W.R.A.P.
Project Overview

BE Seminar

Epameinondas Galatas

Motivation: GUI Strategy within ATS

Every application suffers from 2 potential risks:

- **Breaking**
  - Hardware **changes** (evolution / renovation)
  - Incompatible execution environment
- Difficult (if not impossible) to **maintain** & **extend**
  - Dependencies **unmaintained**
  - **Hiring** experts of older frameworks **difficult**

Result:

- Impact on **beam time**.
- Cannot adapt to changing needs and op requirements. **Decreased efficiency**

Solution: **Streamline** development and evolution of GUIs, **anticipate** problems

Types of applications & users

Equipment experts need **specific applications** to configure, control, maintain, tune visualize, diagnose, and monitor equipment.

Operation crews need **use-case-driven applications** to operate, optimise, and supervise accelerators and automate repetitive tasks.

However:

OP also needs to diagnose, and monitor equipment. CCM contains some applications not developed for them. Usability suffers.

System specific applications, by equipment groups, can target OP.

GUI development offering

The GUI strategy comes together with a GUI Development Offering. Key aims:

- Keep under **control** the potentially huge expense due to **obsolete technologies** that would require rewrite of the applications
- **Reduce** the **cost** of GUI **developments** in the long run, for both creation and **maintenance** of applications

Several coherent solutions:

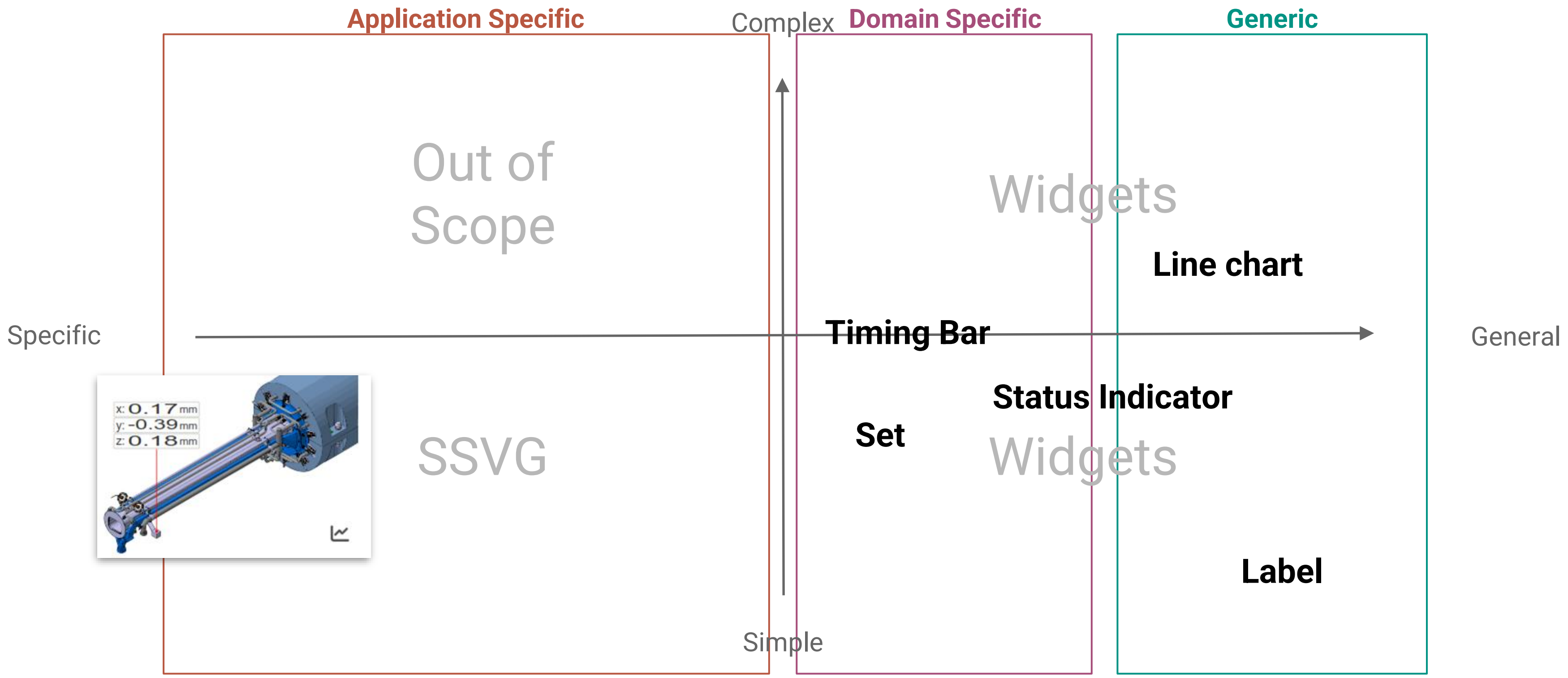
- 2 no-code application **platforms**: WRAP (low-code tbd) & NavPy
- 2 application **frameworks**:
  - Accsoft-Commons-Web (ACW) with Angular
  - PyUI with PyQt and the Acc-Py distribution

What is WRAP?

