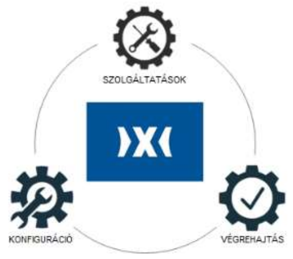
KNORR-BREMSE

OS and language independent network shared memory

- Challanges
- Targets
- Realization
- Usage

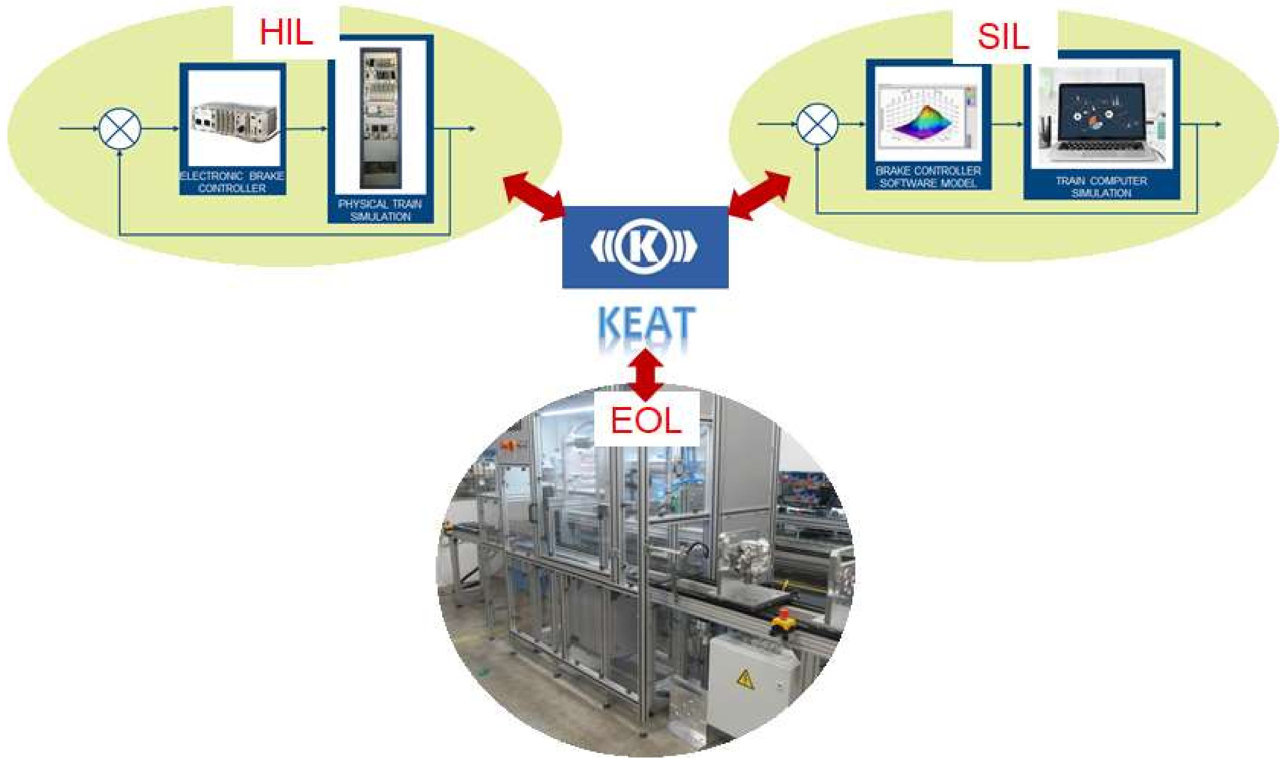
Challanges - 1.

- Railway applications – railway requirements
  - 30 year life cycle for products and even for test environment
  - Documentation according to EN50128
- Integration LabVIEW – C, C++, C# – Python – Oracle technologies in one single platform



Challanges - 1.

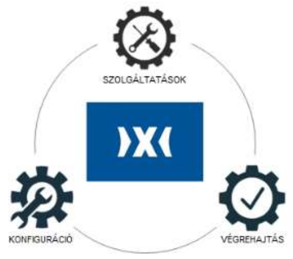
- >20k unique test project vs. 1 test platform
- Number of IO points: 300 – 50,000
- Number of test steps: 500 – 4,000,000


HIL
ELECTRONIC BRAKE CONTROLLER
PHYSICAL TRAIN SIMULATION
SIL
BRAKE CONTROLLER SOFTWARE MODEL
TRAIN COMPUTER SIMULATION
KEAT
EOL

