ASP Mining - Analytics Description

# Introduction

The scope of this document is to describe the logical steps and behaviour of the data processing behind the sample ASP Mining Starter pack and the included components, it’s not a source code documentation (which is available in other documents and the source code itself), but a higher level technical description of what the software does.

The ASP Mining Starter Pack has 3 layers:

1. IRIS Production: integrates mining systems data and transform it into IRIS Mining Canonical Model.
2. IRIS Analytics: performs calculations on top of the Canonical Model to generate higher value KPIs.
3. Visualization: using IRIS as a backend, implements a web-based application using standard web tools to navigate and visualize the IRIS Analytics results.



# IRIS Production: Connectors and Data Flow

The IRIS production consists of one or more Connectors that follow the pattern:

1. External System Connection
2. Service for data collection and transformation to Canonical Model
3. Operation for storage of new data into Canonical Model

The connectors also include a Task to keep updated the relevant cubes that are used with them.



One of the main tasks in the data transformations performed by the included connectors and any other connector, is the transformation **of the Timestamps** in the data source systems, which are not always standard types, but local timestamps relatives to a shift or to other non-standard DB time datatypes.

Since the Canonical Model uses only IRIS TimeDate types, this step is crucial for being able to navigate across different sections of the model with consistency and allow proper “drill through” in the cubes.

## MineCare Connector

### External System Connection

The External System connection of this connector if done using a JDBC connection to the MineCare database which runs on SQL Server (v2019+), the connector uses a standard IRIS SQL Inbound Adapter to perform this connection.

### Service for data Collection and transformation to Canonical Model

The Service of the MineCare connector brings information about multiple health signs of the mining equipment specified by the client into the MineCare software. This data is mainly used to track sensor related events such as: overspeed, valve pressures out of range, fuel levels, temperature of internal components, and many others.

Is important to note, that that the raw sensor data is not being collected by the connector, but the events that this data generates (i.e: one or more sensor values are out of a determined range), is the data that is being collected and saved in IRIS.



|  |  |  |
| --- | --- | --- |
| Source Table | Data obtained | Data Usage |
| OemEventValue | Numeric/Text value of an event | Analysis and Plotting |
| OemEventType | Type or category of the event | Filtering and categorization |
| OemInterface | Phyisical interface that made the reading in the truck | Link between Sensor and Equipment |
| EquipmentUnit | Equipment metadata (capacity, category, model) | Equipment Information |

### Storage of new data into Canonical Model data

The Operation in the MineCare connector saves data to the following Canonical Model Classes

|  |
| --- |
| Equipment.Equipment |
| Equipment.EquipmentCategory |
| Equipment.EquipmentModel |
| HWMonitoring.OEMEvent |
| HWMonitoring.OEMEventType |
| HWMonitoring.OEMInterface |
| HWMonitoring.OEMInterfaceMap |

## PowerView Connector

### External System Connection

The External System connection of this connector if done using a JDBC connection to the PowerView database which runs on SQL Server (v2019+), the connector uses a standard IRIS SQL Inbound Adapter to perform this connection.



|  |  |  |
| --- | --- | --- |
| Source Table | Data obtained | Data Usage |
| Hist\_eqmtlist | Equipment metadata (capacity, category, model) | Equipment Information |
| Hist\_exproot | Shifts information | Shift Type and Start Time |
| Hist\_reasontable | Reasons for Status Event changes | Details about events |
| Hist\_statusevents | Status events history for every equipment | Events Information |
| Hist\_dumps | History of material Dumps |  |
| Hist\_loads | History of material Loads |  |

## Canonical Model

The canonical model details can be found in the Canonical Data Model.pdf file.

# IRIS Analytics: DeepSee Cubes

The DeepSee cubes consolidate raw data and processed data from the Canonical Model, making it available to be accessed from the FrontEnd to the end users. These cubes are updated continuously by Scheduled Tasks.



