

# Business Automation in a Nutshell

PV207 BPM

Mgr. Ivo Bek  
Senior Product Manager

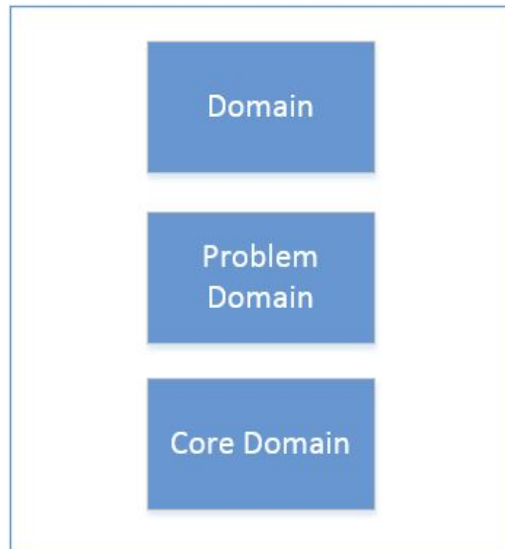
April 2021

Mgr. Marian Macik  
Senior Quality Engineer

# DOMAIN-DRIVEN DESIGN (DDD)

Concept that the structure and language of software code should match the business domain.

## Problem space

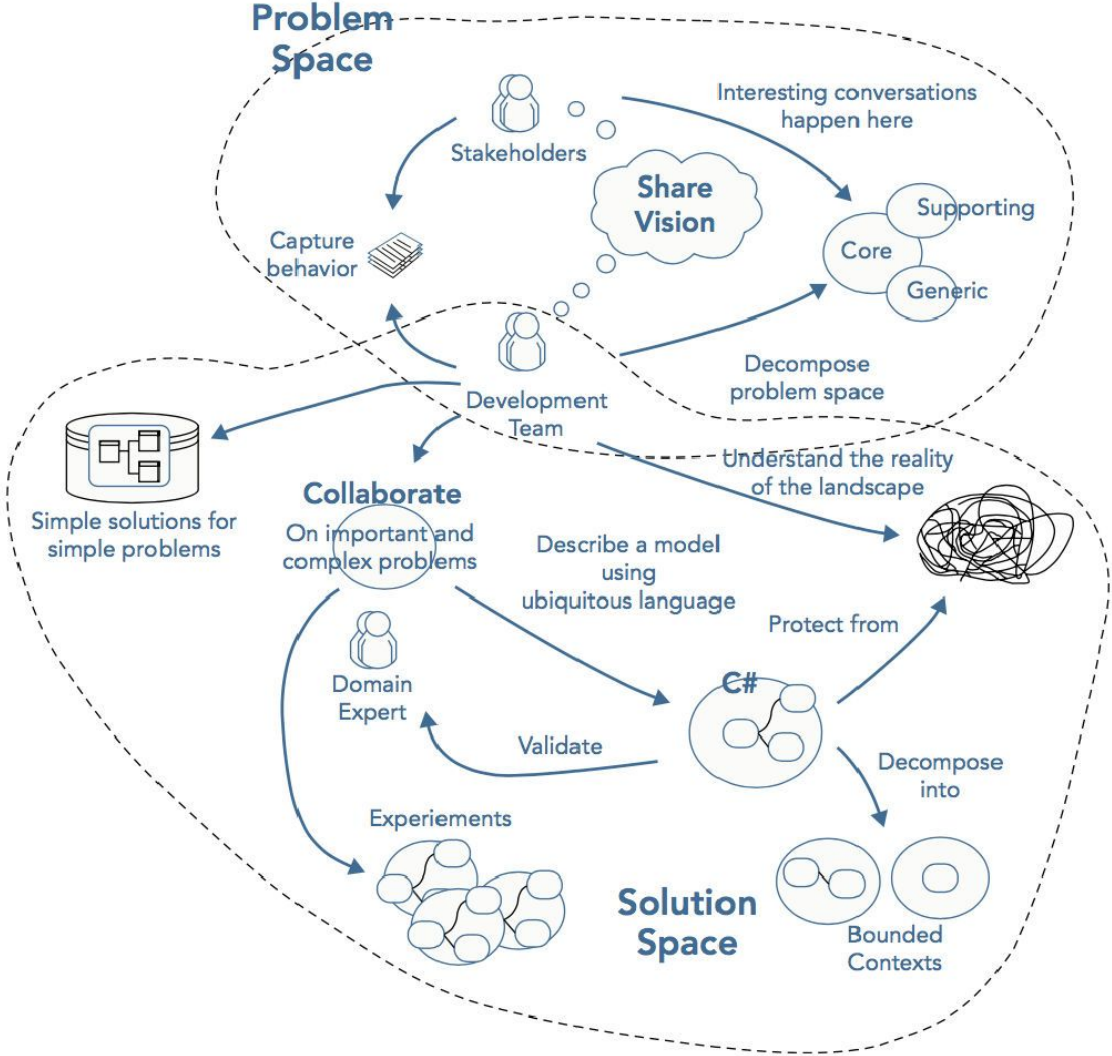


## Solution space

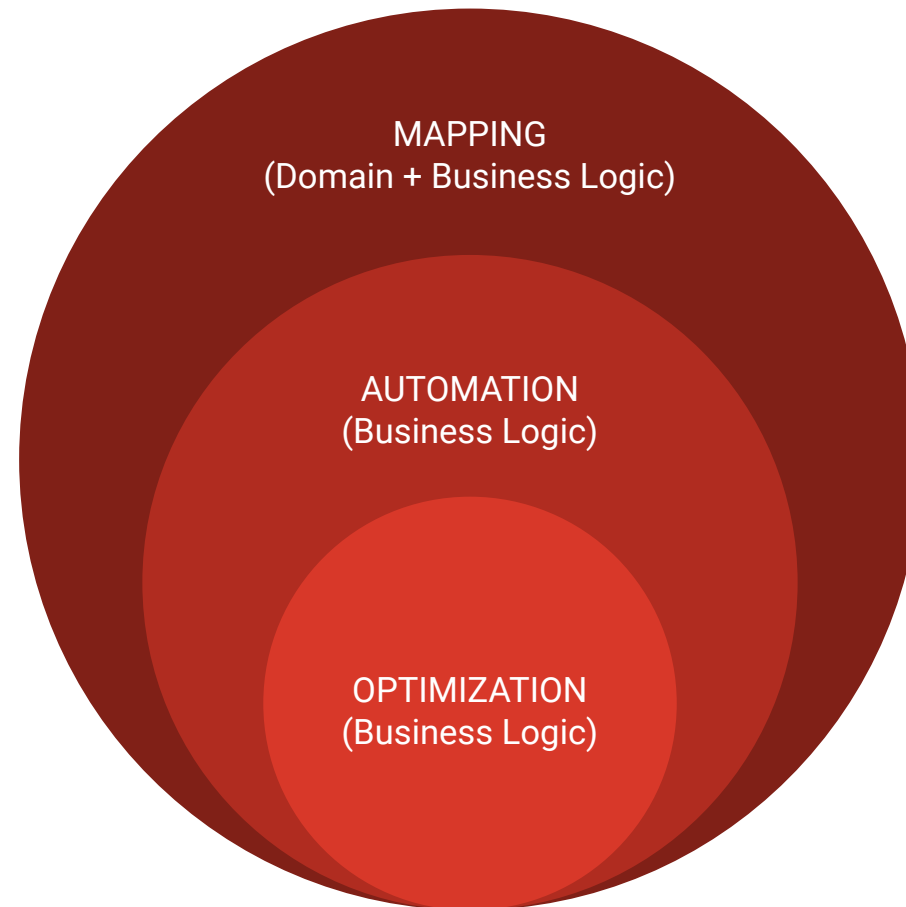


- Eases communication
- Improves flexibility
- Emphasizes domain over interface
- Requires robust domain expertise
- Encourages iterative practices
- Ill-suited for highly technical projects

# DOMAIN-DRIVEN DESIGN (DDD)



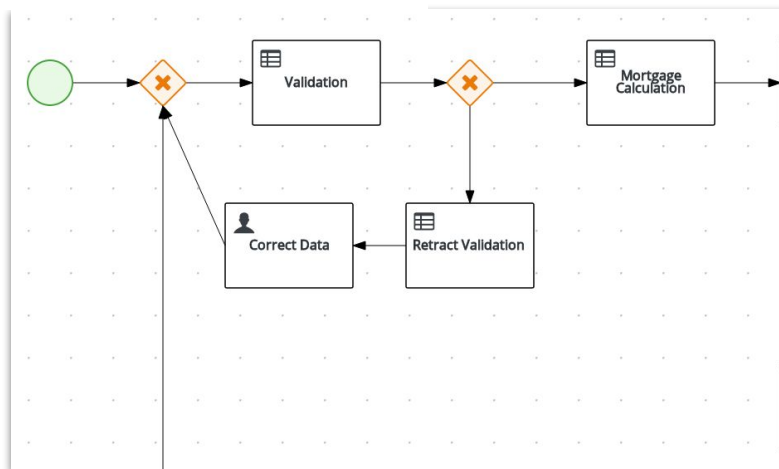
# BUSINESS AUTOMATION



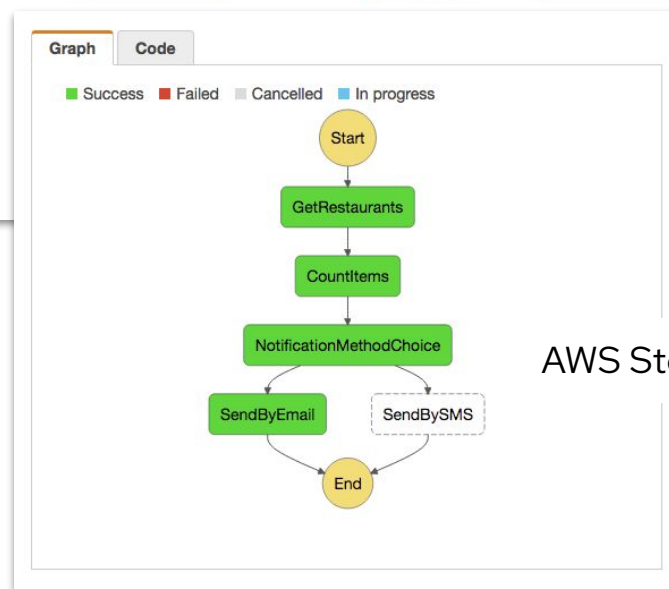
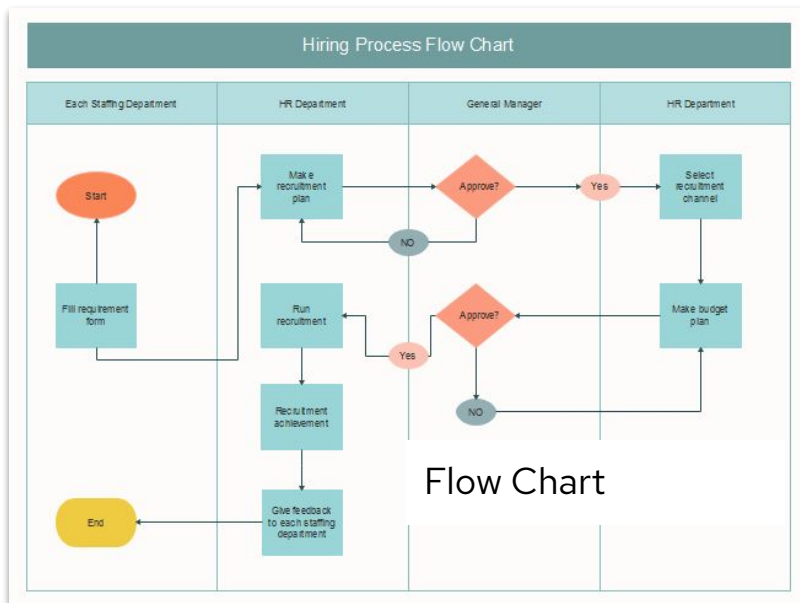
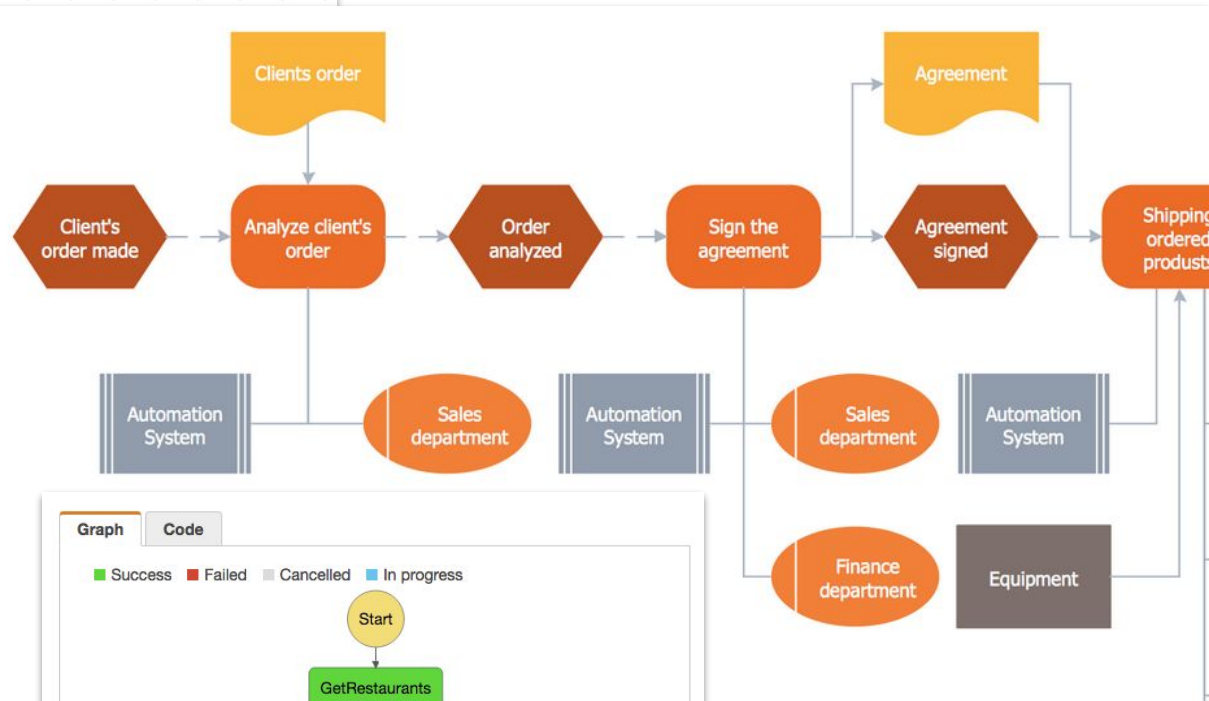
# PROCESS MODELS