WRAP is a **w**eb-based **r**apid (GUI) **a**pplication development **p**latform (WRAP)

- **Reduce cost** of developing new GUIs for the control system
    - Low/no-code app development
    - Abstract control system complexities (NXCALS, InCA, JAPC, CMW, etc.)
- Guarantee **low maintenance** cost for created GUIs (technology evolution hidden by the platform)
- Cater for operational and expert application needs
    - **equipment centric** & **operation centric** applications

Project Scope



EXTERNAL DATA SOURCES

Device discovery via Controls Configuration Service (CCS)
Metadata via Injector Controls Architecture (INCA)

EXTENSIVE CUSTOMIZATION

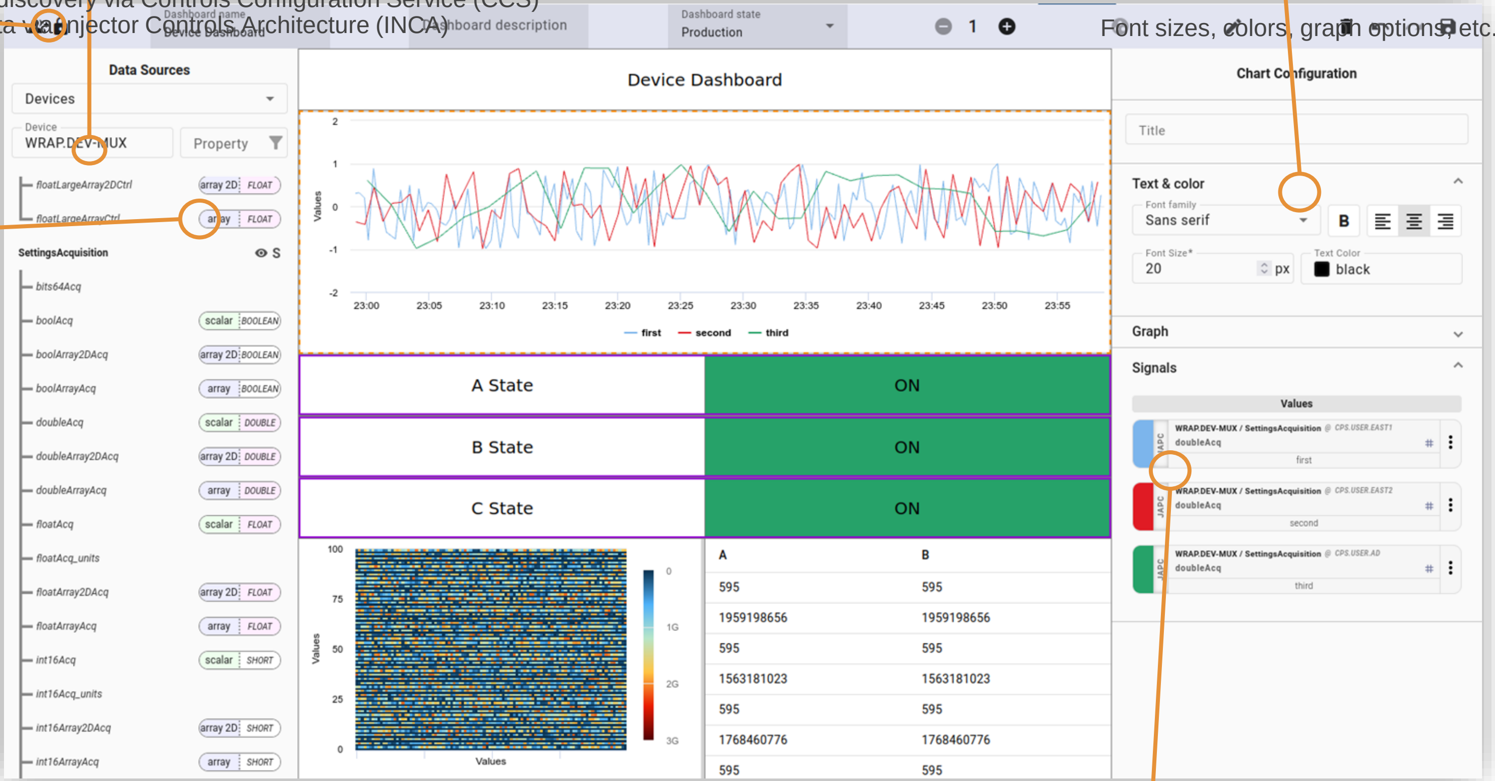
Font sizes, colors, graph options, etc.