The cubes provided by the Starter Pack are:

|  |  |  |
| --- | --- | --- |
| Cube Name | Contents | Usage |
| Equipment Cube | Equipment Information | General Information |
| OEMEvents Cube | OEM Events | Hardware Maintenance KPIs |
| Production Dump Events Cube | Material Dump Events | Production KPIs |
| Production Load Events Cube | Material Load Events | Production KPIs |
| Status Event Cube | Equipment State Changes only | State changes trazability |
| Status Events Sliced Cube | Equipmente State changes with interpolation over time | State changes KPIs (daily analysis) |
| Unified Events Cube | Production Events combined with Status Events | OEE and OEE related KPIs: Utilization, Capacity Performance, Time Performance |

<Explain Unified Events cube in detail>

## OEE Calculation

The Overall Equipment Efficiency KPI is a very known KPI used in many industries, since it indicates how efficiently the process equipment is being.

Calculating the OEE form a mine in the site side of the mine has been a challenge for many decades, not only for the information available but also for the way the standard mining systems have been designed. Being able to provide this dodgy KPI showcases 2 values of IRIS:

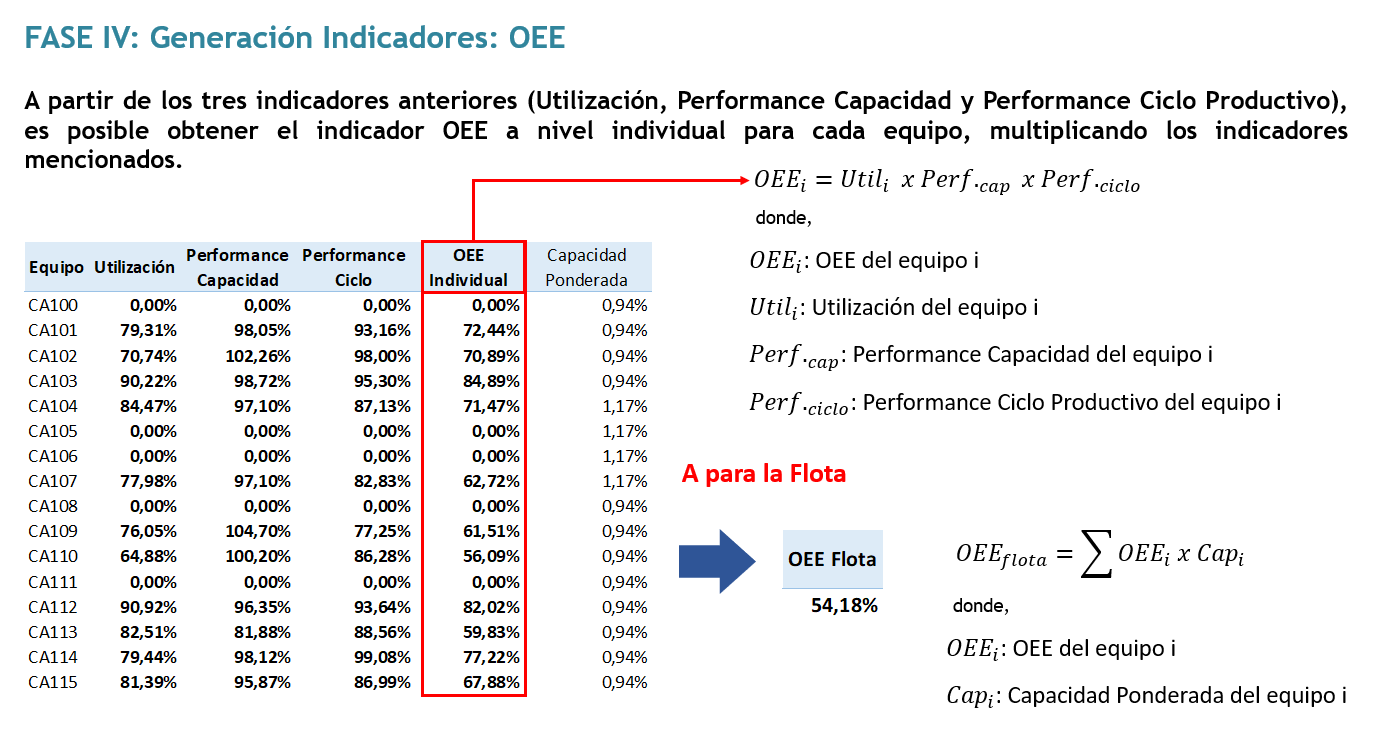
1. Simplicity to integrate an existing system to the data platform (e.g: connection to PowerView/MineCare)
2. Model oriented design that simplifies the KPI and value extraction from the data (e.g: being able to unify the time events in the UnifiedEventsCube)
3. Performance benefits that enables to calculate these new KPIs in realtime

For the ASP Mining starterpack, a different approach has been taking to calculate the OEE of the Mine like:

Where:

* Utilization: Percentage of time that the equipment was operative
* Capacity Performance: How many Tons was the equipment loaded with vs its nominal capacity.
* Cycle Time Performance: How fast the Load/Dump cycle is being completed for a given source/destination location vs previous same trips.