SZOLGÁLTATÁSOK
KONFIGURÁCIÓ
VÉGREHAJTÁS

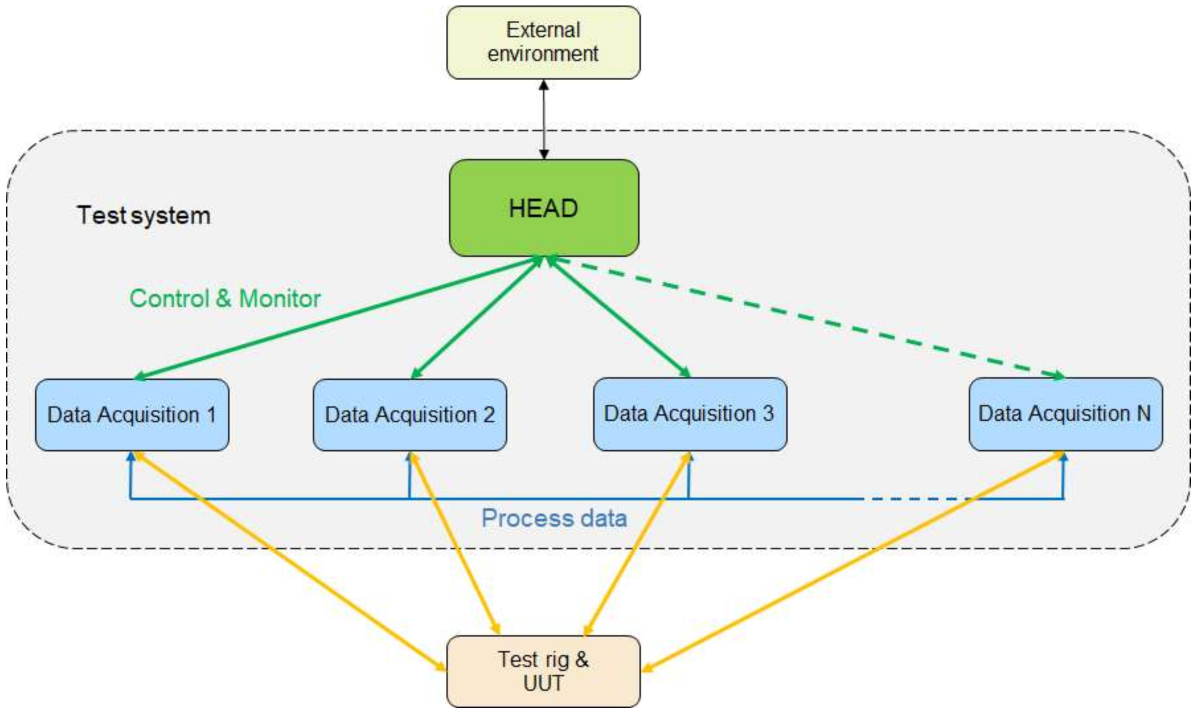
Challanges - 1.

- Any product
- Any test method
- With 30 year support



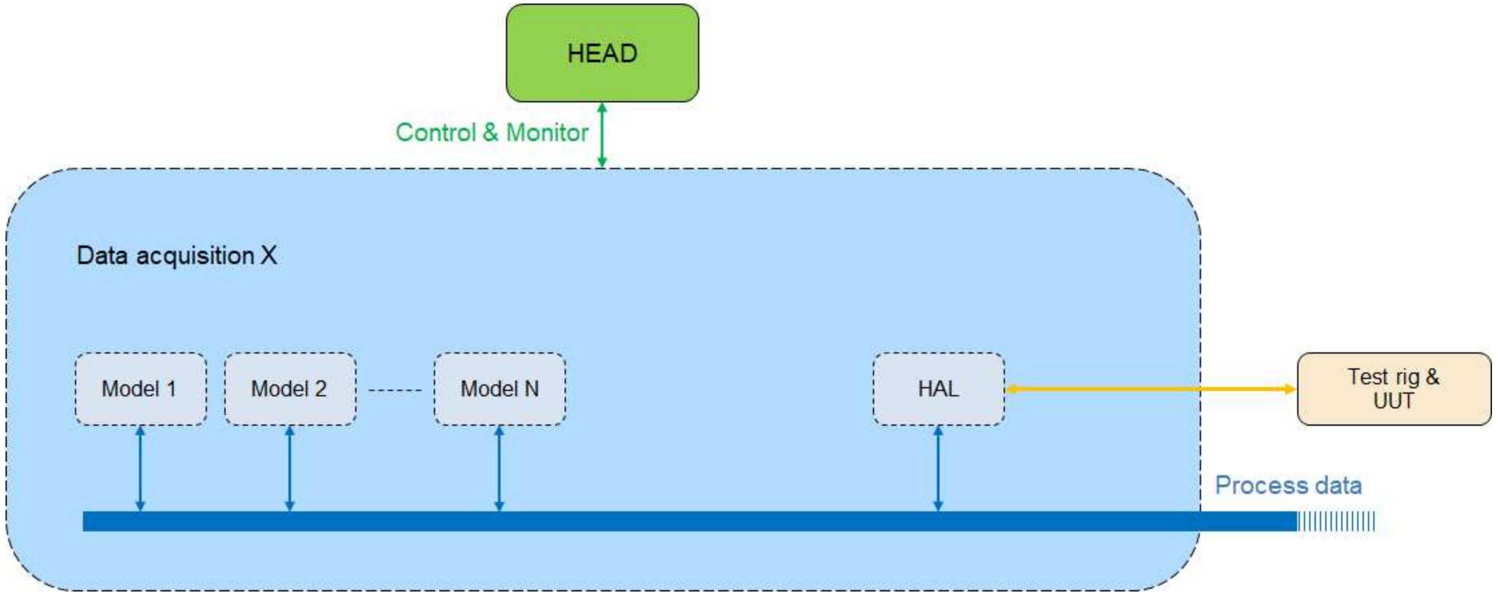
Challanges - 2.

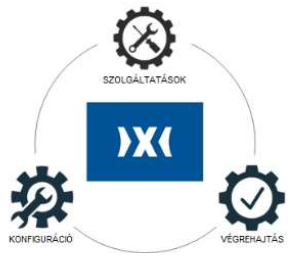
- Microservice architecture
- Any data type



Challanges - 2.