- MAPPING
- Process models
- Process mining
- Triggers
- Participants
- Inputs and outputs
- Domain model
- DSL
- Business rules
- Decision tables
- Decision models

BPMN model



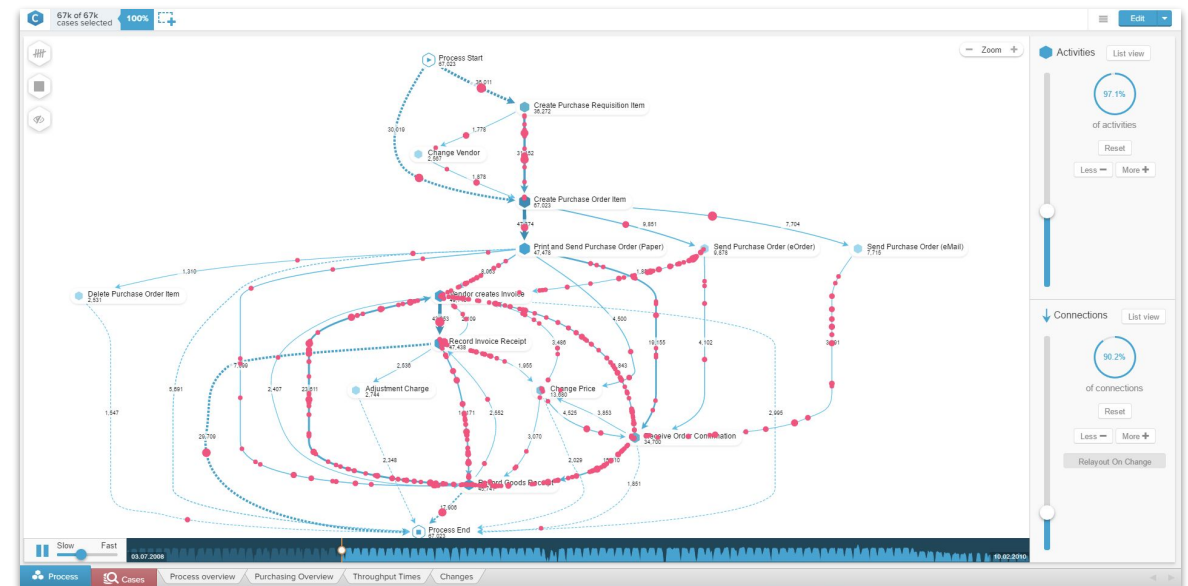
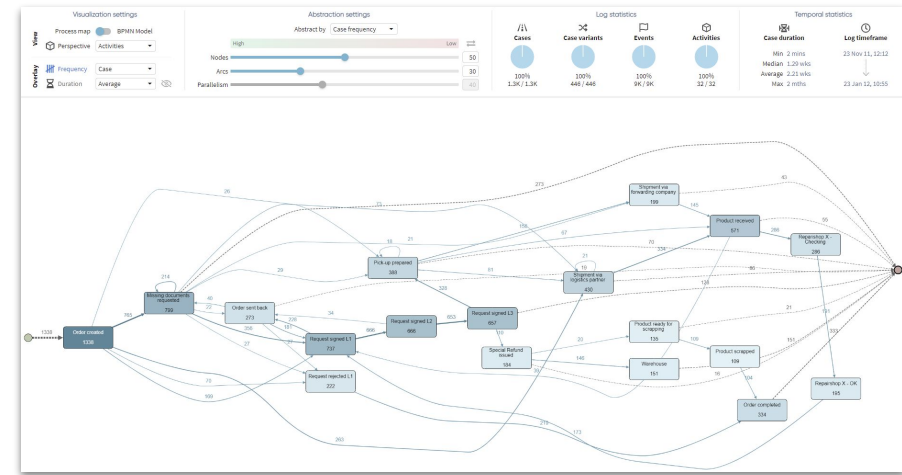
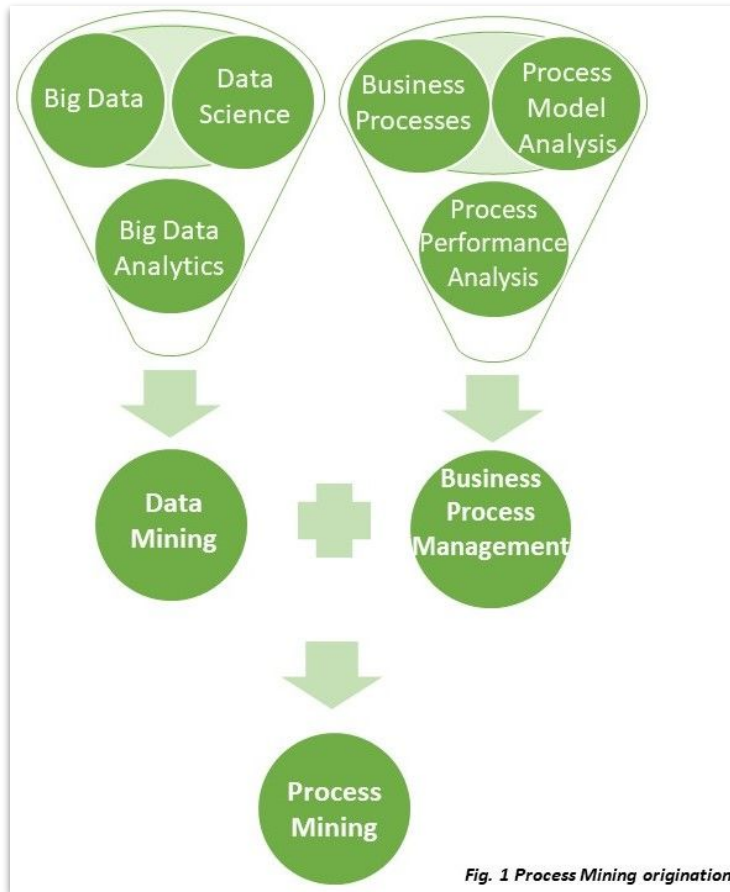
Event-driven process chain diagram



AWS Step Function

# PROCESS MINING

- MAPPING
- Process models
- Process mining**
- Triggers
- Participants
- Inputs and outputs
- Domain model
- DSL
- Business rules
- Decision tables
- Decision models



# TRIGGERS

## MAPPING

Process models

Process mining

**Triggers**

Participants

Inputs and outputs

Domain model

DSL

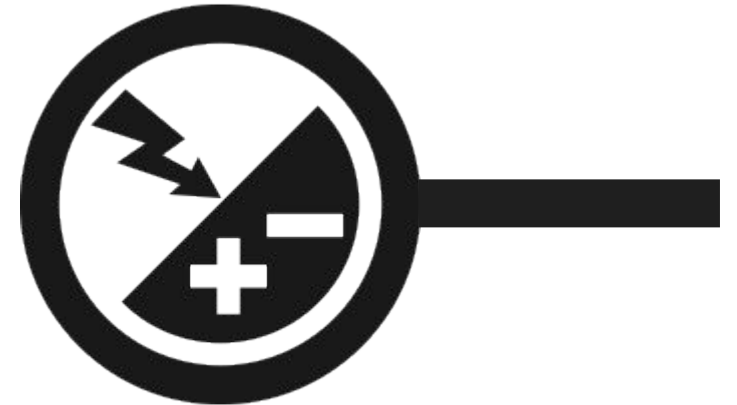
Business rules

Decision tables

Decision models

A trigger is the initial stimulus that initiates the subsequent steps of the process.

- Manual
- Scheduled
- Form
- Webhook
- Email
- Callable (subprocess)
- Alert (error)
- App-specific (from CRM, helpdesk, ...)



# PARTICIPANTS

## MAPPING

Process models

Process mining

Triggers

**Participants**

Inputs and outputs

Domain model

DSL

Business rules

Decision tables

Decision models

Participants are the main entities involved in a business process.

- People - individuals, groups, employees, customers, suppliers, partners, ..
- Robots
- Smart things
- Systems
- etc.





# INPUTS AND OUTPUTS

## MAPPING

Process models

Process mining

Triggers

Participants

**Inputs and outputs**

Domain model

DSL

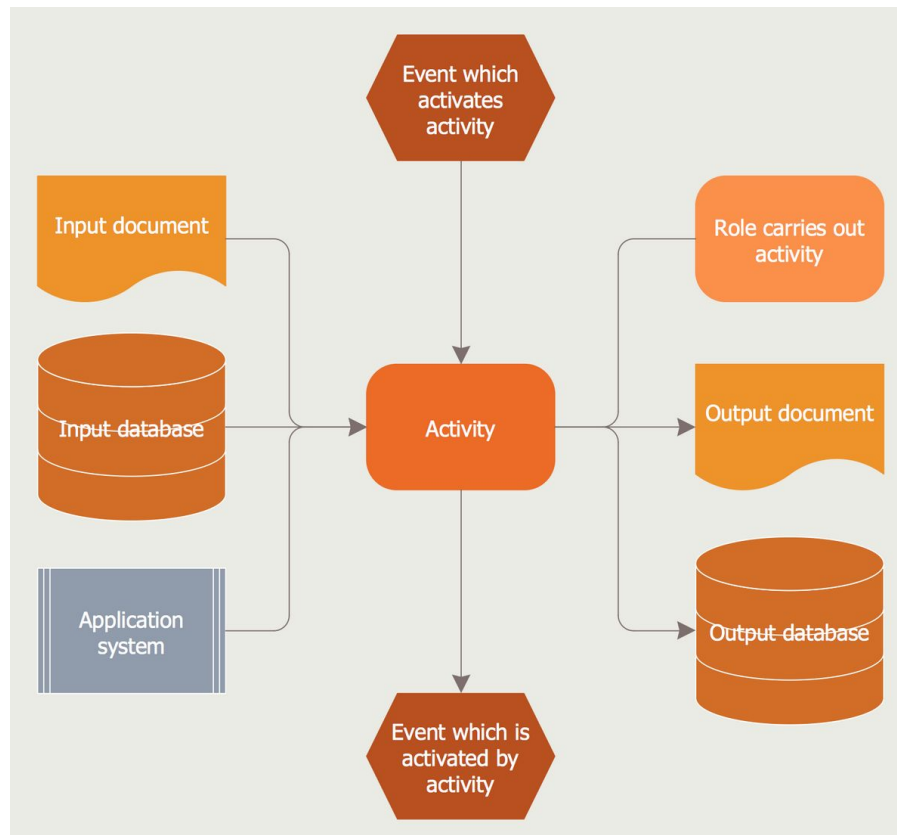
Business rules

Decision tables

Decision models

Processes, tasks and decisions require input data objects to be processed and produce results as their output.