These 3 KPIs are conveniently precalculated in the UnifiedEventsCube to make it easier for the frontend to access these results.



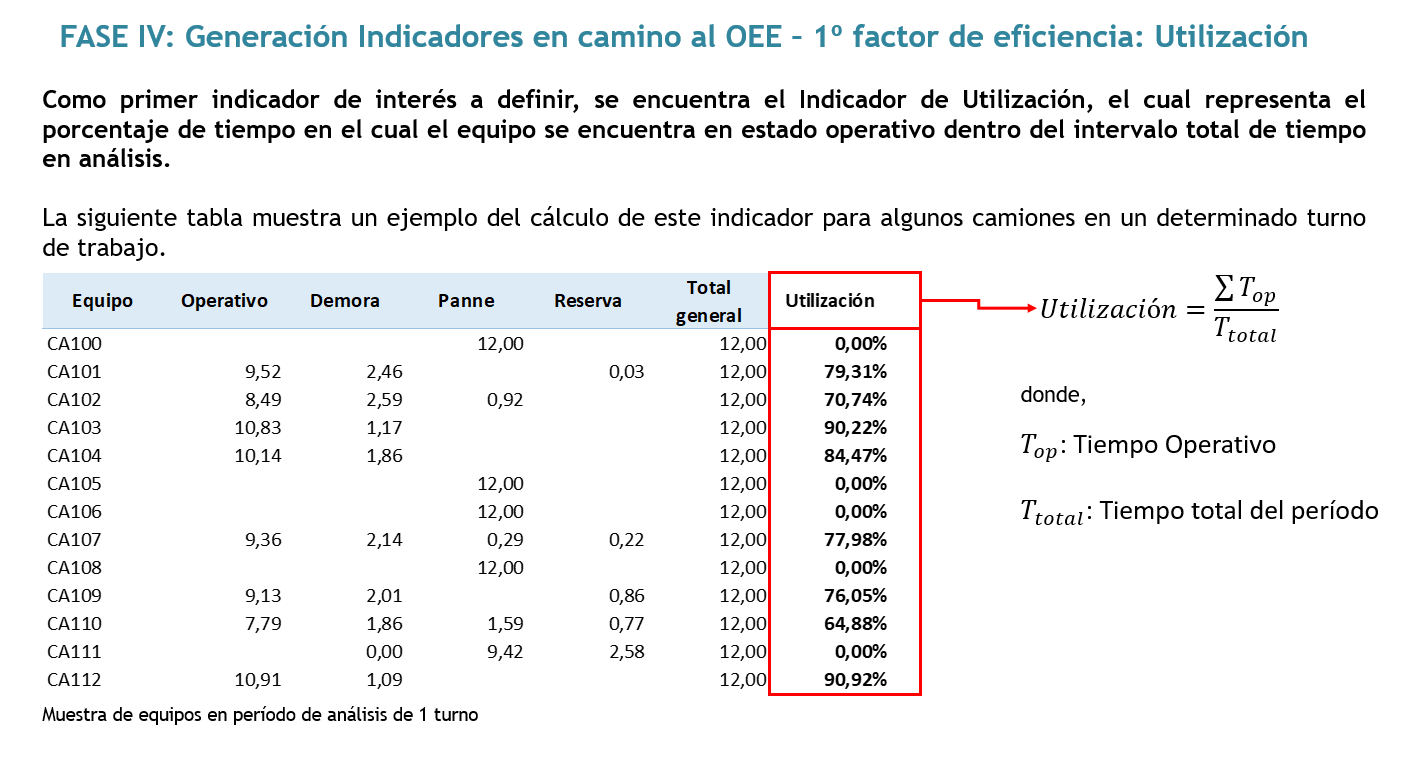
### Utilization

The equipment Utilization is calculated based on the time the equipment stays in state Operative vs other states. This state is defined in the StatusType Class of the canonical model. To calculate the total utilization, the StatusEventsSlicedCube and the UnifiedEventsCube can be used.