- Microservice architecture
- Any data type





Targets

- Every Consumers shall be able to access the necessary data in different processes running on different machines
- The system doesn’t contains unexplicable artificial elements
  - Artificial ”Master”
  - Artificial cycle times
- Automated priority and conflict handling
- It shall be esteblished when the data is created
- It shall be esteblished that the data is valid or not
- Every known data types shall be handled
- Every data shall be accessable for read / write in parallel
- Data handling shall be independent from OS and language
- The solution shall be portable
- The solution doesn’t requires special hardware
- The solution doesn’t requires license

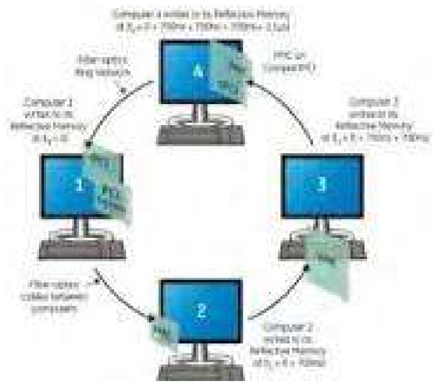
Realization

- Which solution should be selected?
  - Standard solution
    - Reflective memory
    - OPC-UA
    - PROFINET
    - MQTT
    - …
  - Proprietary solution

Realization – case study

- Reflective memory?
  - Fast (some us per node)

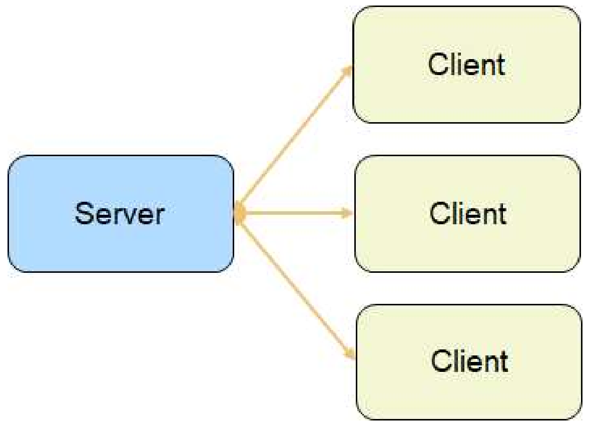
  - Requires special hardware (card + HUB)
  - No event from write / data change. When shall be read the data in which cycle time?
  - Only ring topology
  - Priority and conflict handling on the consumption side
  - Handling of time stamps and validity on the consumption side



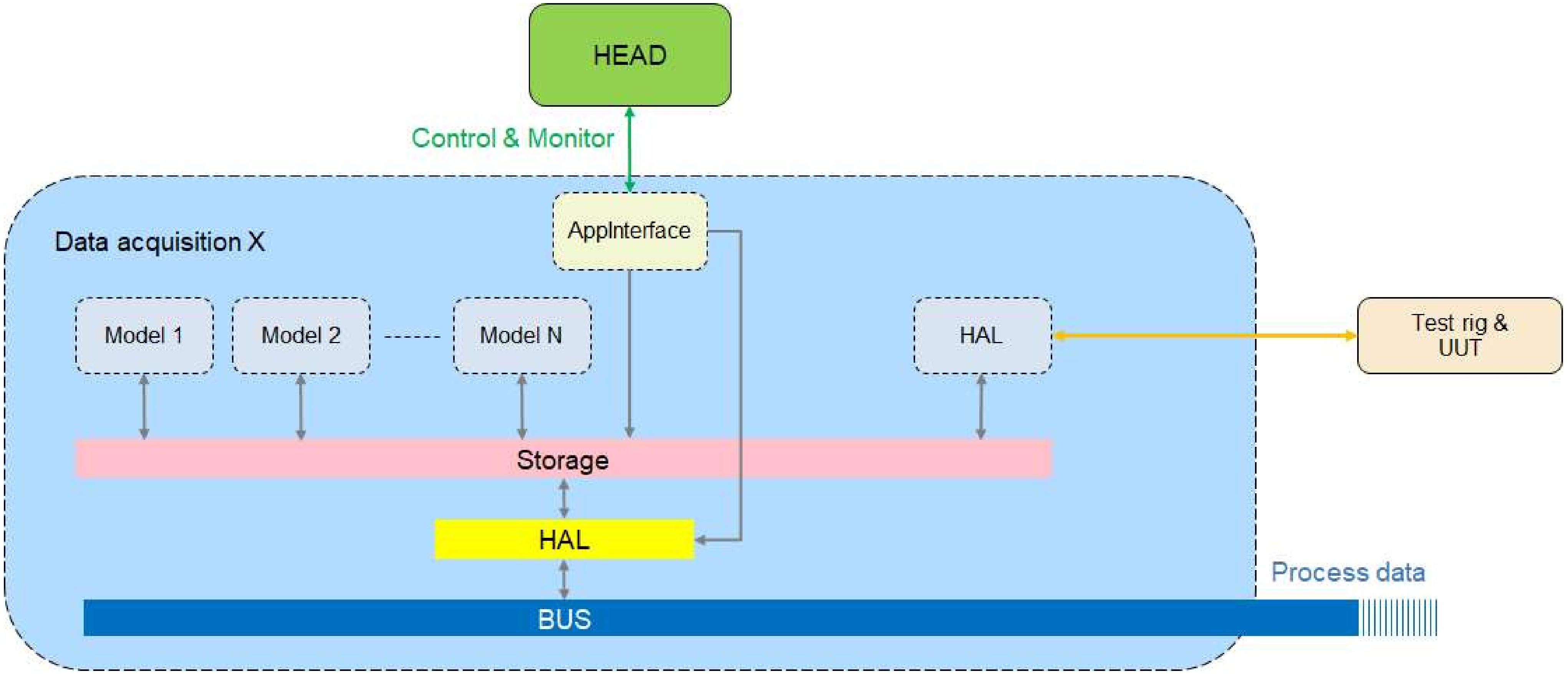
Realization – case study

- OPC-UA?
  - Widespread
  - Existing APIs

  - Who is the Server?
  - Double route of data
  - Priority and conflict handling on the consumption side
  - Handling of validity on the consumption side
  - LabVIEW Toolkit:
    - Requires license
    - Bugs
    - Too slow