PERMISSIONS

Others can view, edit, administrate

METADATA

Consolidated from low level definition / high-level services



Dashboard name: Device Dashboard
Dashboard description
Dashboard state: Production
1

Data Sources

Devices

Device: WRAP.DEV-MUX   Property

floatLargeArray2DCtrl	array 2D	FLOAT
floatLargeArrayCtrl	array	FLOAT
SettingsAcquisition		S
bits64Acq		
boolAcq	scalar	BOOLEAN
boolArray2DAcq	array 2D	BOOLEAN
boolArrayAcq	array	BOOLEAN
doubleAcq	scalar	DOUBLE
doubleArray2DAcq	array 2D	DOUBLE
doubleArrayAcq	array	DOUBLE
floatAcq	scalar	FLOAT
floatAcq_units		
floatArray2DAcq	array 2D	FLOAT
floatArrayAcq	array	FLOAT
int16Acq	scalar	SHORT
int16Acq_units		
int16Array2DAcq	array 2D	SHORT
int16ArrayAcq	array	SHORT

Device Dashboard

A State	ON
B State	ON
C State	ON

A	B
595	595
1959198656	1959198656
595	595
1563181023	1563181023
595	595
1768460776	1768460776
595	595

Chart Configuration

Title

Text & color
Font family: Sans serif
Font Size*: 20 px
Text Color: black

Graph

Signals

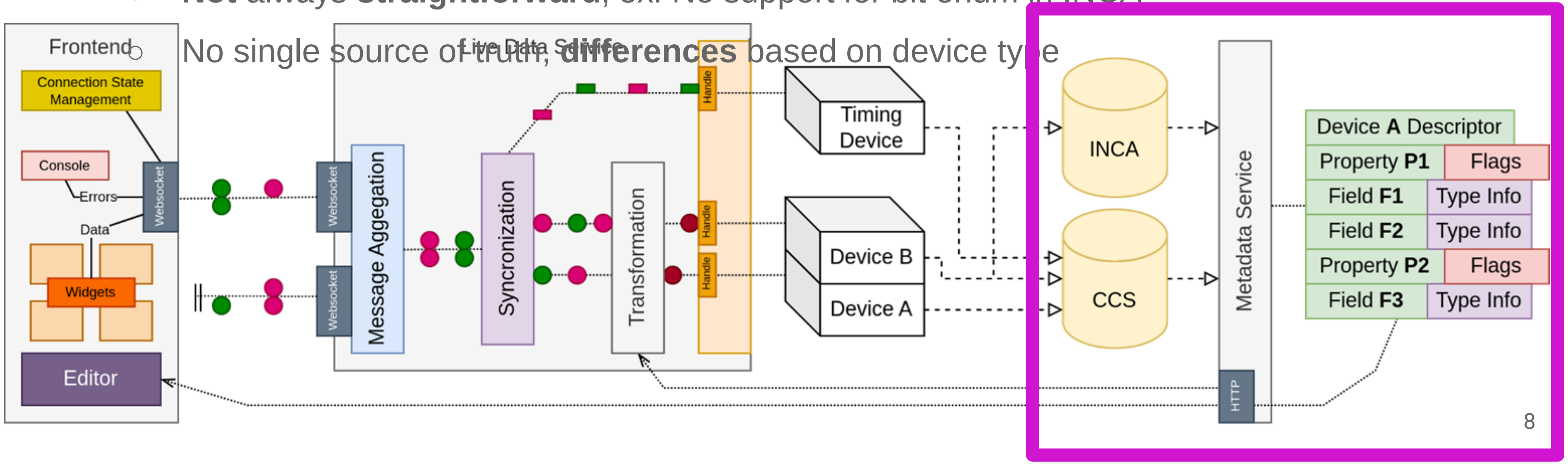
Values

JAPC	WRAP.DEV-MUX / SettingsAcquisition @ CPS.USER.EAST1 doubleAcq	first
JAPC	WRAP.DEV-MUX / SettingsAcquisition @ CPS.USER.EAST2 doubleAcq	second
JAPC	WRAP.DEV-MUX / SettingsAcquisition @ CPS.USER.AD doubleAcq	third

DATA SOURCE ASSIGNMENTS

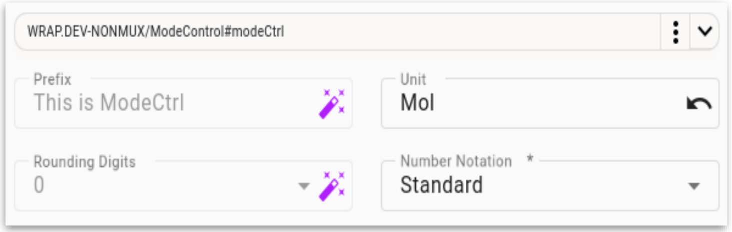
Data Model

- **Metadata** must be reconstructed from multiple services (CCS, INCA, NXCALS)
    - **Not** always **straightforward**, ex. No support for bit enum in INCA
    - No single source of truth, **differences** based on device type