- Documents - paper, electronic, ...
- Form data
- Events - signals, messages, errors, timeout, ...
- Sensitive data
- Data transformations should be done outside of a business process / decision



# DOMAIN MODEL OF BUSINESS OBJECTS

## MAPPING

Process models

Process mining

Triggers

Participants

Inputs and outputs

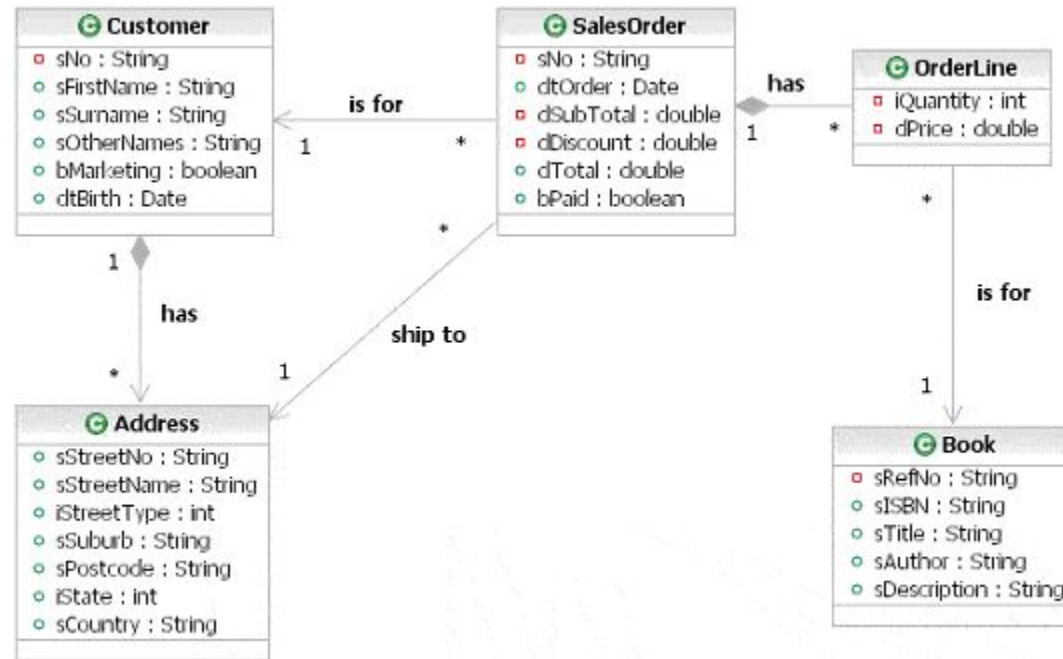
**Domain model**

DSL

Business rules

Decision tables

Decision models



Translates to domain-specific APIs

GET	/onboarding	Returns a list of onboarding
POST	/onboarding	Creates new instance of onboarding
GET	/onboarding/{id}	Returns information about specified onboarding
DELETE	/onboarding/{id}	Cancels specified onboarding

# DOMAIN SPECIFIC LANGUAGE

- MAPPING
- Process models
- Process mining
- Triggers
- Participants
- Inputs and outputs
- Domain model
- DSL**
- Business rules
- Decision tables
- Decision models

A computer language specialized to a particular application domain.

By hiding technical details, DSLs can empower users by giving them ability to set feature requirements and verify system behaviour.

Open up a functionality to internal users without having to prematurely build out a super complex, much more rigid GUI

The image shows a Gherkin scenario editor for a 'check age' scenario. The scenario is structured as follows:

```

SCENARIO check age
  Comment (Optional)
  scenario comment
  Tag (Optional)
  scenariotag
  GIVEN I'm logged in
  WHEN I check age for <name>
  THEN it should be <age>
  
```

Below the scenario, there are input fields for 'name' and 'age'. To the right, a code editor shows the corresponding DSL code:

```

Event: Internet on button: 1 Greeting: Did you know that our internet is fas
On button: 1 --> Discout
On button: 2 --> Data limit
On button: * --> Return to main menu

Event: Discout on button: 1 Greeting: Welcome in section of discounts!
On button: 1 --> Summer discount
On button: 2 --> Hidden discounts
On button: * --> Step back

Event: Summer discount on button: 1 Greeting: Hello!
other
Event: Hidden discounts on button: 2
get info
Event: Step back on button: *
  
```

A dropdown menu is visible over the code, listing actions like BACK, GENERAL, GET\_INFO, MENU, and OTHER.

The image shows an IDE window displaying a DSL file named 'Test01.dslr'. The code is as follows:

```

1 package rules1
2 import com.nobleprog.FactModel.*;
3 import java.lang.Math
4 dialect "mvel"
5
6 expander Test01.dslr
7
8 rule "Rule 1"
9
10 when
11 There is an employeee
12 then
13 Show name
14 end
  
```

dslr file

The image shows an IDE window displaying a DSL file named 'Test01.dsl'. The code is as follows:

```

1 package rules1
2 import com.nobleprog.FactModel.*;
3 import java.lang.Math
4 dialect "mvel"
5
6 expander Test01.dslr
7
8 rule "Rule 1"
9
10 when
11 There is an employeee
12 then
13 Show name
14 end
  
```

Below the code, a table titled 'Description:' maps language expressions to rule language mappings and objects:

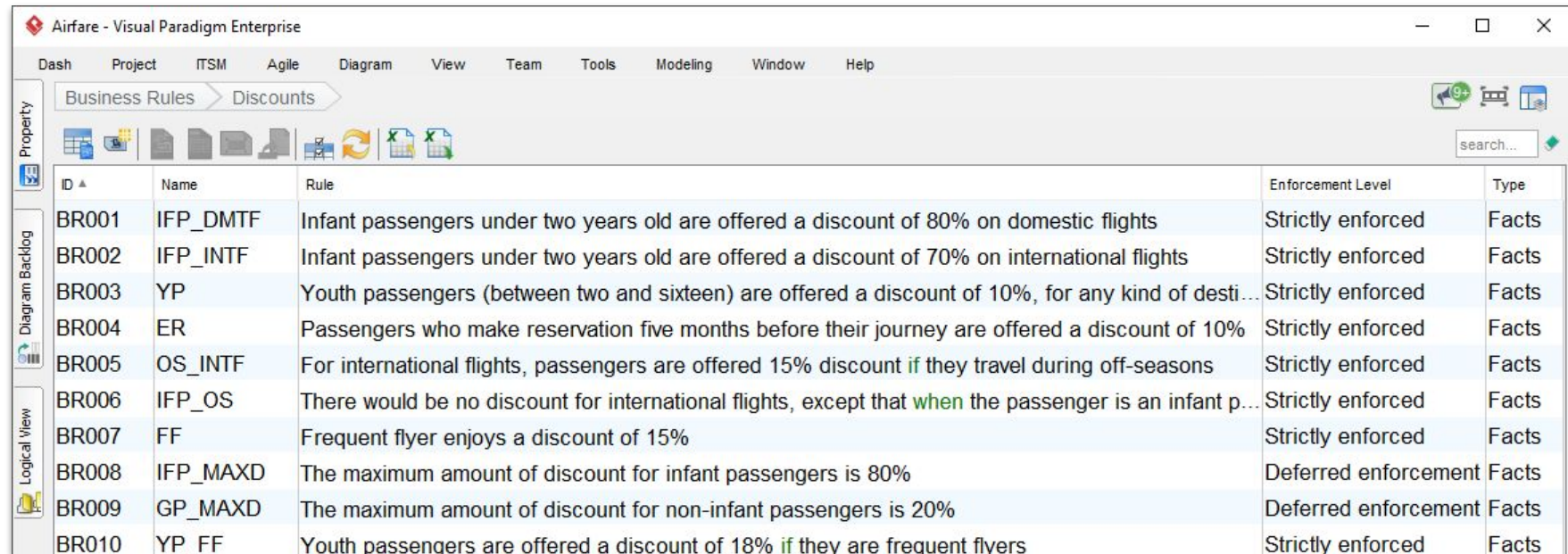
Language Expression	Rule Language Mapping	Object	Scope
There is an employee	e: Emp()		[condition]
Show name	System.out.println(e.name)		[consequence]

dsl file

# BUSINESS RULES

Business rules are directives that define an organization's business activities.

- Influence business processes
- Apply to people, processes, corporate behavior and computing systems in an organization
- Legal constraints and obligations



The screenshot shows the 'Airfare - Visual Paradigm Enterprise' application window. The interface includes a menu bar (Dash, Project, ITSM, Agile, Diagram, View, Team, Tools, Modeling, Window, Help) and a toolbar with various icons. A search bar is visible in the top right corner. The main area displays a table of business rules under the 'Business Rules' and 'Discounts' tabs. The table has five columns: ID, Name, Rule, Enforcement Level, and Type. The rules listed are:

ID	Name	Rule	Enforcement Level	Type
BR001	IFP_DMTF	Infant passengers under two years old are offered a discount of 80% on domestic flights	Strictly enforced	Facts
BR002	IFP_INTF	Infant passengers under two years old are offered a discount of 70% on international flights	Strictly enforced	Facts
BR003	YP	Youth passengers (between two and sixteen) are offered a discount of 10%, for any kind of desti...	Strictly enforced	Facts
BR004	ER	Passengers who make reservation five months before their journey are offered a discount of 10%	Strictly enforced	Facts
BR005	OS_INTF	For international flights, passengers are offered 15% discount if they travel during off-seasons	Strictly enforced	Facts
BR006	IFP_OS	There would be no discount for international flights, except that when the passenger is an infant p...	Strictly enforced	Facts
BR007	FF	Frequent flyer enjoys a discount of 15%	Strictly enforced	Facts
BR008	IFP_MAXD	The maximum amount of discount for infant passengers is 80%	Deferred enforcement	Facts
BR009	GP_MAXD	The maximum amount of discount for non-infant passengers is 20%	Deferred enforcement	Facts
BR010	YP_FF	Youth passengers are offered a discount of 18% if they are frequent flyers	Strictly enforced	Facts