For more detail in the analysis the StatusEventsSlicedCube is preferred, since this cube provides 5 minute windows with all the details related to StatusEvents, where the time of all states inside this window (Operative, Delay, etc.) is calculated as PartialTime, allowing fast calculation and comparison of time durations for any given time range.

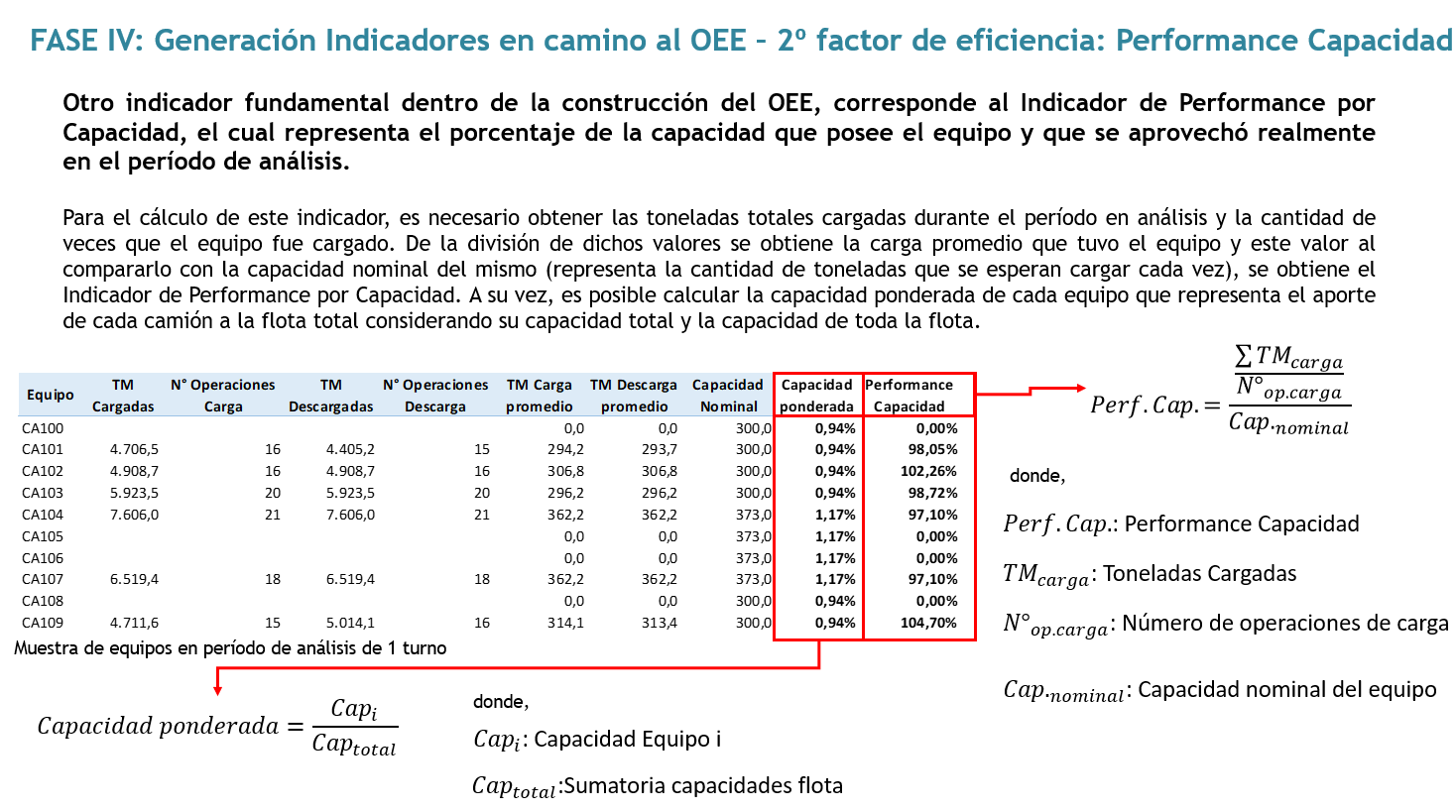
<math equation>

<equation using cube parameters>



### Capacity Performance

<math equation><equation using cube parameters>