Static Metadata Integration

- Widget is partly **preconfigured** with metadata
- Metadata changes, the **configuration updates** on dashboard load
- Fields that can be derived from metadata, can always also be **explicitly set**



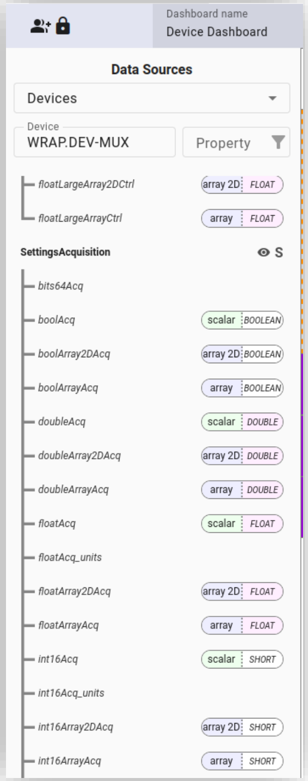
WRAP.DEV-NONMUX/ModeControl#modeCtrl

Prefix: This is ModeCtrl

Unit: Mol

Rounding Digits: 0

Number Notation *: Standard



Dashboard name: Device Dashboard

Data Sources

Devices

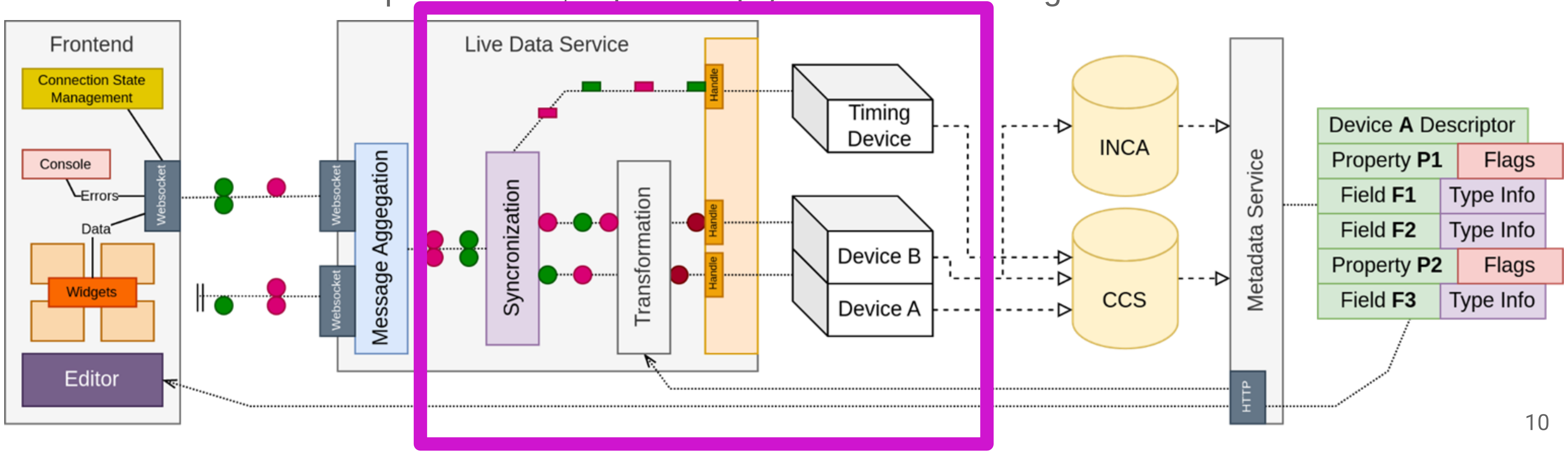
Device: WRAP.DEV-MUX

Property

floatLargeArray2DCtrl	array 2D	FLOAT
floatLargeArrayCtrl	array	FLOAT
SettingsAcquisition		S
bits64Acq		
boolAcq	scalar	BOOLEAN
boolArray2DAcq	array 2D	BOOLEAN
boolArrayAcq	array	BOOLEAN
doubleAcq	scalar	DOUBLE
doubleArray2DAcq	array 2D	DOUBLE
doubleArrayAcq	array	DOUBLE
floatAcq	scalar	FLOAT
floatAcq_units		
floatArray2DAcq	array 2D	FLOAT
floatArrayAcq	array	FLOAT
int16Acq	scalar	SHORT
int16Acq_units		
int16Array2DAcq	array 2D	SHORT
int16ArrayAcq	array	SHORT