## MAPPING

Process models

Process mining

Triggers

Participants

Inputs and outputs

Domain model

DSL

**Business rules**

Decision tables

Decision models

# DECISION TABLES

## MAPPING

Process models

Process mining

Triggers

Participants

Inputs and outputs

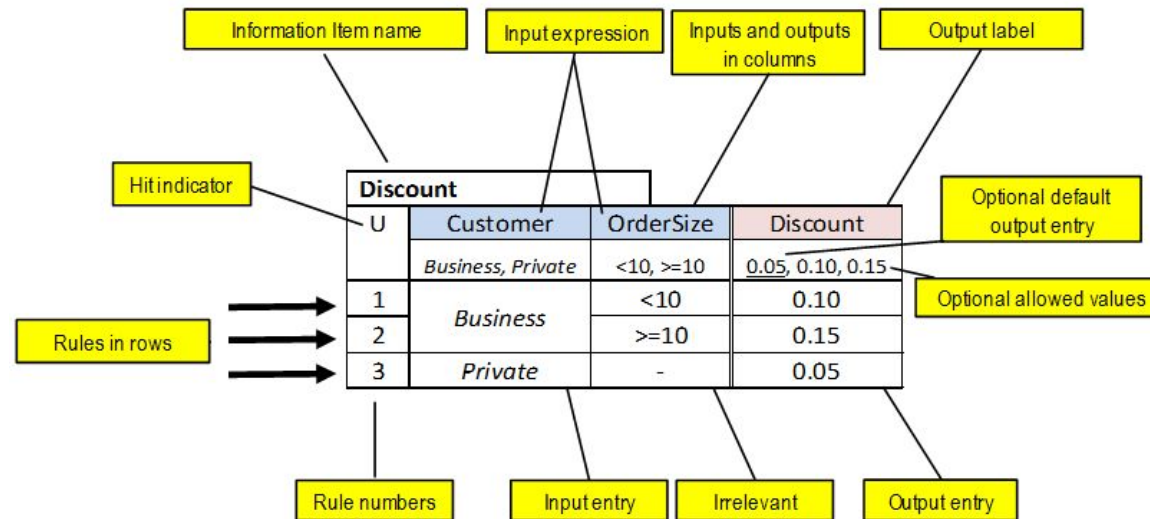
Domain model

DSL

Business rules

**Decision tables**

Decision models



Types of decisions

- Selection (routing)
- Scoring
- Categorizing

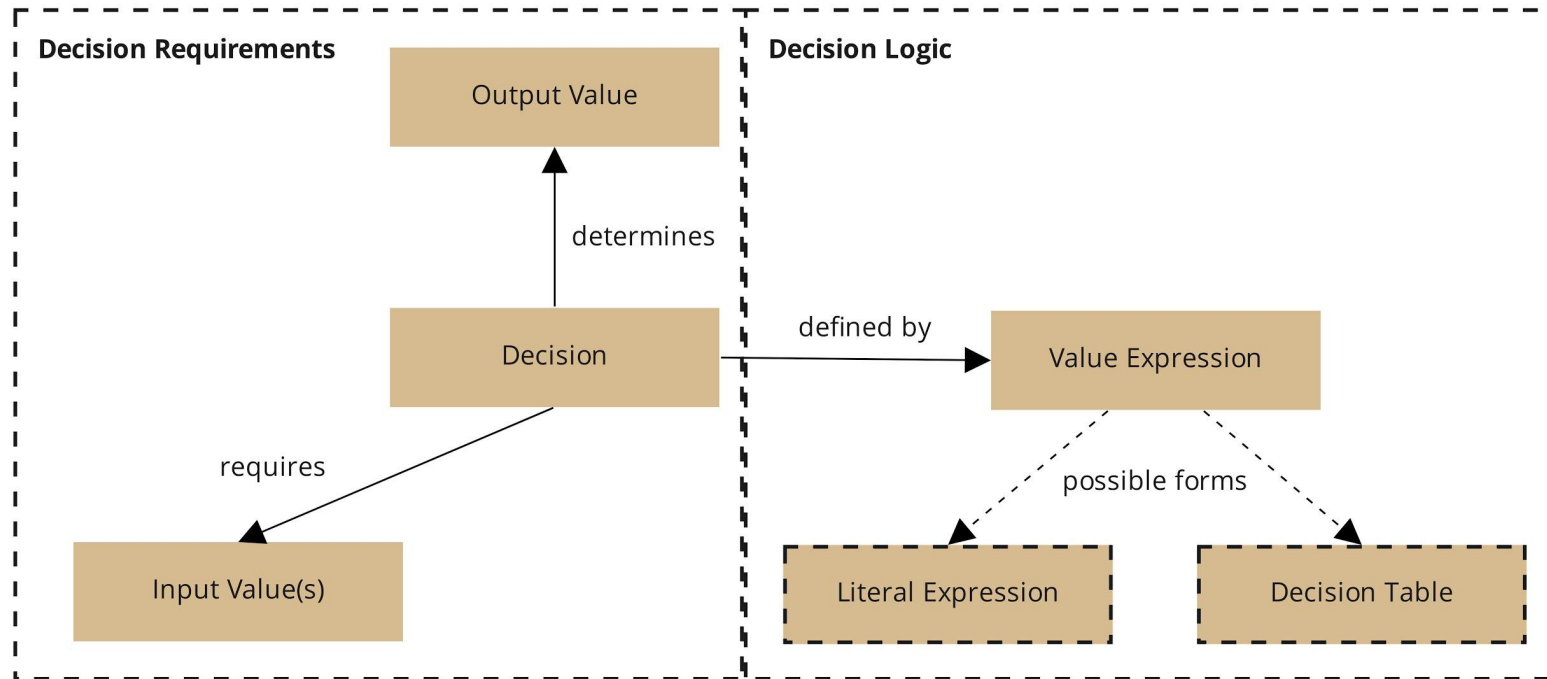
The decision doesn't take an action (no side effect), just determines a data value

# DECISION MODELS

## MAPPING

- Process models
- Process mining
- Triggers
- Participants
- Inputs and outputs
- Domain model
- DSL
- Business rules
- Decision tables

## Decision models

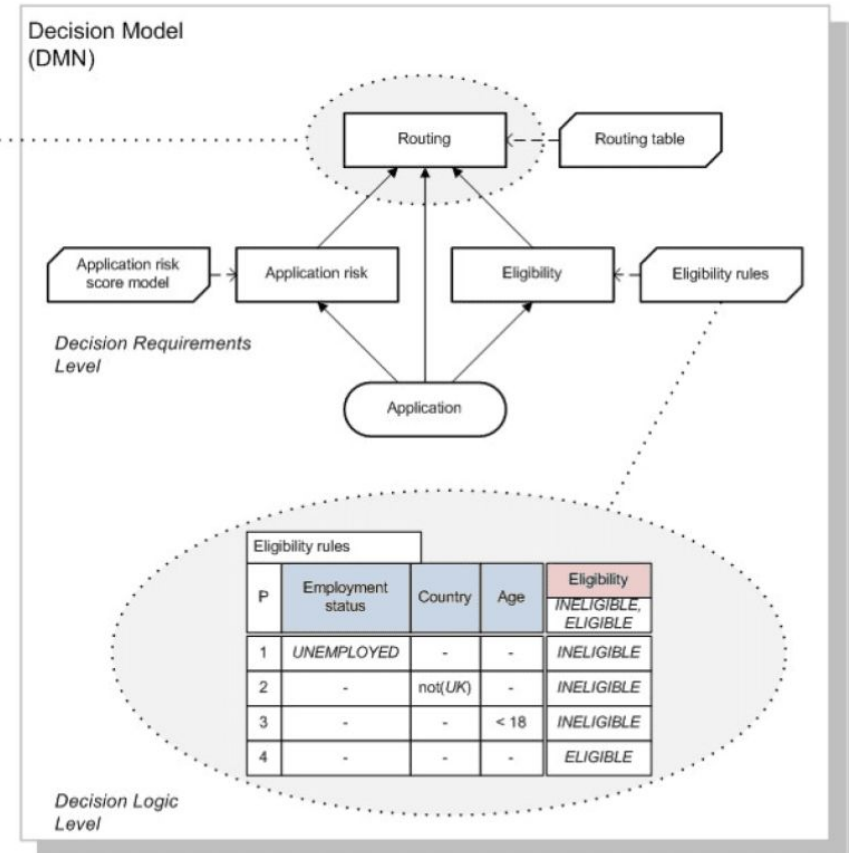
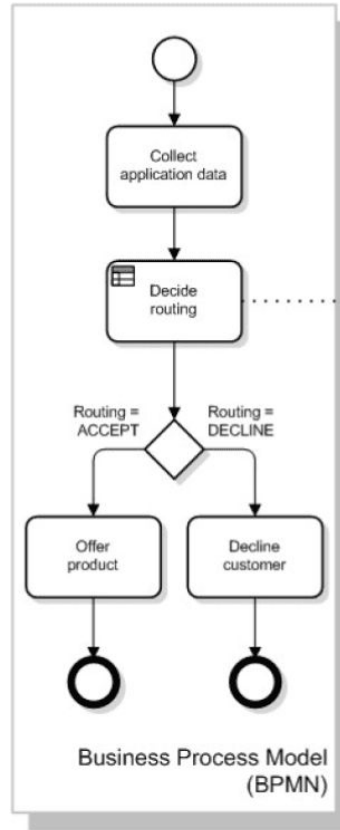
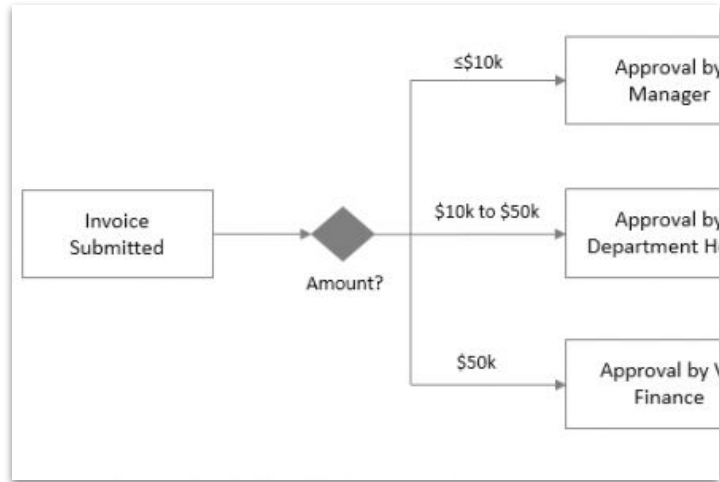




# DECISION MODELS

## MAPPING

- Process models
- Process mining
- Triggers
- Participants
- Inputs and outputs
- Domain model of business objects
- Business rules
- Decision tables
- Decision models**



[Learn DMN in 15 minutes](#)

[DMN and Kogito presentation](#)

[DMN.new editor](#)

## AUTOMATION

### Processes

Rules

Decisions

Service orchestration  
/ choreography

Event-driven  
processes

Decision streaming

Scheduled jobs

State machine

Serverless workflow

FaaS

Pipelines

Compensations

# PROCESSES

## System-centric processes

- Straight-through processes (request-response)
- Asynchronous processes - with wait states

## Human-centric processes

- Escalation processes
- Collaboration processes

## Adhoc processes (cases)

- Dynamic processes
- Data-heavy processes

*Event-driven processes*

*Serverless workflows*



## AUTOMATION

Processes

### Rules

Decisions

Service orchestration  
/ choreography

Event-driven  
processes

Decision streaming

Scheduled jobs

State machine

Serverless workflow

FaaS

Pipelines

Compensations

# RULES



## Kogito DRL Rules language

logicdrop Project: Insurance Sample

Home Data Rules Author Publish Test

Projects > Insurance Sample > Rules > Approve or Deny Policy >

### Driver is a single male

```
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
```

Save Analyze Collapse

Ruleset Source

- Freeform Rules +
- Calculate insurance extras
- Calculate insurance price
- Driver has another car
- Driver has prior claims
- Driver is a single male**
- Driver is married with chil...

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

Processes

Rules

### Decisions

Service orchestration  
/ choreography

Event-driven  
processes

Decision streaming

Scheduled jobs

State machine

Serverless workflow

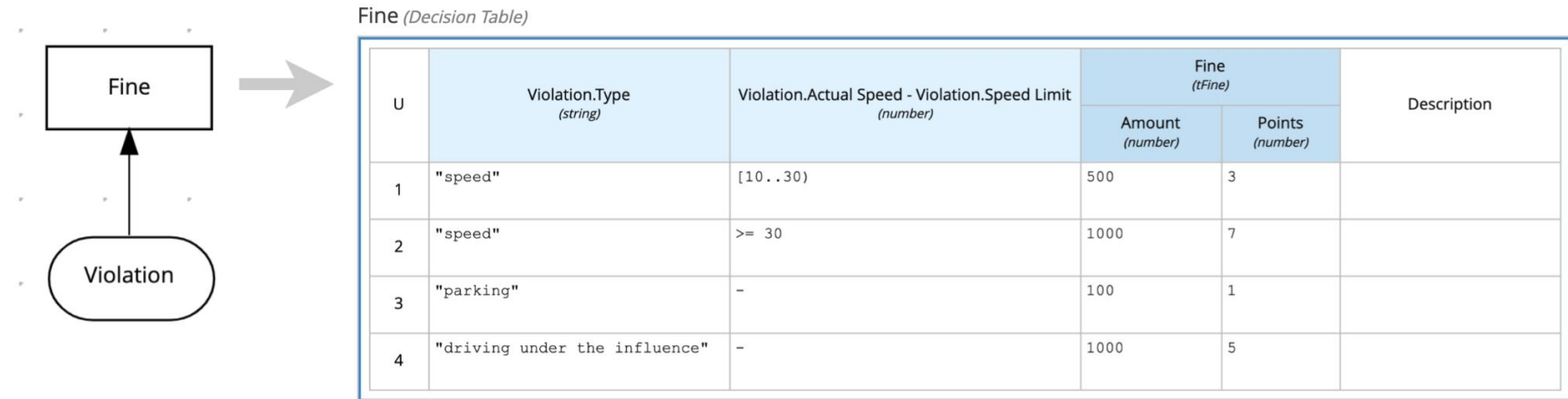
FaaS

Pipelines

Compensations

# DECISIONS

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

Processes

Rules

Decisions

**Service orchestration / choreography**

Event-driven processes

Decision streaming

Scheduled jobs

State machine

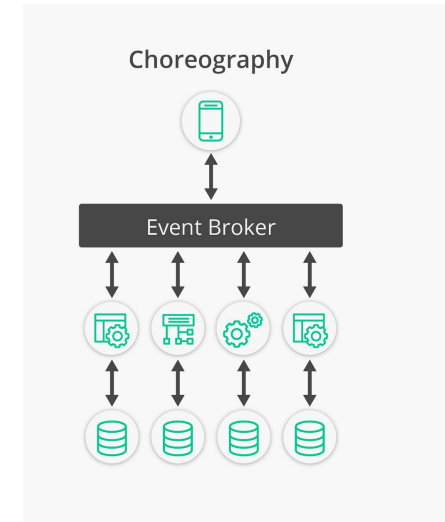
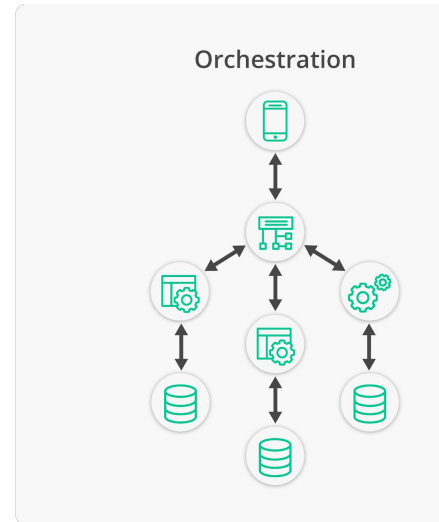
Serverless workflow