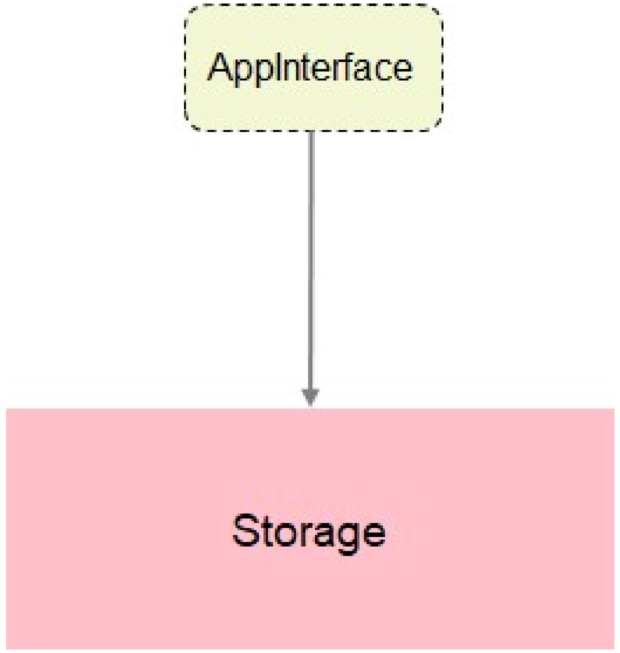


Realization



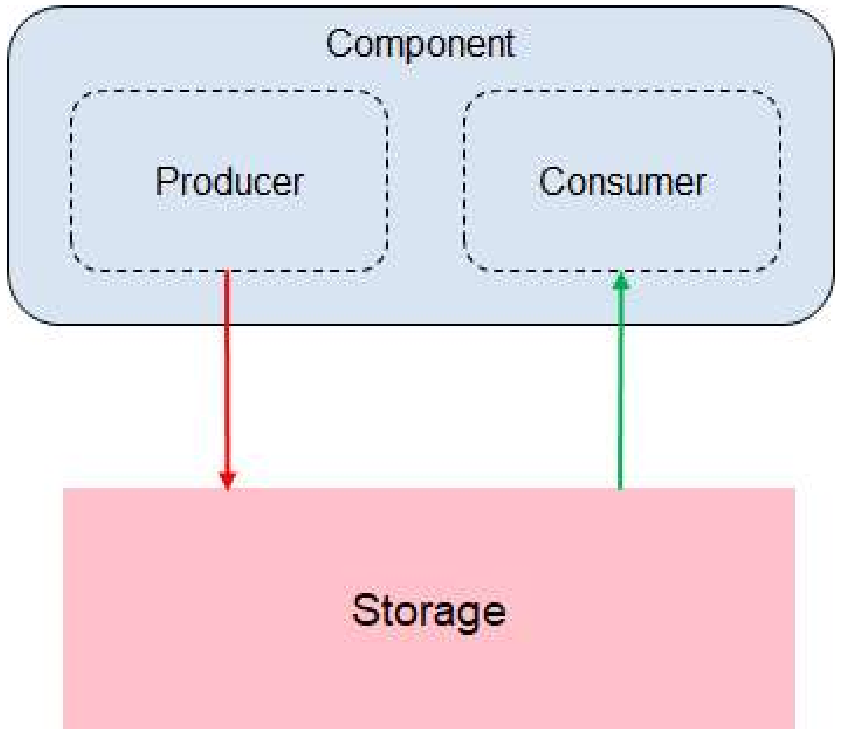
Realization

- Control
  - Creation
    - Configuration of the items
      - Startup value / time stamp / validity
      - Data type
      - Priorities
  - Closing