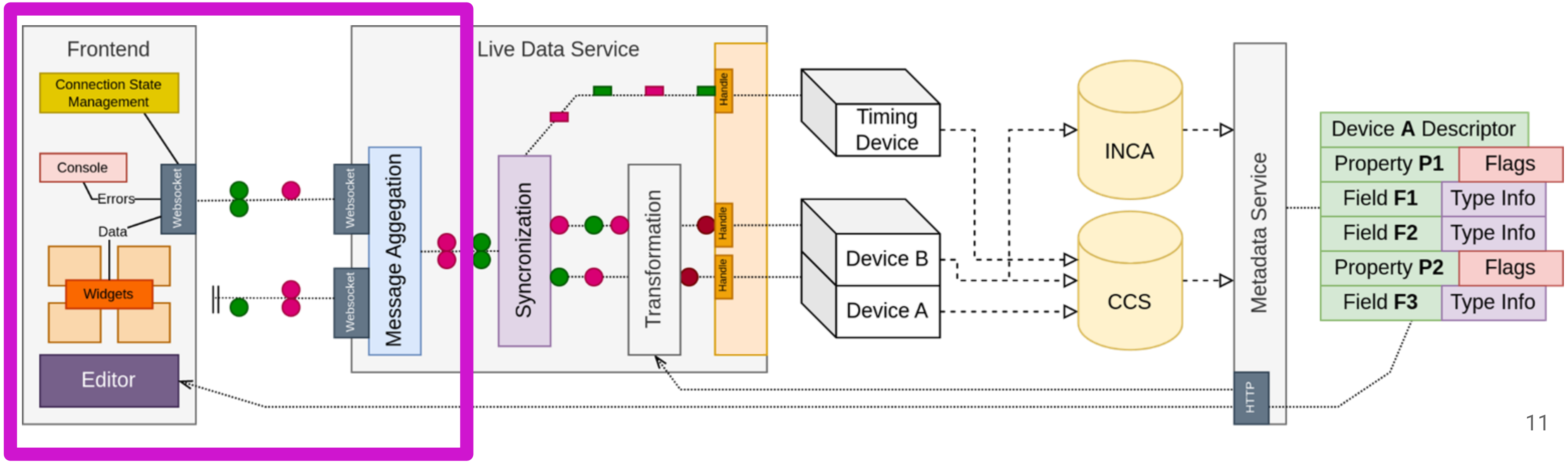
Data Model

- Data has to be **synchronized**, to guarantee coherency of updates
  - Publish once per selector, explicit empty values for messages that are late / lost.



Data Model

- Message **aggregation** through **websocket**, to make the data available



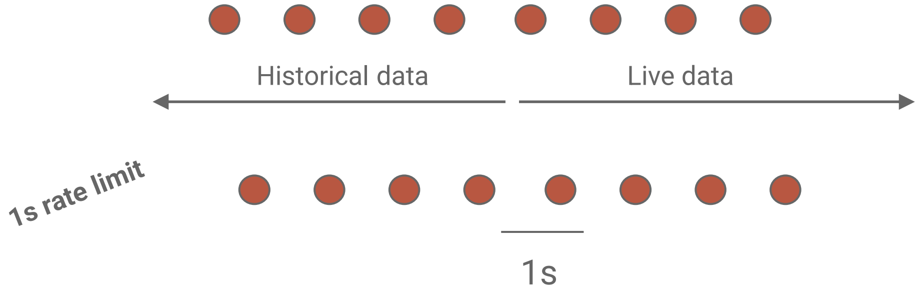
Note: Combining Data

Goal: **Plot** a parameter for the last **N minutes**, update based on **live data**.

Internally → Get last N minutes from NXCALS, subscribe to device property

Problem:

- There may exist **none or multiple variables** tracking this data.
- A filter might be set leading to **information loss**.



Widgets - Visualization

	Data Type	Scalar	1D Array	2D Array
Status Indicator	Any	Yes	Sliced	Sliced
Progress Bar	Numeric	Yes	Sliced	Sliced
Bit Enum	Bit enum	Yes	Sliced	Sliced
Chart	Numeric	Yes (with history)	Sliced	Sliced
Array Chart	Numeric		Yes	Sliced
Heatmap	Numeric		Yes (with history)	Sliced
Waterfall	Numeric			Yes
Data Grid	Any	Yes (multiple, optionally sliced)		Yes (optionally sliced)
SSVG	Dynamic	Yes	No	No
Timing Bar	None			
Label	None			
Date Time	None			