FaaS

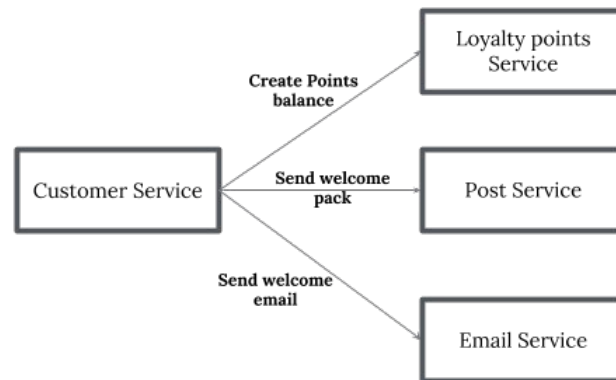
Pipelines

Compensations

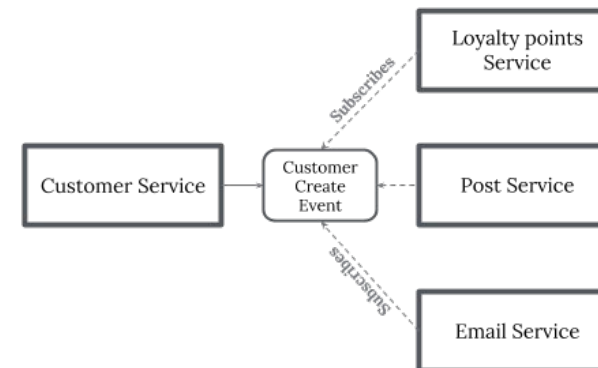
# SERVICE ORCHESTRATION / CHOREOGRAPHY



Orchestration Example



Choreography Example



## AUTOMATION

Processes

Rules

Decisions

Service orchestration  
/ choreography

**Event-driven  
processes**

Decision streaming

Scheduled jobs

State machine

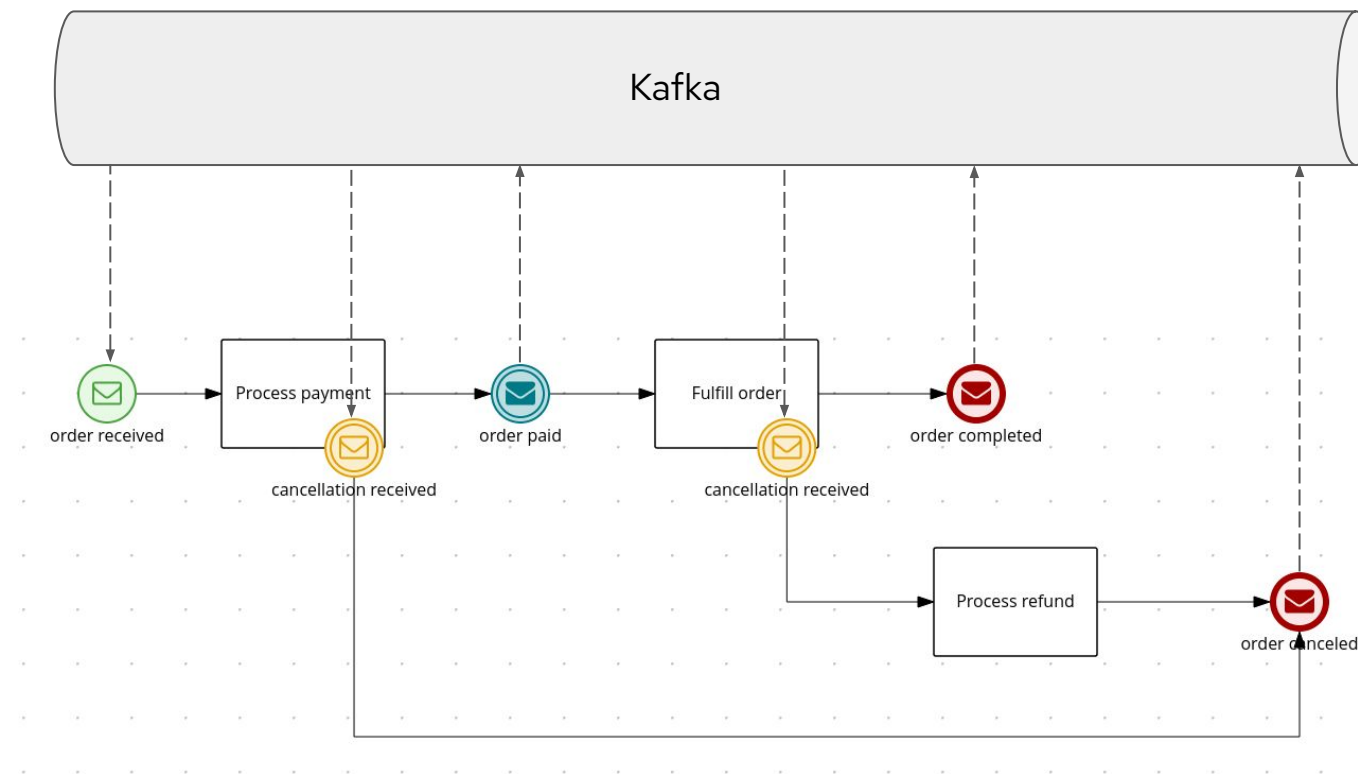
Serverless workflow

FaaS

Pipelines

Compensations

# EVENT-DRIVEN PROCESSES



## AUTOMATION

Processes

Rules

Decisions

Service orchestration  
/ choreography

Event-driven  
processes

### Decision streaming

Scheduled jobs

State machine

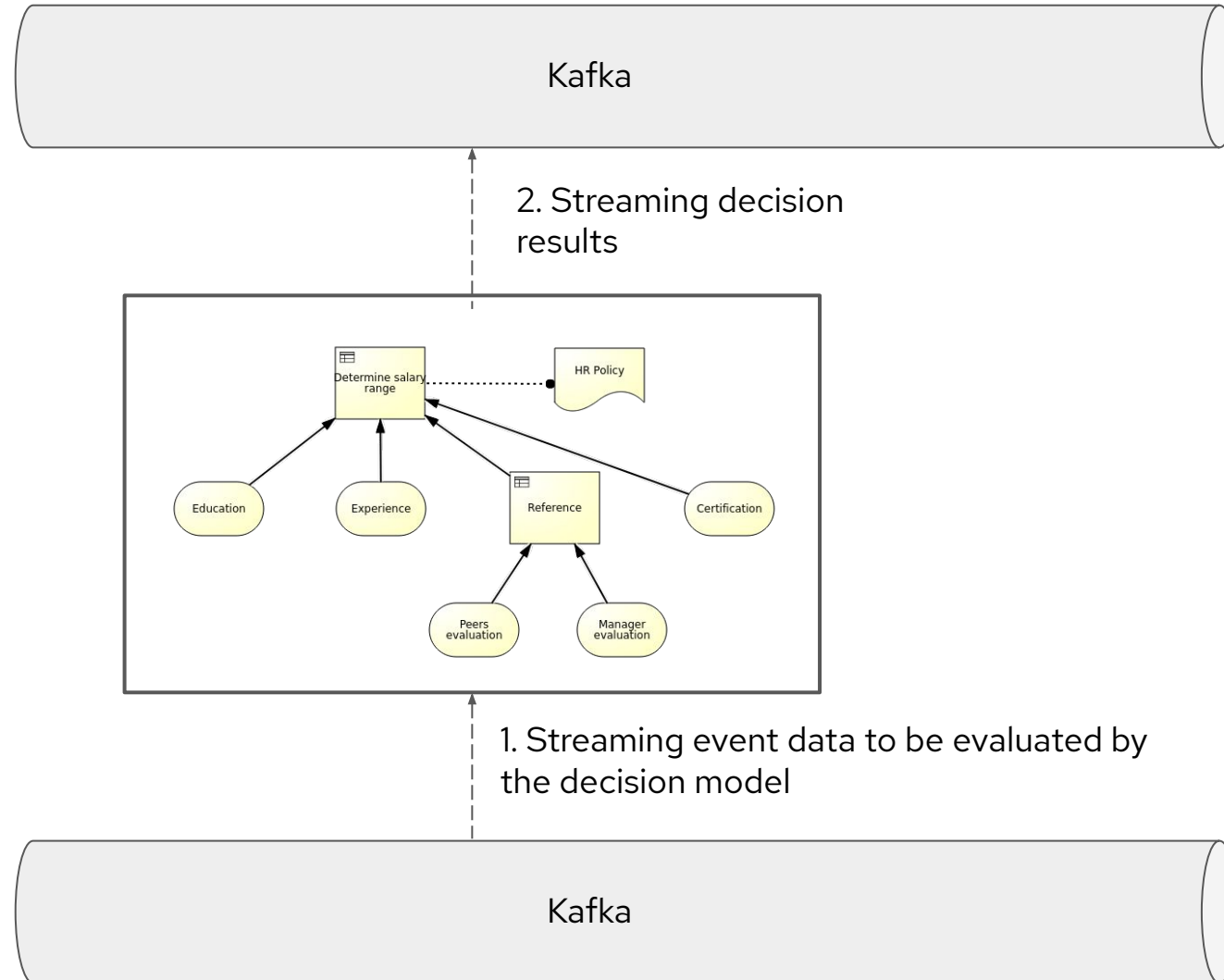
Serverless workflow

FaaS

Pipelines

Compensations

# DECISION STREAMING



## AUTOMATION

Processes

Rules

Decisions

Service orchestration  
/ choreography

Event-driven  
processes

Decision streaming

**Scheduled jobs**

State machine

Serverless workflow

FaaS

Pipelines

Compensations

# SCHEDULED JOBS

- Repetitive tasks - every day, every hour, at the end of each month, etc. - avoid peak loads by distributing load throughout the period of time
- Timouts, reminders
- Retries after failures - delay, max attempts

[Retries in Serverless Workflows](#)

[Timers in jBPM](#)

## AUTOMATION

Processes

Rules

Decisions

Service orchestration  
/ choreography

Event-driven  
processes

Decision streaming

Scheduled jobs

**State machine**

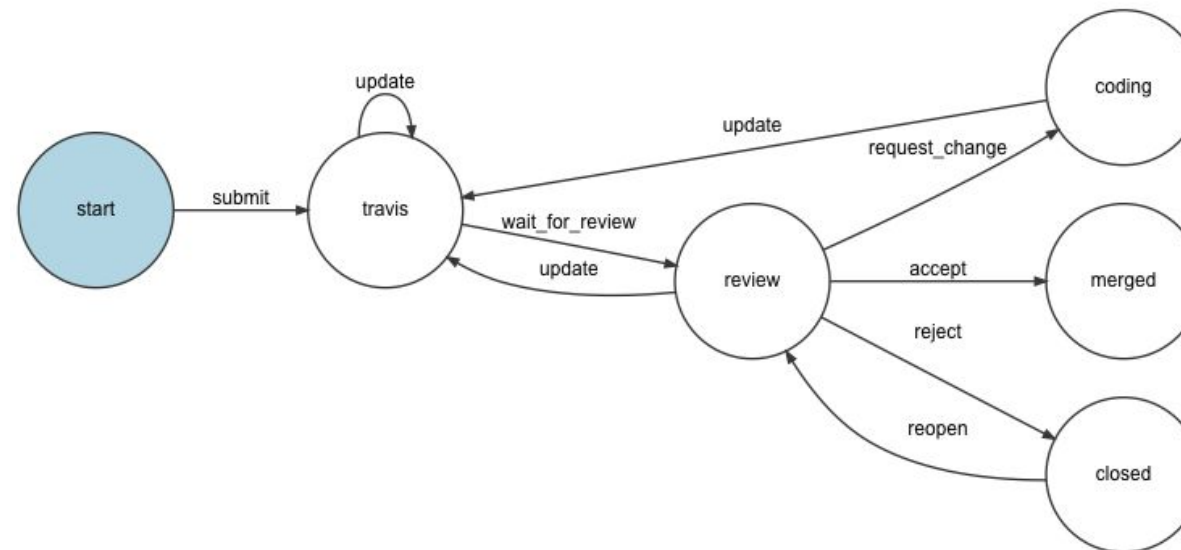
Serverless workflow

FaaS

Pipelines

Compensations

# STATE MACHINE



*Not a domain logic*

States in processes are moments when a process persists its state to the database

- When waiting on asynchronous task to receive a response to continue - for example waiting for form data from a user, waiting for an external system to finish processing, etc.
- When a process starts / finishes

# AUTOMATION

Processes

Rules

Decisions

Service orchestration / choreography

Event-driven processes

Decision streaming

Scheduled jobs

State machine

**Serverless workflow**

FaaS

Pipelines

Compensations

# SERVERLESS WORKFLOW



Workflows which do not actively wait to be triggered.