Realization

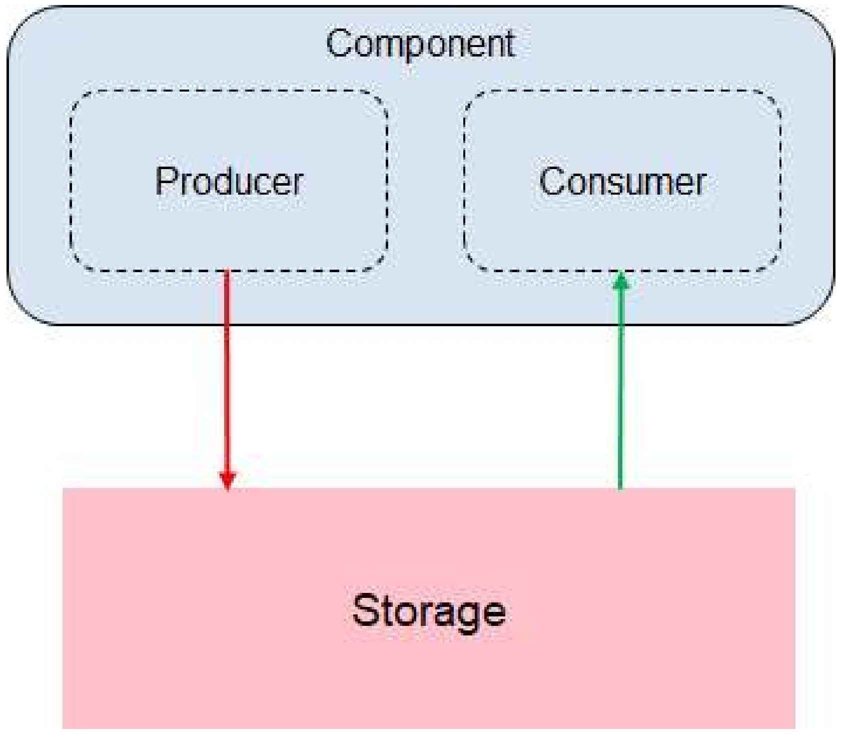
- Data access
  - Write
  - Read



Realization

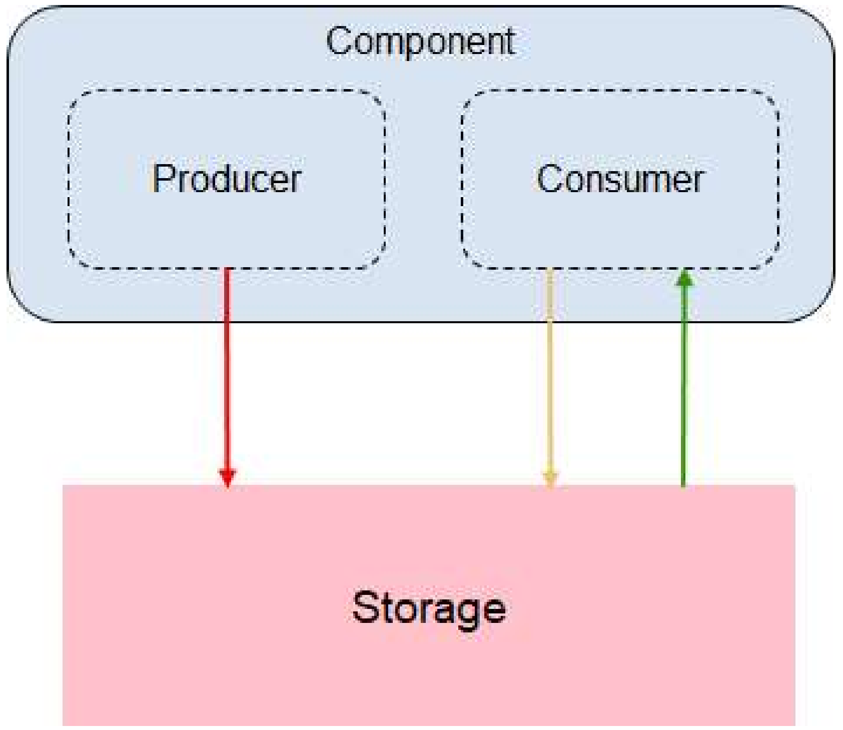
- Data access
  - Write
  - Read

- Cyclic Models
  - Reading inputs
  - Writing outputs
- Test Execution APP
  - Writing Stimulies
  - Reading any data



Realization

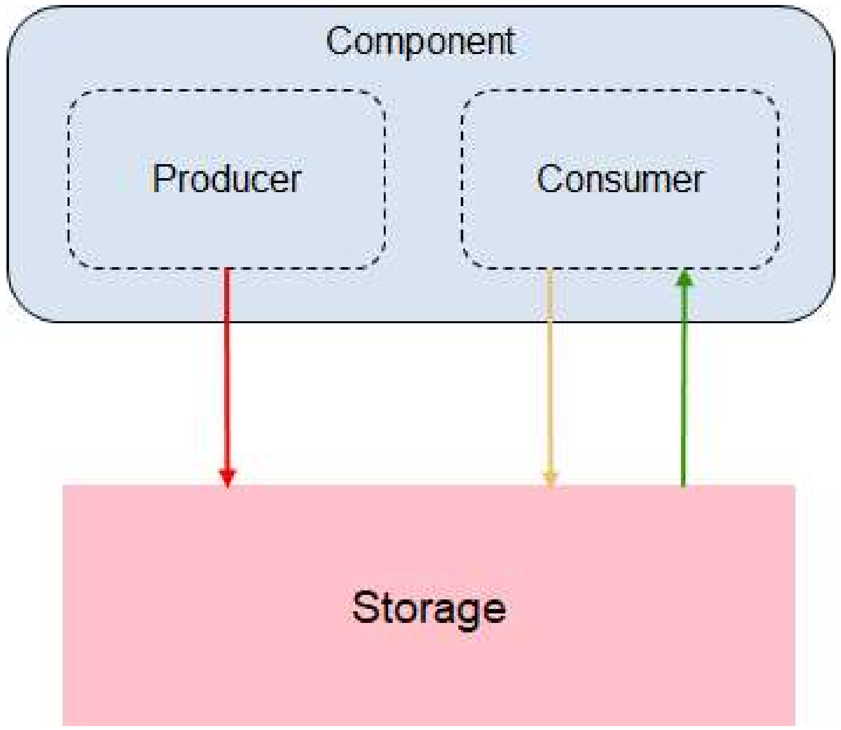
- Data access
  - Writing
  - Registration
  - Event callback from any data change
    - All required data
    - Only changed data



Realization

- Data access
  - Writing
  - **Registration**
  - Event callback from any data change
    - **All required data**
    - Only changed data