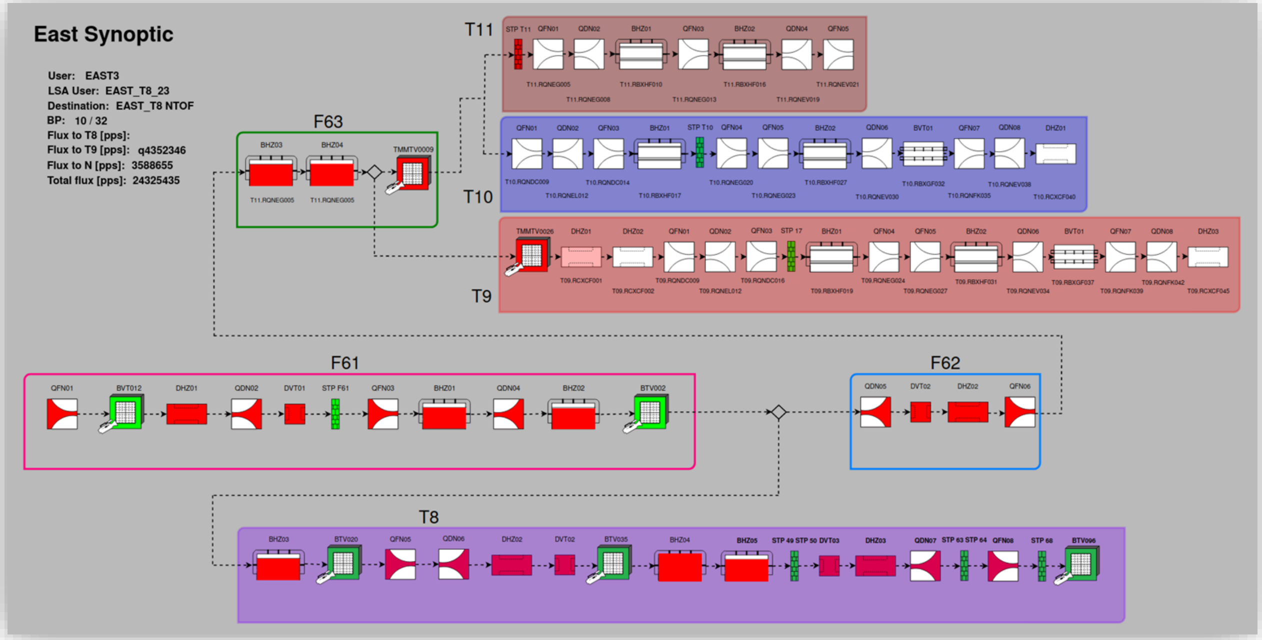
Numeric = Number, Boolean, Enum

Extensibility: Custom visualizations

Stateful Scalable Vector Graphics (SSVG) files can be created by **embedding** metadata on regular SVG files.

A static image file turns into a dynamic **custom widget**.

Data acquisition options can be applied like any native widget.



East Synoptic

User: EAST3
LSA User: EAST_T8_23
Destination: EAST_T8 NTOF
BP: 10 / 32
Flux to T8 [pps]:
Flux to T9 [pps]: q4352346
Flux to N [pps]: 3588655
Total flux [pps]: 24325435

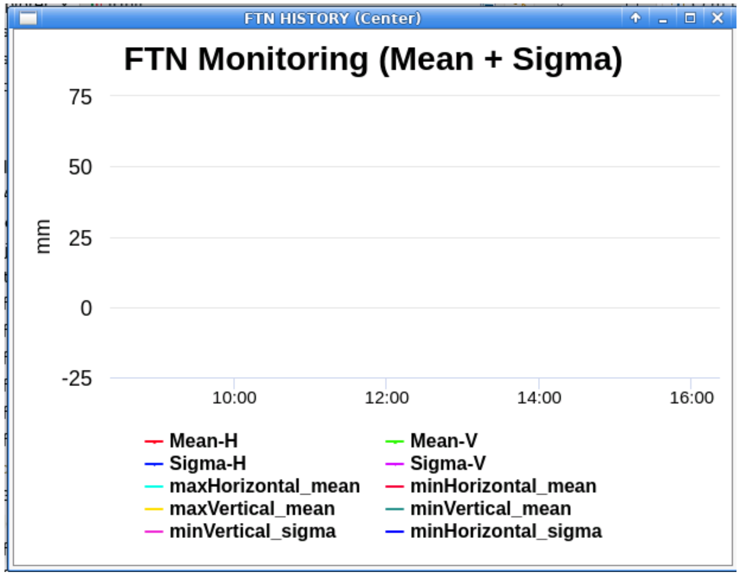
Widgets - Control

	Data Type	Scalar	1D Array	2D Array
Input	Number/String	Yes	Sliced	Sliced
Wheel Switch	Number	Yes	Sliced	Sliced
Combo Box	Enum	Yes	Sliced	Sliced
Toggle Group	Enum	Yes	Sliced	Sliced
Check Box	Boolean	Yes	Yes	Sliced
Bit Enum Set	Bit enum	Yes	Sliced	Sliced
Grid Set (tba)	Any	Yes (multiple, optionally sliced)		Yes (optionally sliced)
Button	Trigger			

CCM Integration at CCC

WRAP is provided as a **standalone application**, packaged with electron.