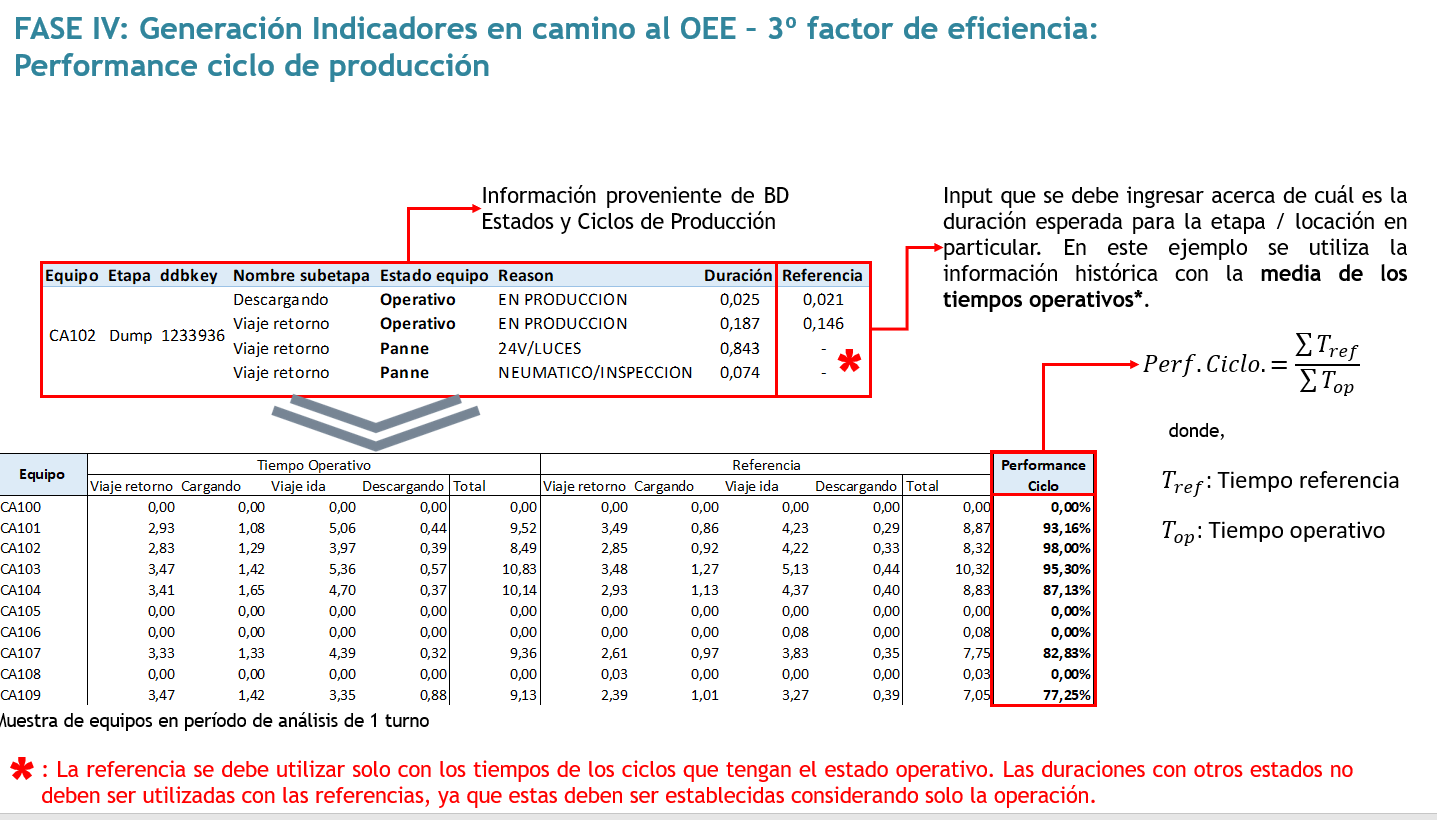
### Cycle Time Performance

<math equation>

<equation using cube parameters>

The mining Cycle in the open consists of: Material extraction on pit and Material dumping to its corresponding material bay.

The cycle time performance is calculated for each trip, comparing the last trip time vs its expected time, and the result is stored in the **UnifiedEventsCube** in the Calculated Dimension: Time Performance.