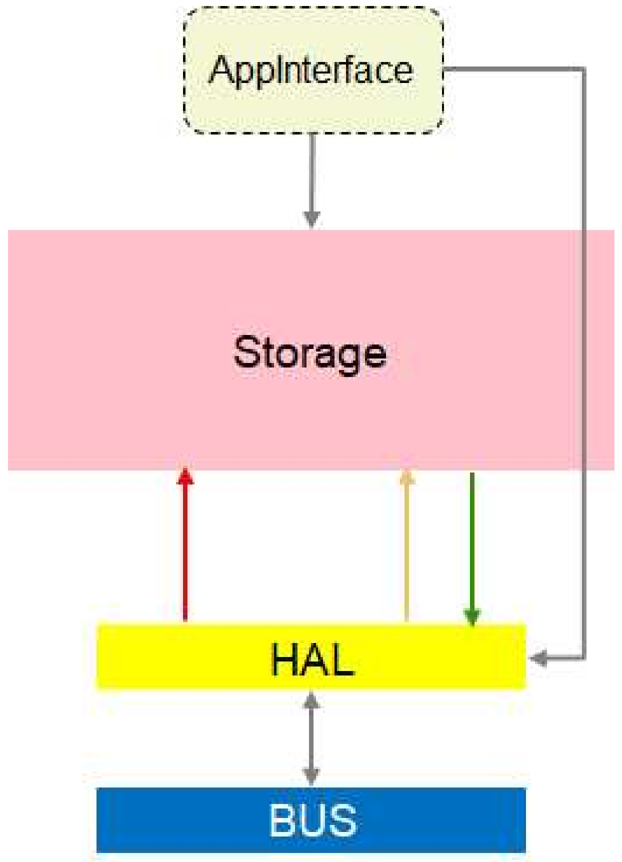
- ***Event-based Models***
  - ***Running the Model only after any input is changed***
- ***Every outputs generated by a data acquisition APP***



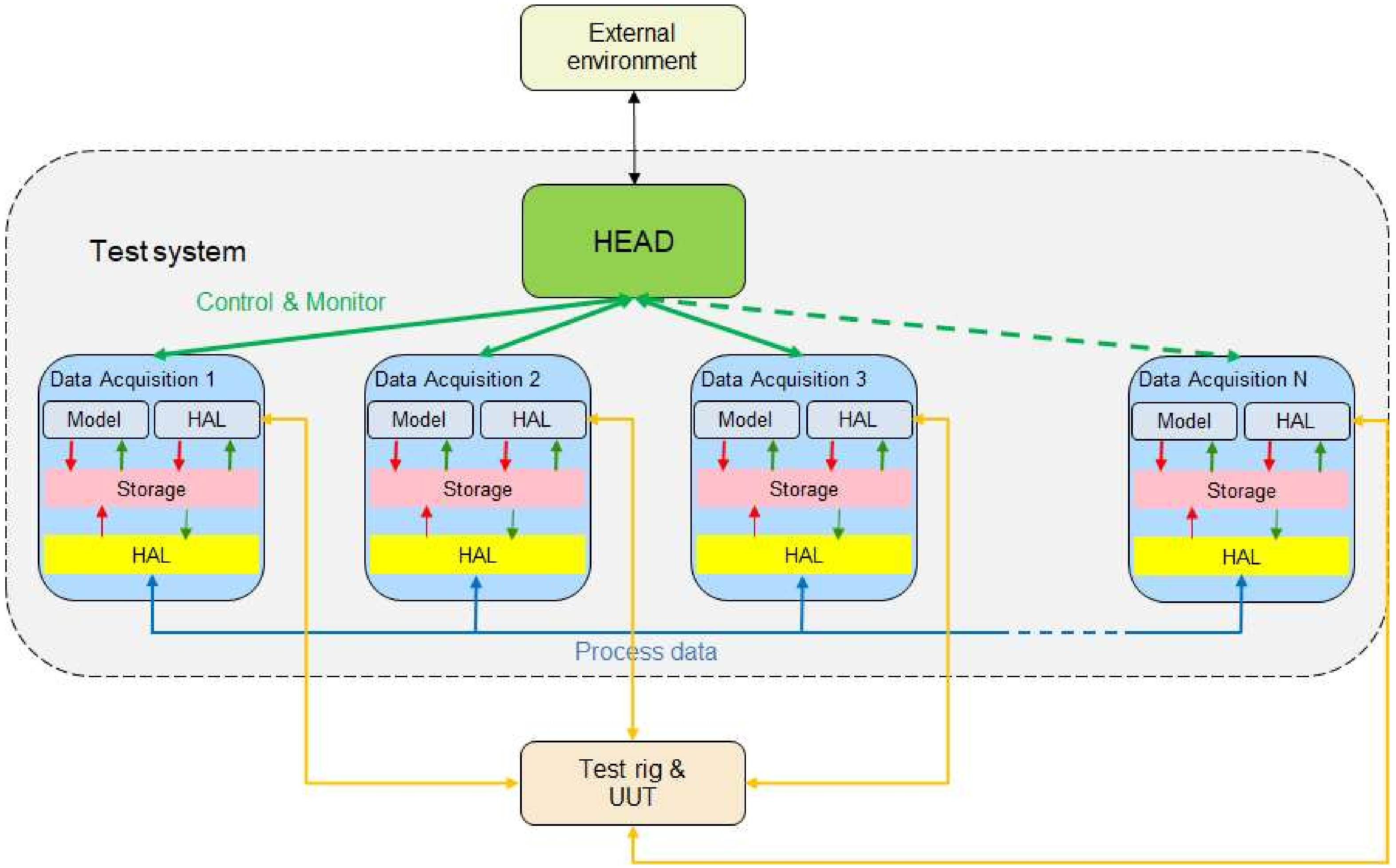
Realization

- Data access
  - Writing
  - **Registration**
  - Event callback from any data change
    - All required data
    - **Only changed data**