Event-based decisions example

```
1 {
2   "id": "eventbasedswitch",
3   "version": "1.0",
4   "name": "Event Based Switch Transitions",
5   "description": "Event Based Switch Transitions",
6   "start": "checkVisaStatus",
7   "events": [
8     {
9       "name": "visaApprovedEvent",
10      "type": "VisaApproved",
11      "source": "visaCheckSource"
12     },
13     {
14       "name": "visaRejectedEvent",
15       "type": "VisaRejected",
16       "source": "visaCheckSource"
17     }
18   ],
19   "states": [
20     {
21       "name": "CheckVisaStatus",
22       "type": "Switch",
23       "eventConditions": [
24         {
25           "eventRef": "visaApprovedEvent",
26           "transition": "HandleApprovedVisa"
27         },
28         {
29           "eventRef": "visaRejectedEvent",
30           "transition": "HandleRejectedVisa"
31         }
32       ]
33     }
34   ]
35 }
```

Generate workflow diagram

State Types and Border Colors:  
Event | Operation | Switch | Delay | Parallel | SubFlow | Inject | ForEach | CallBack

[Serverless Workflow](#)

[Intro to CNCF Serverless Workflow](#)

[AWS Step Functions](#)



## AUTOMATION

Processes

Rules

Decisions

Service orchestration  
/ choreography

Event-driven  
processes

Decision streaming

Scheduled jobs

State machine

Serverless workflow

**FaaS**

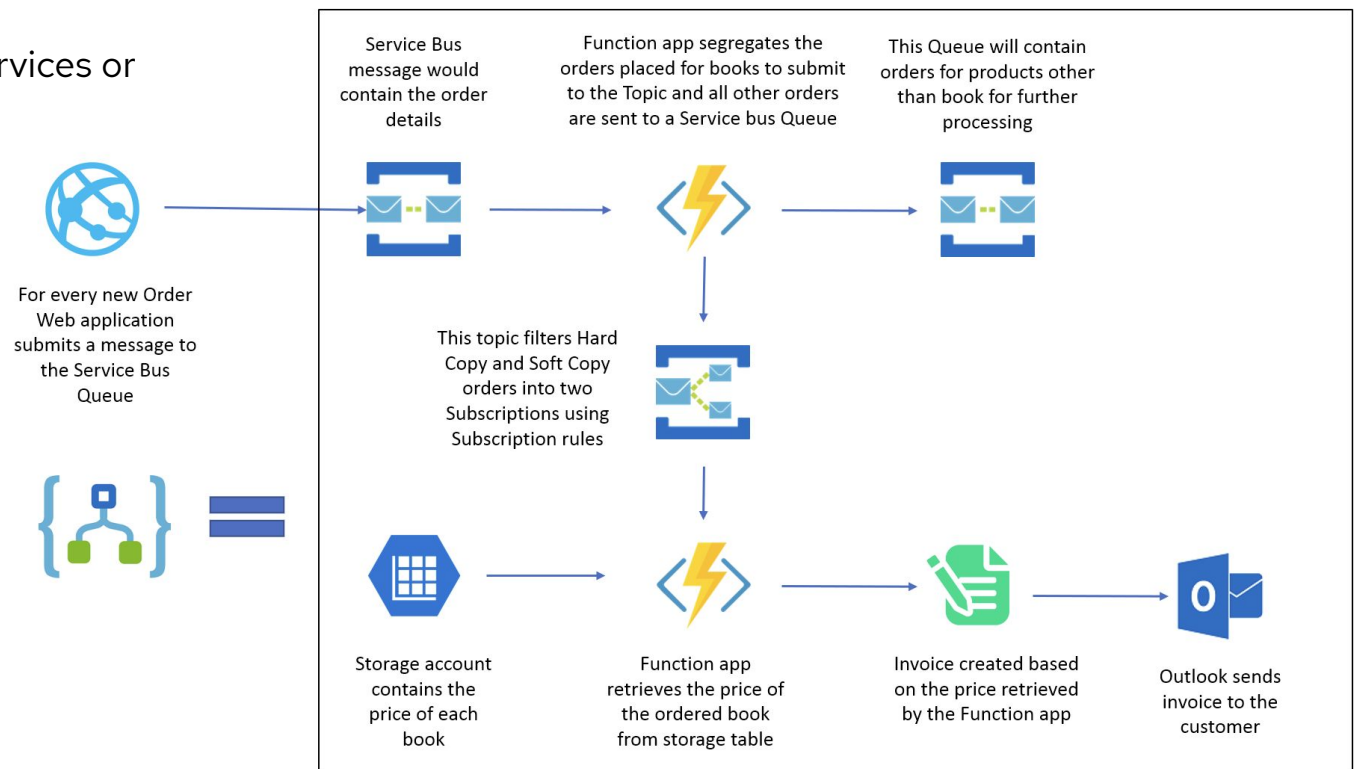
Pipelines

Compensations

# FAAS = Function as a Service

A serverless function is a small, discrete, and reusable chunk of code that:

- Is short-lived
- Is not a daemon (long-running)
- Does not publish TCP services
- Is not stateful
- Makes use of your existing services or third-party resources
- Executes in a few seconds



## AUTOMATION

Processes

Rules

Decisions

Service orchestration / choreography

Event-driven processes

Decision streaming

Scheduled jobs

State machine

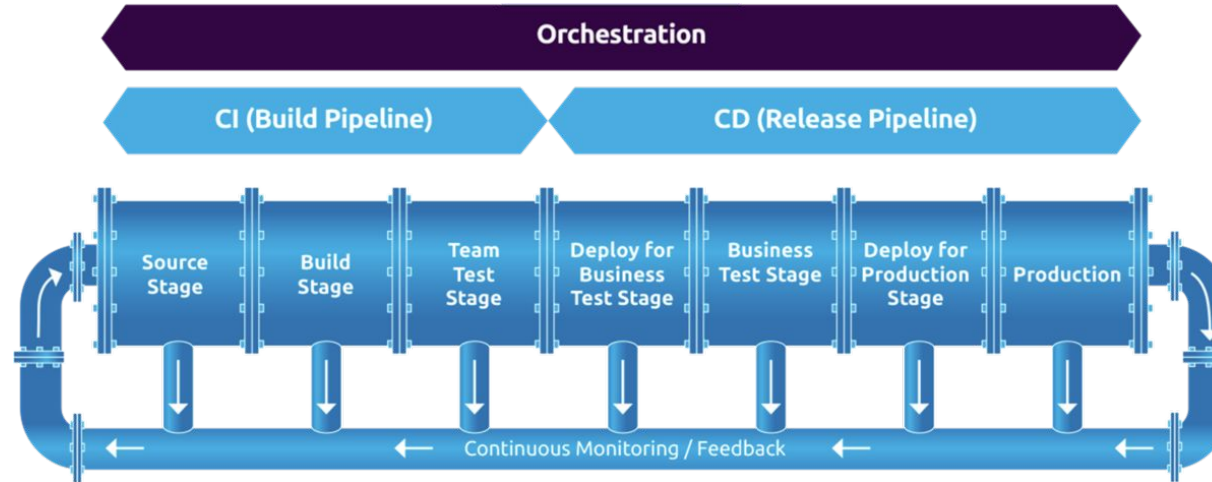
Serverless workflow

FaaS

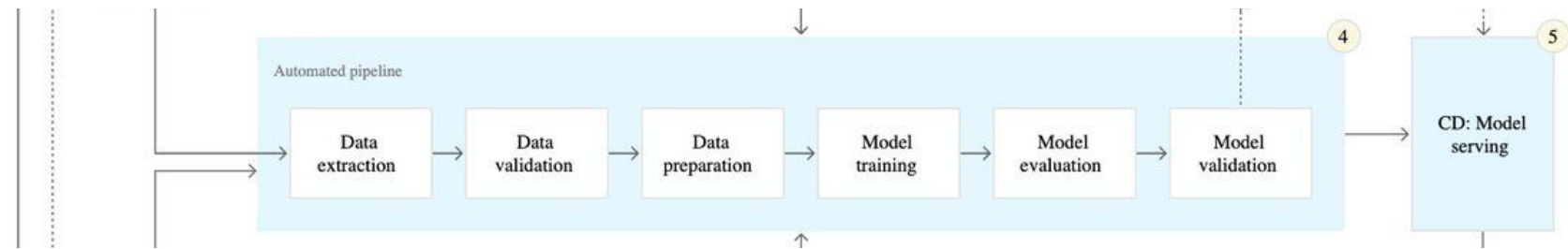
**Pipelines**

Compensations

# PIPELINES



Automation pipelines in  
Machine learning



## AUTOMATION

Processes

Rules

Decisions

Service orchestration  
/ choreography

Event-driven  
processes

Decision streaming

Scheduled jobs

State machine

Serverless workflow

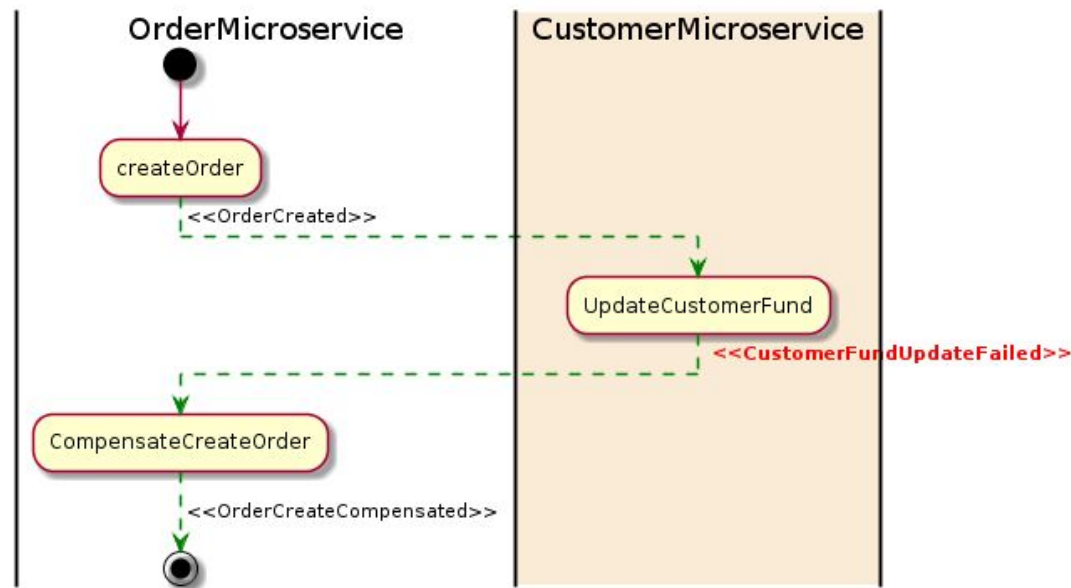
FaaS

Pipelines

**Compensations**

# COMPENSATIONS

Saga pattern for microservice architectures



[Simple Kogito BPMN example](#)

[Saga pattern powered by Kogito](#)

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.

# BPI AND BPR

## OPTIMIZATION

### BPI and BPR

RPA

Chatbots

Business activity  
monitoring

Process simulation

Task assignments

Constraint planning

Machine learning

Scorecards

### Business Process Improvement

- On-going effort
- Improvement of existing process
- Limited organizational change
- Requires an incremental change in mind-set

Use BPI when:

1. As-is process is already mapped/documentated
2. As-is process fundamentally works but not well enough with some areas in need of improvement
3. Your focus is the process – not on implementing an overarching business strategy.