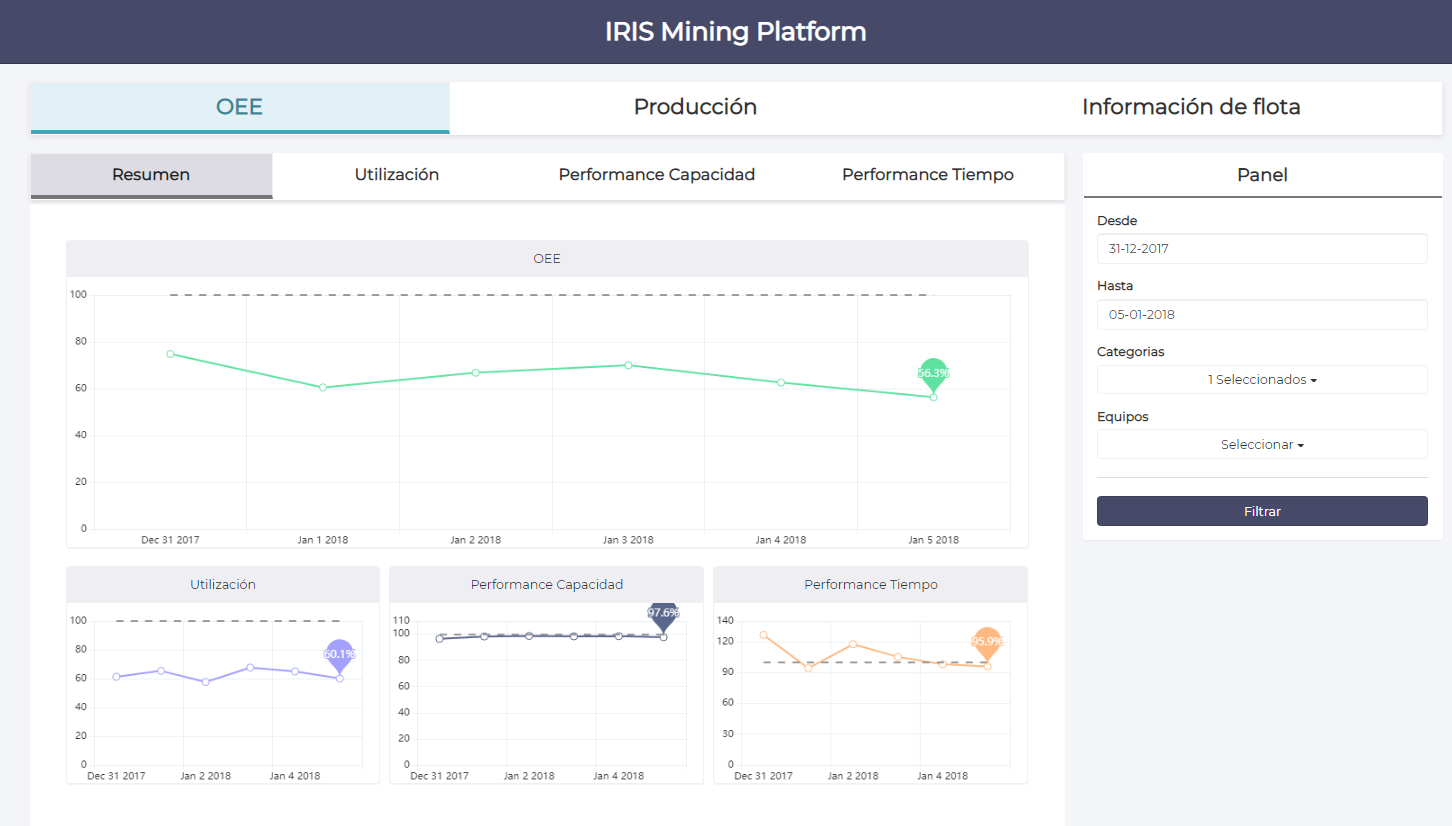


# Visualization: Web App

The Visualization frontend is an application based on AngularJS, currently using 2 libraries for graphics: Chart.js and ECharts.js.

The JavaScript code performs queries to the IRIS cubes using MDX queries through the MDX2JSON libraries.

Example of the visualizations made on JavaScript showing the resulting graphics of the OEE.



<Example of generation of 1 graphic>

<Project Structure>