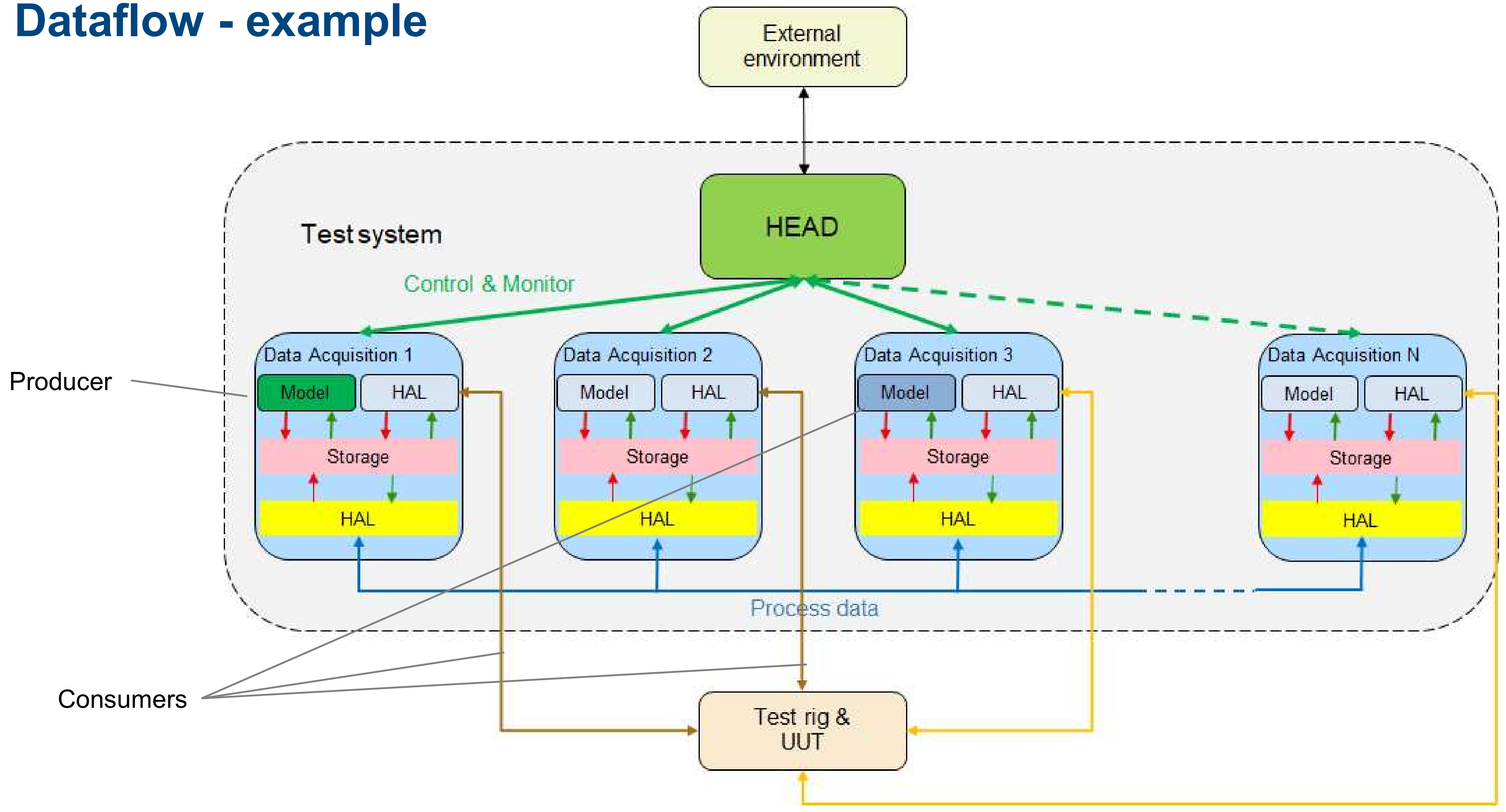
- Event-based Models
  - Running the Model only after any input is changed
- Every outputs generated by a data acquisition APP
- **Communication layer for data synchronization**