### Business Process Reengineering

- Project-based effort
- Build process from scratch (whiteboard)
- Greater organizational change
- Requires a fundamental change in mind-set

Use BPR when:

1. As-is process is redundant or in need of a rethink and redesign
2. The as-is process fundamentally no longer works and a major overhaul is required.
3. The company focuses is the overall strategy to be achieved, rather tasks or process optimization

# RPA = Robotic Process Automation

## OPTIMIZATION

BPI and BPR

### RPA

Chatbots

Business activity  
monitoring

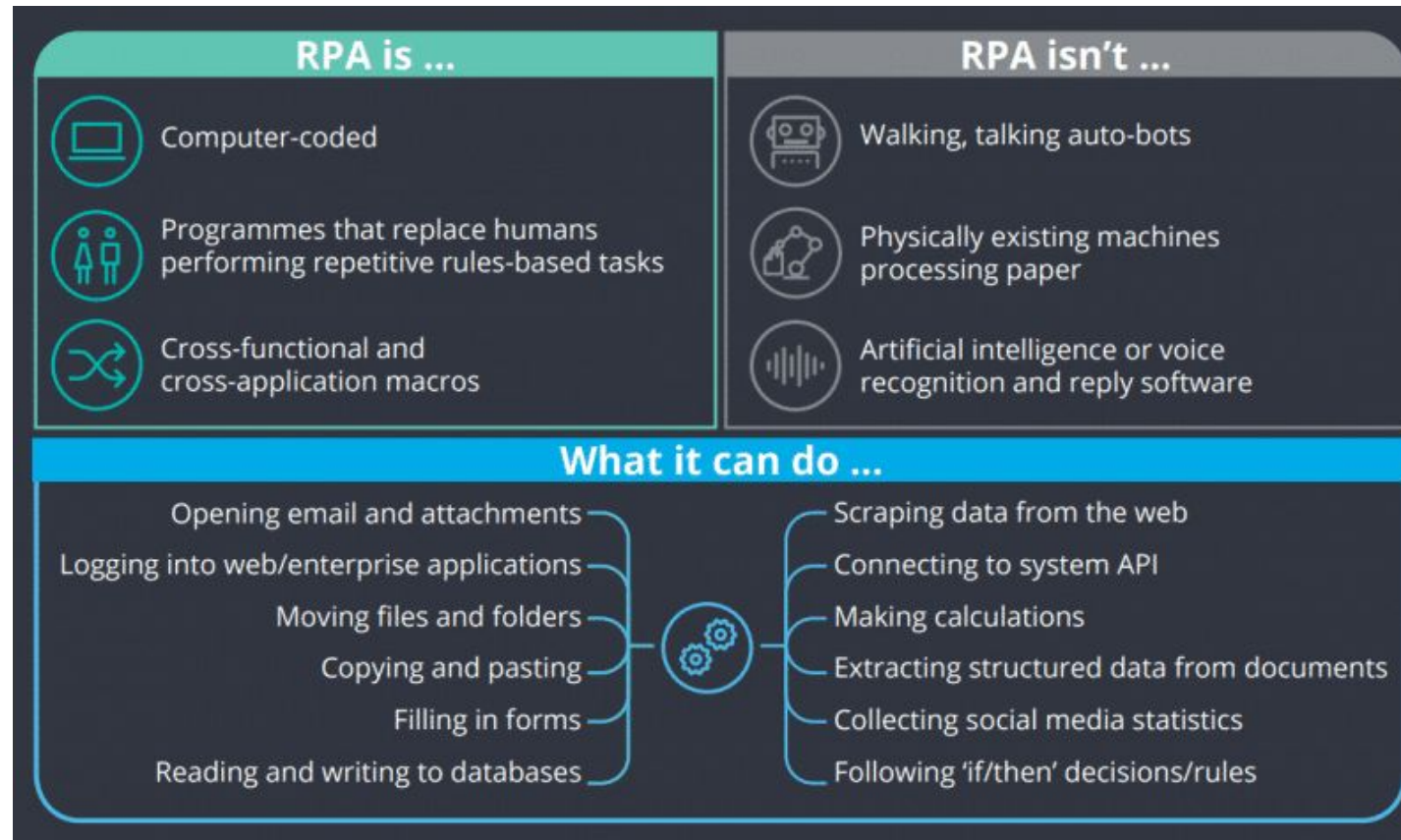
Process simulation

Task assignments

Constraint planning

Machine learning

Scorecards



# CHATBOTS

## OPTIMIZATION

BPI and BPR

RPA

### Chatbots

Business activity monitoring

Process simulation

Task assignments

Constraint planning

Machine learning

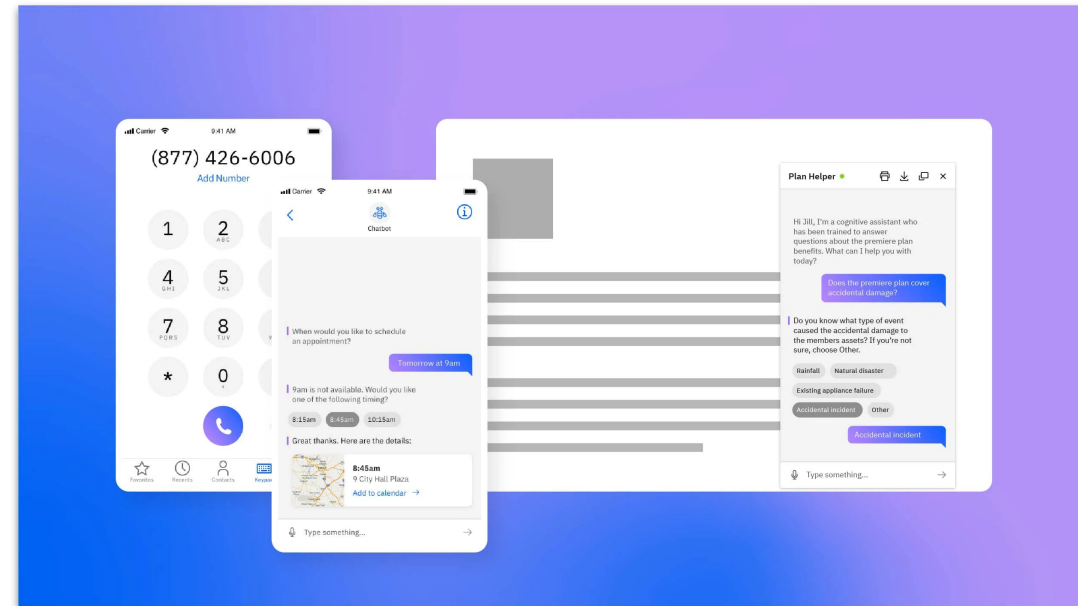
Scorecards

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

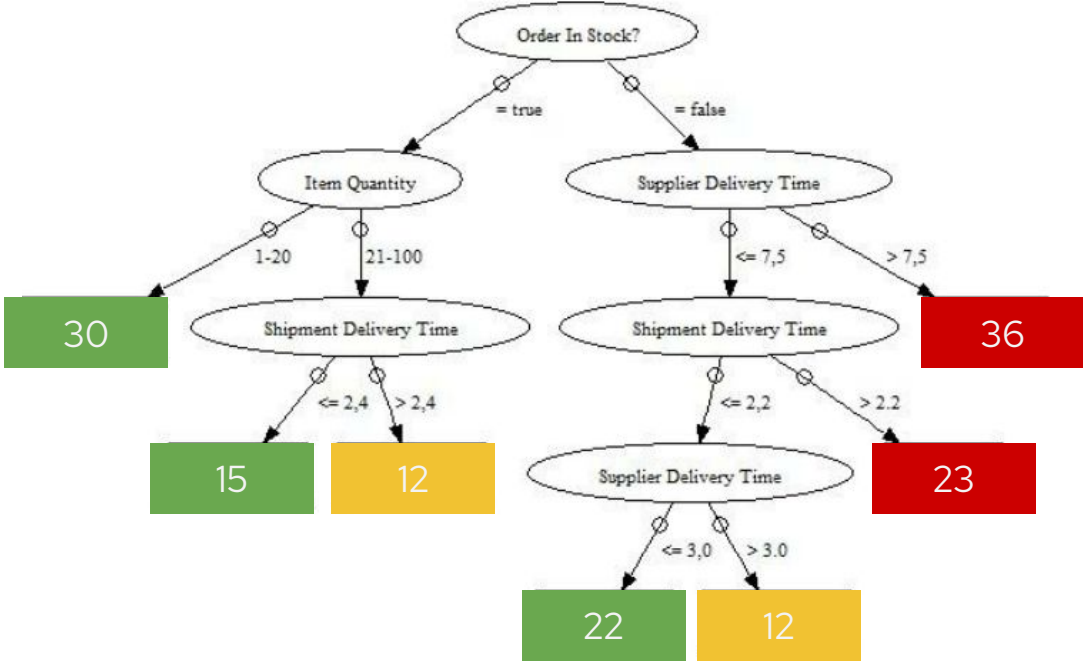


# BUSINESS ACTIVITY MONITORING

- OPTIMIZATION
- BPI and BPR
- RPA
- Chatbots
- Business activity monitoring**
- Process simulation
- Task assignments
- Constraint planning
- Machine learning
- Scorecards