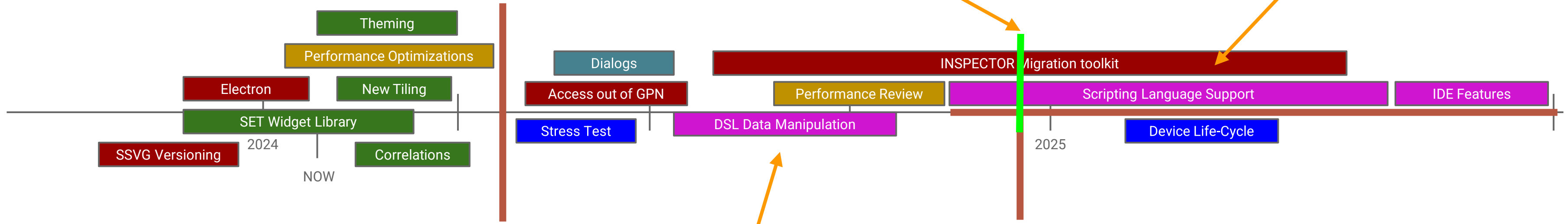
- **No dependency** on client’s web **browser**
- **Native** application **look** & feel
- Accessible through **nfs**: `/acc/java/data/app/wrap/`



Roadmap

All features, max possible performance, no advanced scripting

We target Python



Feature Complete

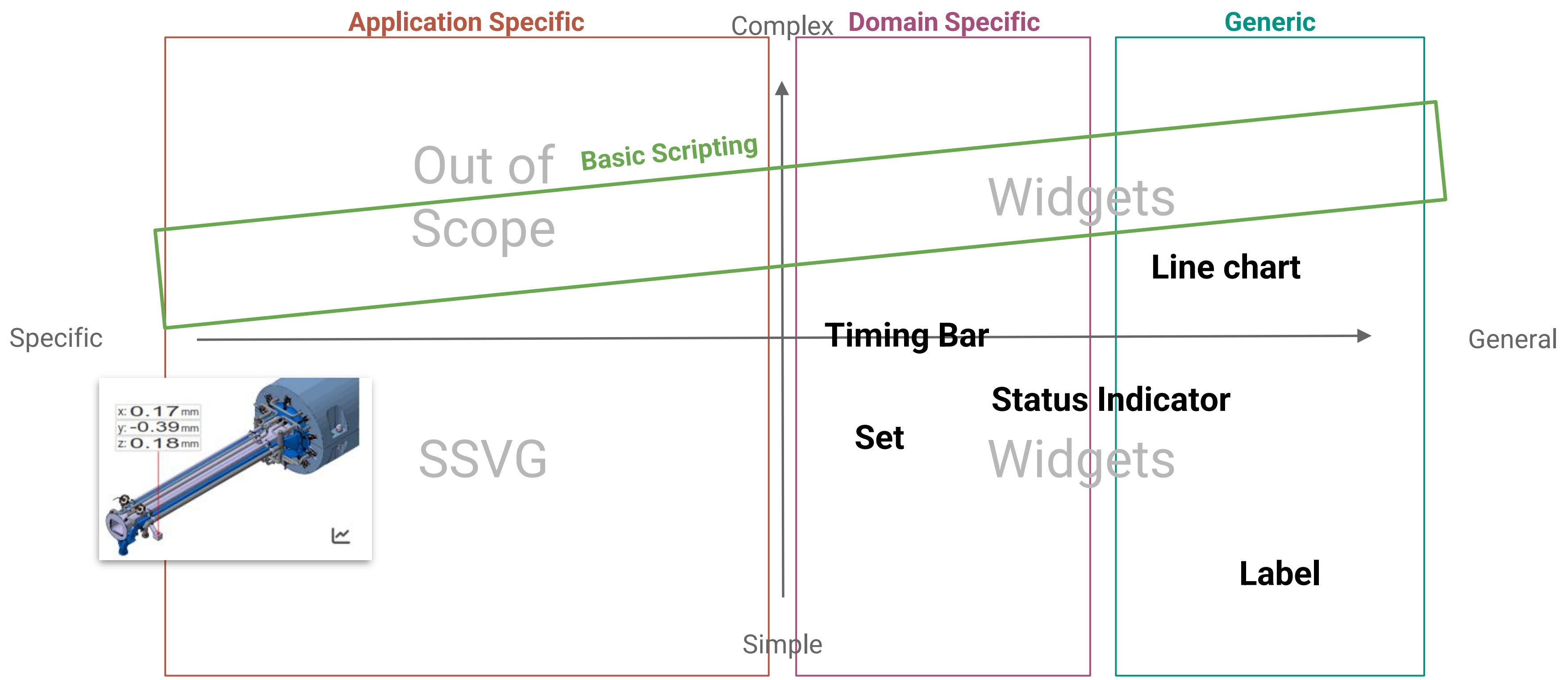
Missing function support, **data processing**, **dialogs**

