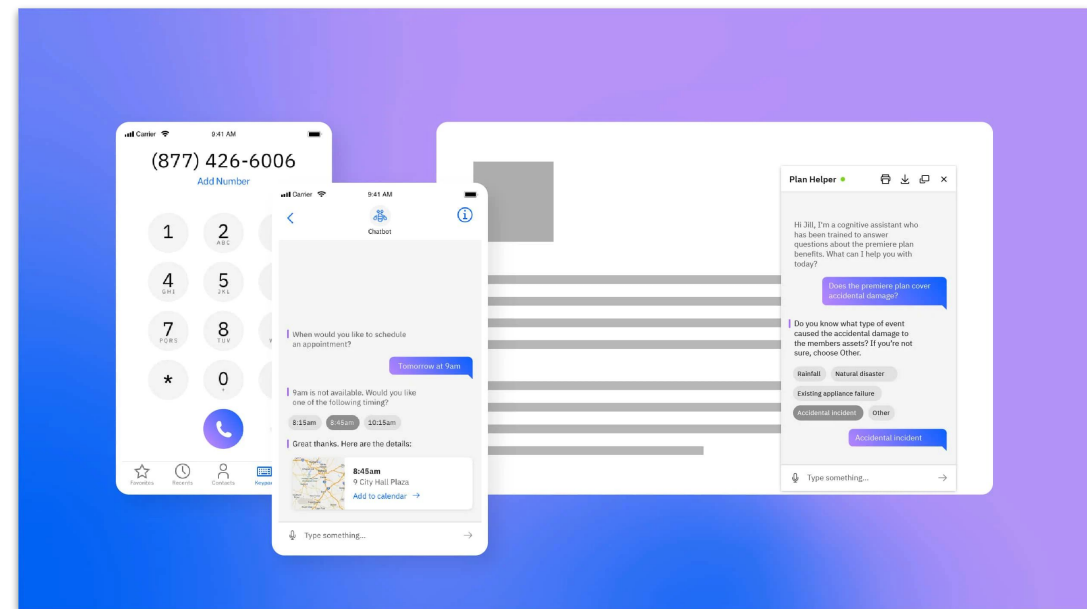
## Chatbots

A chatbot is software that simulates human-like conversations with users via text messages on chat or text-to-speech.

- Customer self-service - call centres, scheduling doctor appointments
- Employee self-service - HR assistants, meeting and scheduling, expenses, people finder

Omnichannel - communication is done on all kinds of channels - phone calls, chat and email on computers, sms messages, ...

Watson Assistant - COVID-19 response automation

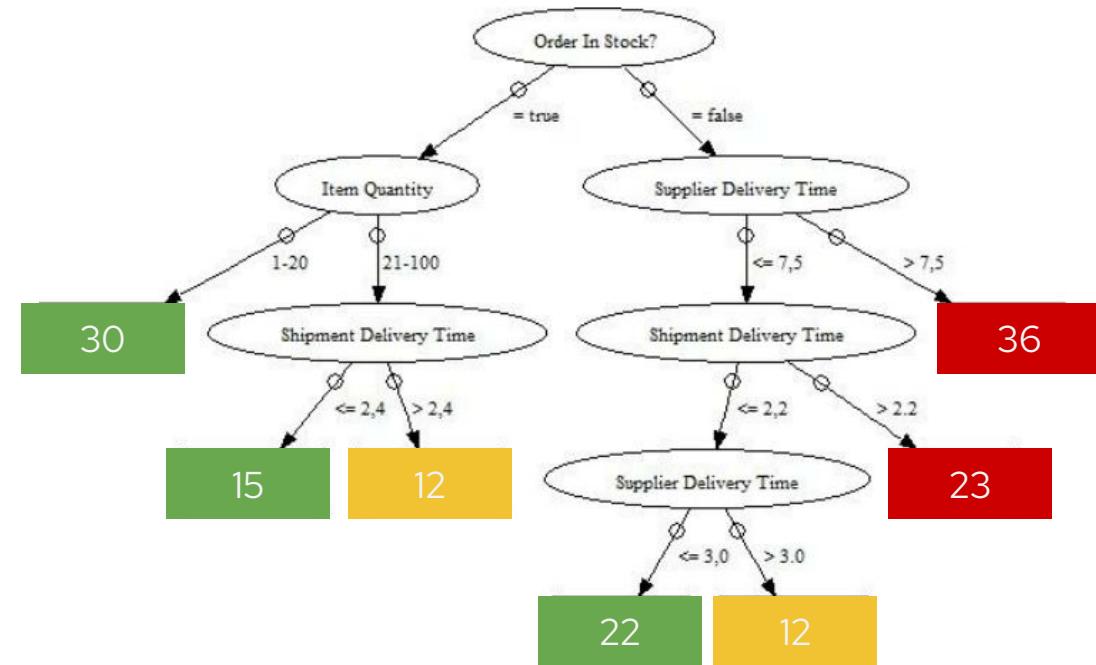


# AUTOMATION OF BUSINESS MONITORING

## Patient Satisfaction Dashboard



## KPI Dependency Tree



Alerts, triggers, ...

Choose the right software and  
services to fit your needs

# Kogito

Cloud-native business automation for building intelligent applications, backed by battle-tested capabilities.





# KOGITO

<https://kogito.kie.org/>

- Domain-Driven Development
- Generated Domain-specific APIs from your BPMN and DMN models
- Lightweight orchestration microservices
- Event-driven business logic
- Dev-mode hot reload
- Distributed sagas for microservices
- Serverless workflows
- Polyglot programming

More info at

- [Blogs](#)
- [Youtube](#)

# KOGITO DEMO

- Available on GitHub
  - git clone <https://github.com/MarianMacik/kogito-intro.git>
  - cd kogito-intro-quarkus
- Contains [BPMN](#) and [DMN](#) integration
- Development mode with hot reload
  - mvn clean compile quarkus:dev
- Compile
  - mvn clean install
- Domain-specific API available at <http://localhost:8080/q/swagger-ui/>
  - OpenAPI specification

Thank you,  
questions?