KPI Dependency Tree





# PROCESS SIMULATION

## OPTIMIZATION

BPI and BPR

RPA

Chatbots

Business activity monitoring

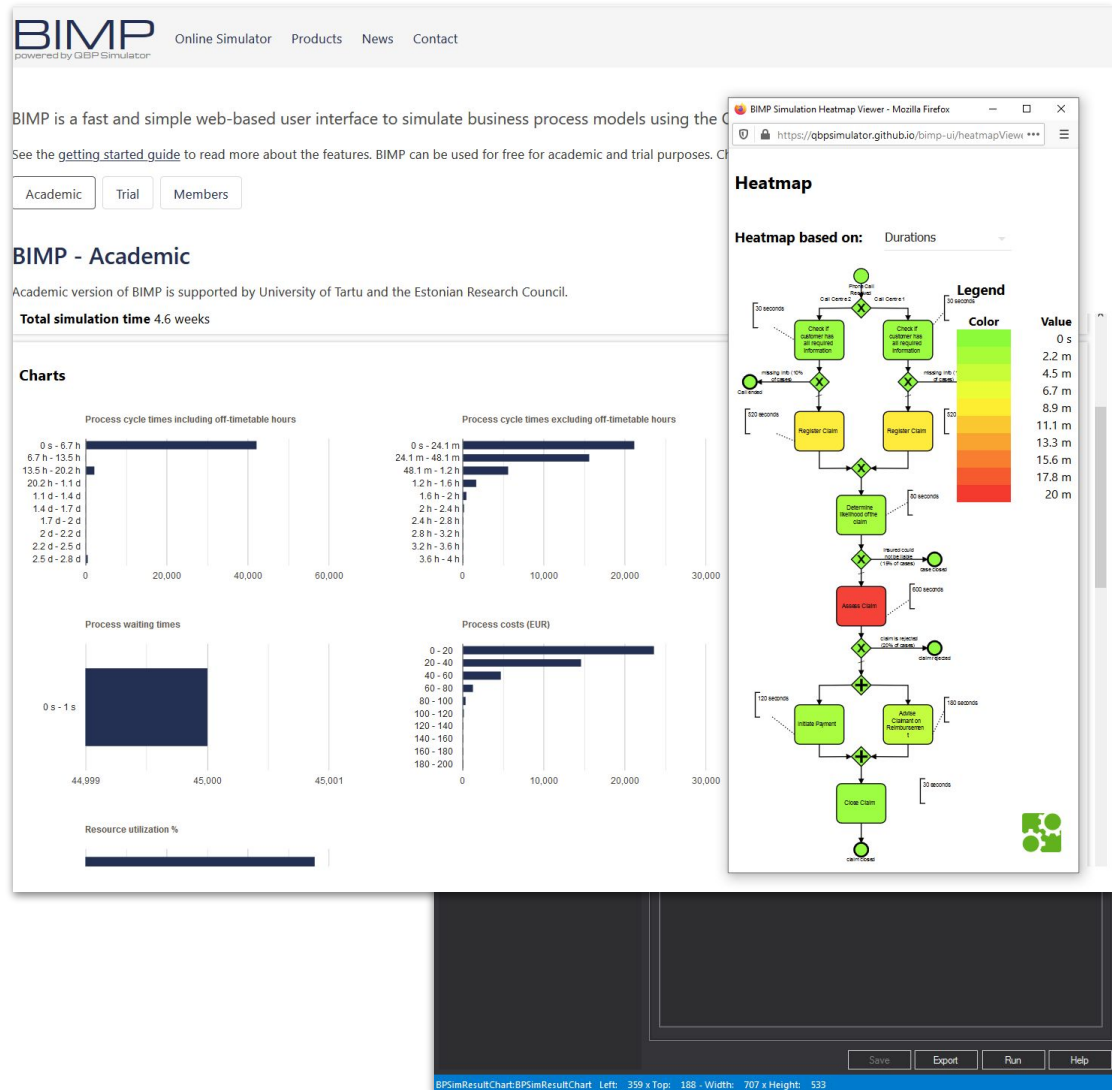
## Process simulation

Task assignments

Constraint planning

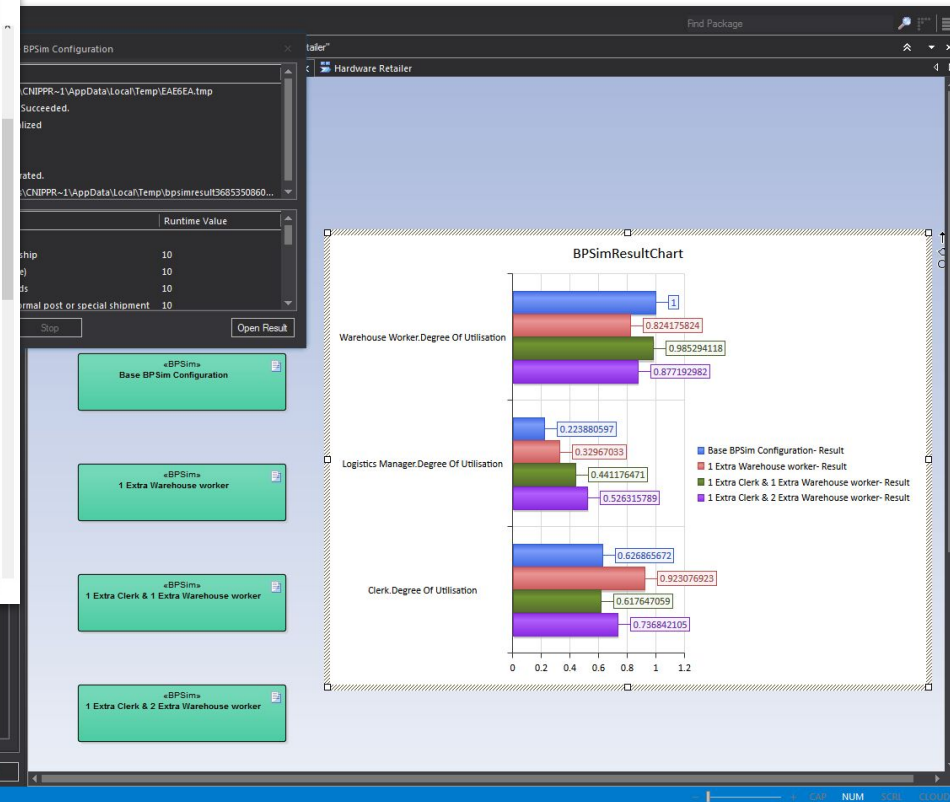
Machine learning

Scorecards



[BPSim Standard](#)

[BIMP Academic simulator](#)





# TASK ASSIGNMENTS

## OPTIMIZATION

BPI and BPR

RPA

Chatbots

Business activity monitoring

Process simulation

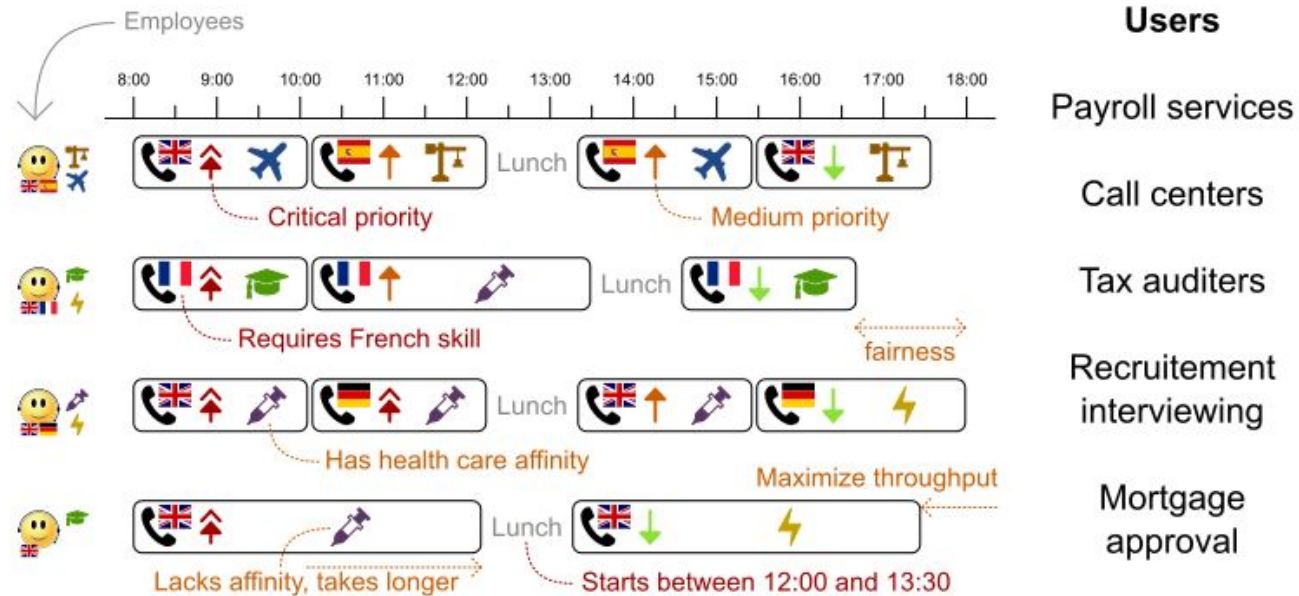
**Task assignments**

Constraint planning

Machine learning

Scorecards

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



[Assigning jBPM human tasks with OptaPlanner](#)

# CONSTRAINT PLANNING

Optimize goals with limited resources under constraints

## Optimize goals

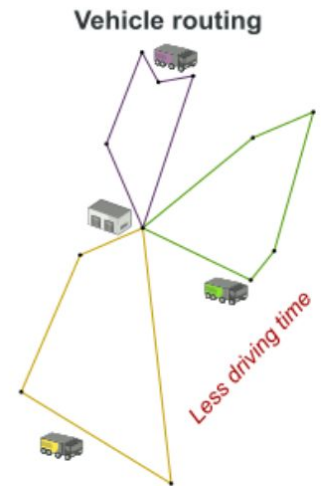
- 💰 Maximize profit
- 🌍 Minimize ecological footprint
- 😊 Maximize happiness of employees / customers
- ...

## With limited resources

- 👤 Employees
- 🚚 Assets (machines, buildings, vehicles, ...)
- 🕒 Time
- 💰 Budget

## Under constraints

- 👤 vs 🕒 Working hours
- 👤 vs 🚚 Skills / affinity
- 🚚 vs 🕒 Logistic conflicts
- ...



Employee rostering table showing a grid of employee availability for Sun, Mon, and Tue. The table is labeled "Employee rostering" and "per employees" on the left. The columns are Sun, Mon, and Tue, with sub-columns for 6, 14, and 22. The rows represent different employees, each with a unique emoji icon. The cells contain either a sun icon (indicating work) or the word "Free".

	Sun			Mon			Tue		
	6	14	22	6	14	22	6	14	22
👤1		☀️		☀️			Free		
👤2		☀️		Free			Free	☀️	
👤3	🌙			Free			Free		
👤4	Free			☀️			☀️		
👤5	Free			🌙			🌙		

[OptaPlanner Constraint satisfaction solver](#)

[Vaccination appointment scheduling optimization with OptaPlanner](#)

## OPTIMIZATION

BPI and BPR

RPA

Chatbots

Business activity monitoring

Process simulation

Task assignments

**Constraint planning**

Machine learning

Scorecards

# MACHINE LEARNING

## OPTIMIZATION

BPI and BPR

RPA

Chatbots

Business activity monitoring

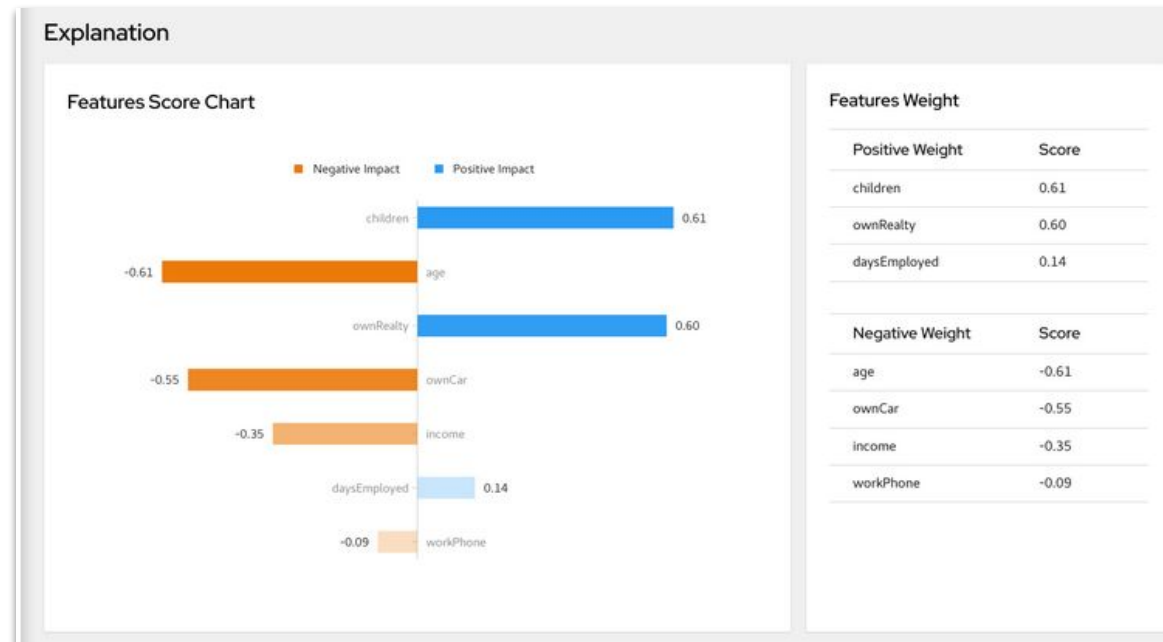
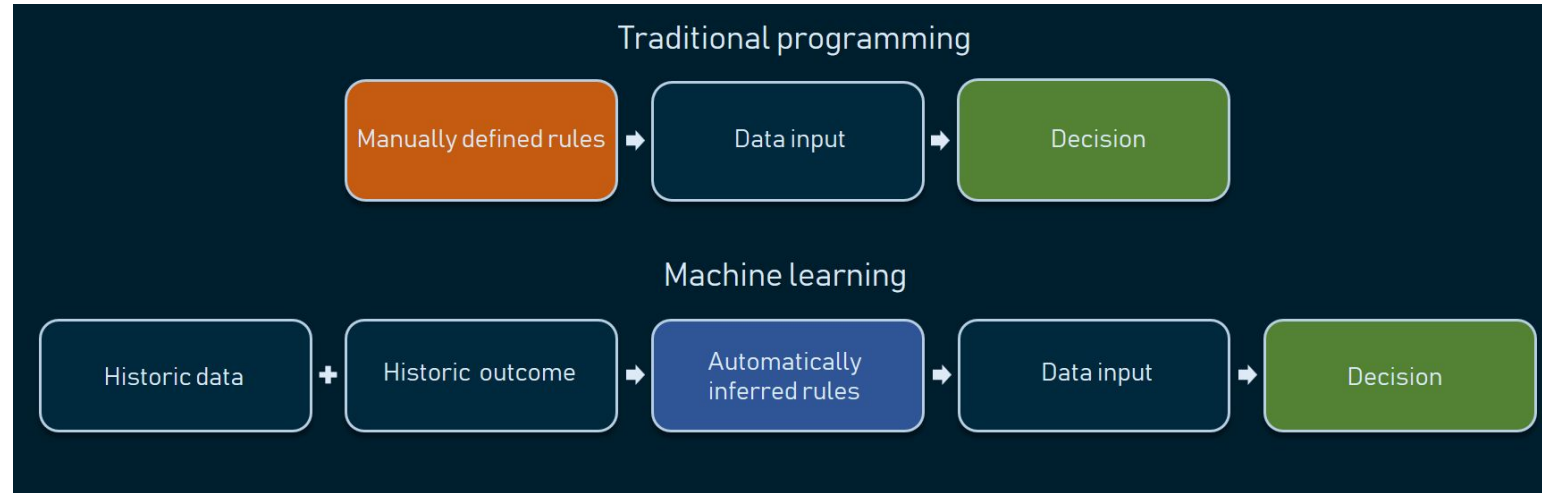
Process simulation

Task assignments

Constraint planning

**Machine learning**

Scorecards



[TrustyAI](#)

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

# SCORECARDS AND PREDICTIONS

## OPTIMIZATION

BPI and BPR

RPA

Chatbots

Business activity  
monitoring

Process simulation

Task assignments

Constraint planning

Machine learning

**Scorecards**

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 right, 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 as follows:

- 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

# 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/sample-kogito.git>
- 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?