Focus on ergonomics

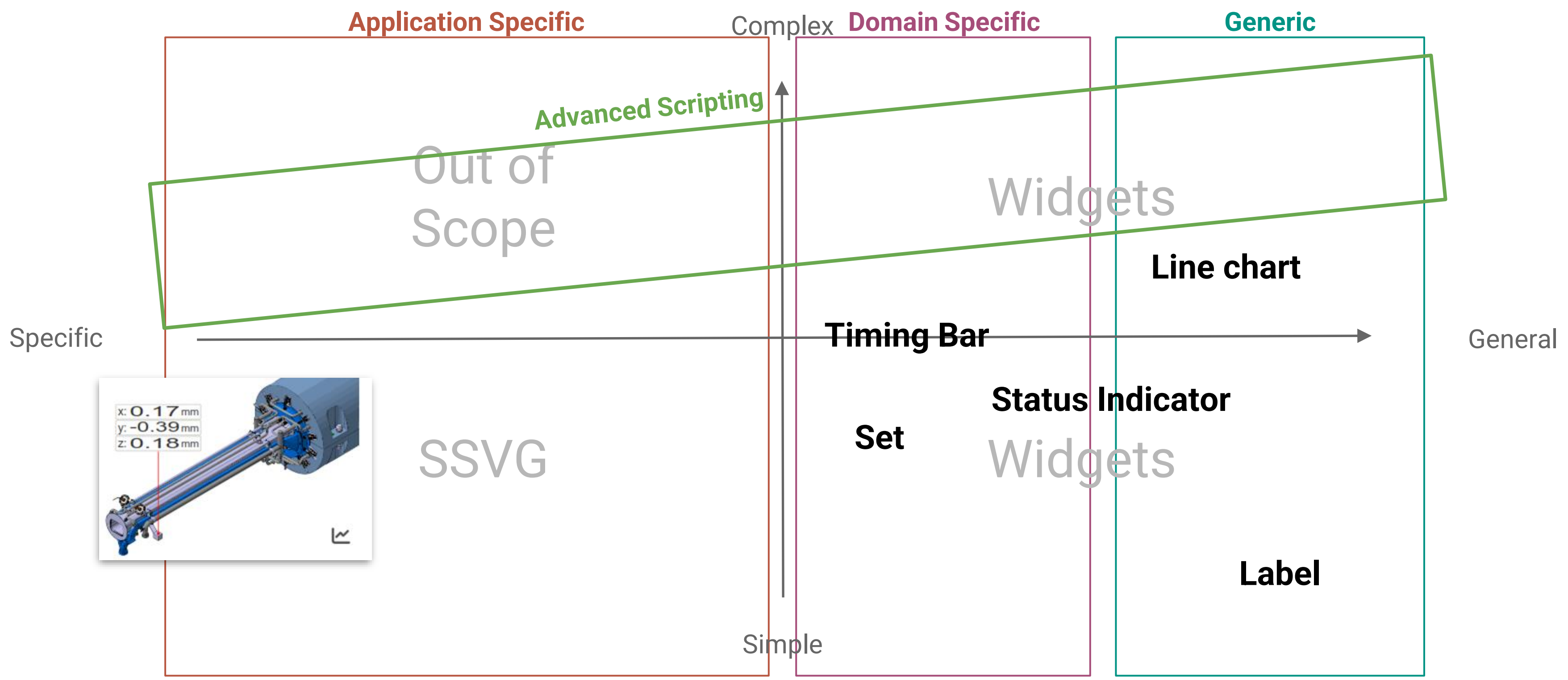
Bulk edit, debug tools

Inspired from Grafana

Project Scope



Project Scope



Closing Notes

Release notes: https://wikis.cern.ch/display/CD/W.R.A.P.-2023.4.0

Electron version info: https://wikis.cern.ch/pages/viewpage.action?pageId=252773401

You can join https://mattermost.web.cern.ch/gui/channels/wrap for:

→ Additional documentation announcements

→ Patch release announcements

→ Questions about general functionality

Please contact us for any problems or ideas at wrap-support@cern.ch