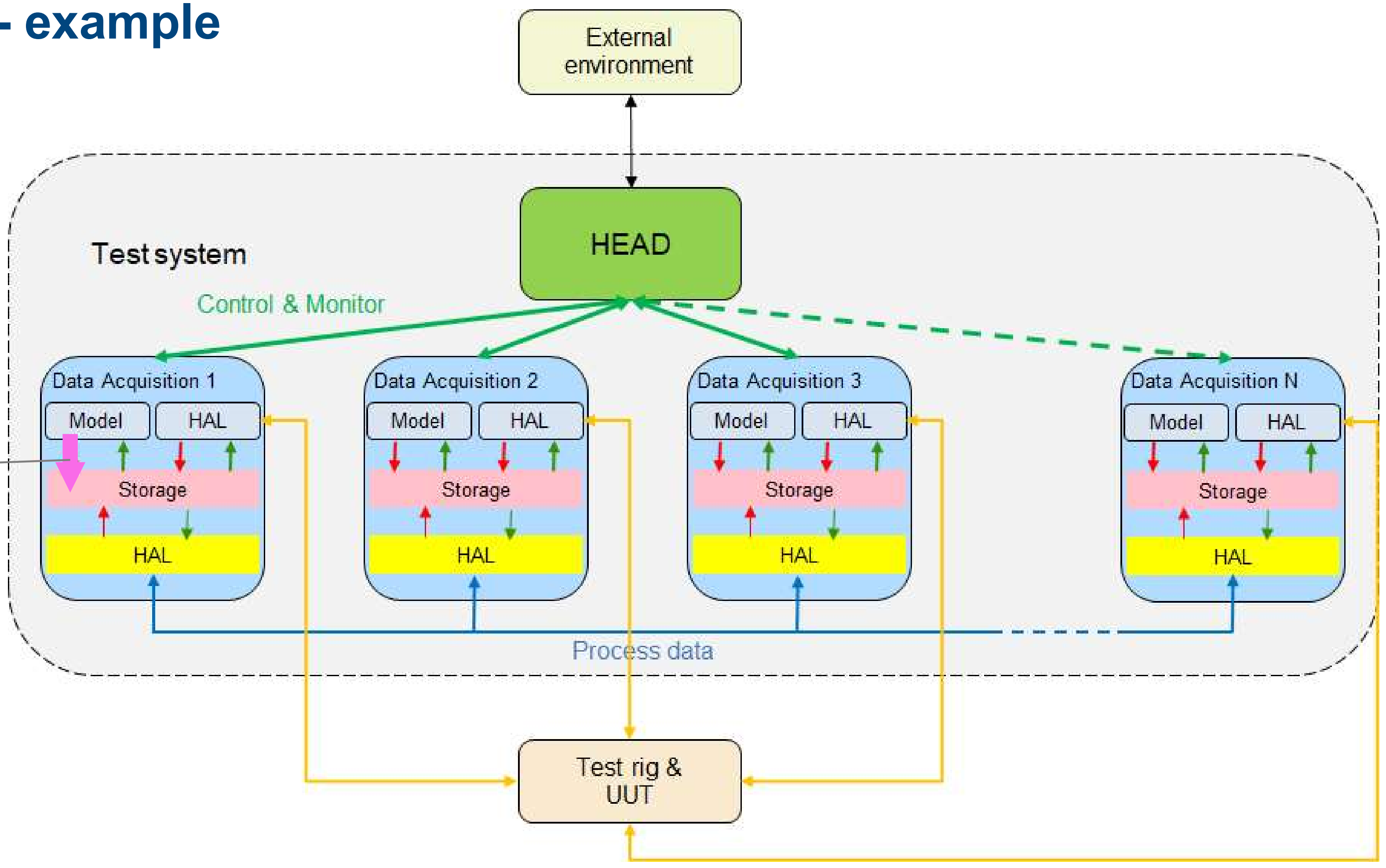
Usage


External environment
HEAD
Test system
Control & Monitor
Data Acquisition 1
Data Acquisition 2
Data Acquisition 3
Data Acquisition N
Model
HAL
Storage
HAL
Process data
Test rig & UUT

Dataflow - example

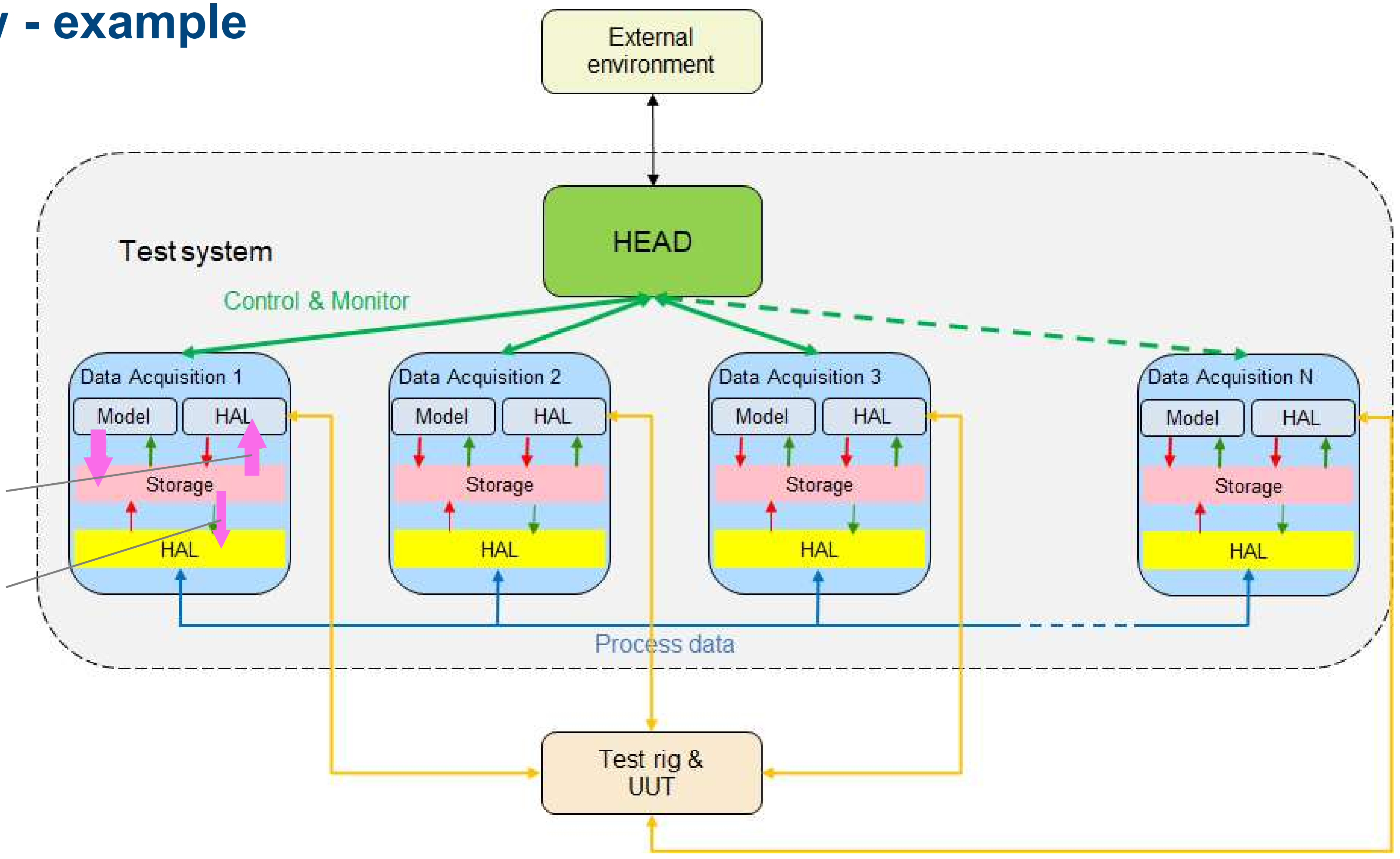


Dataflow - example



New values generated by a Model

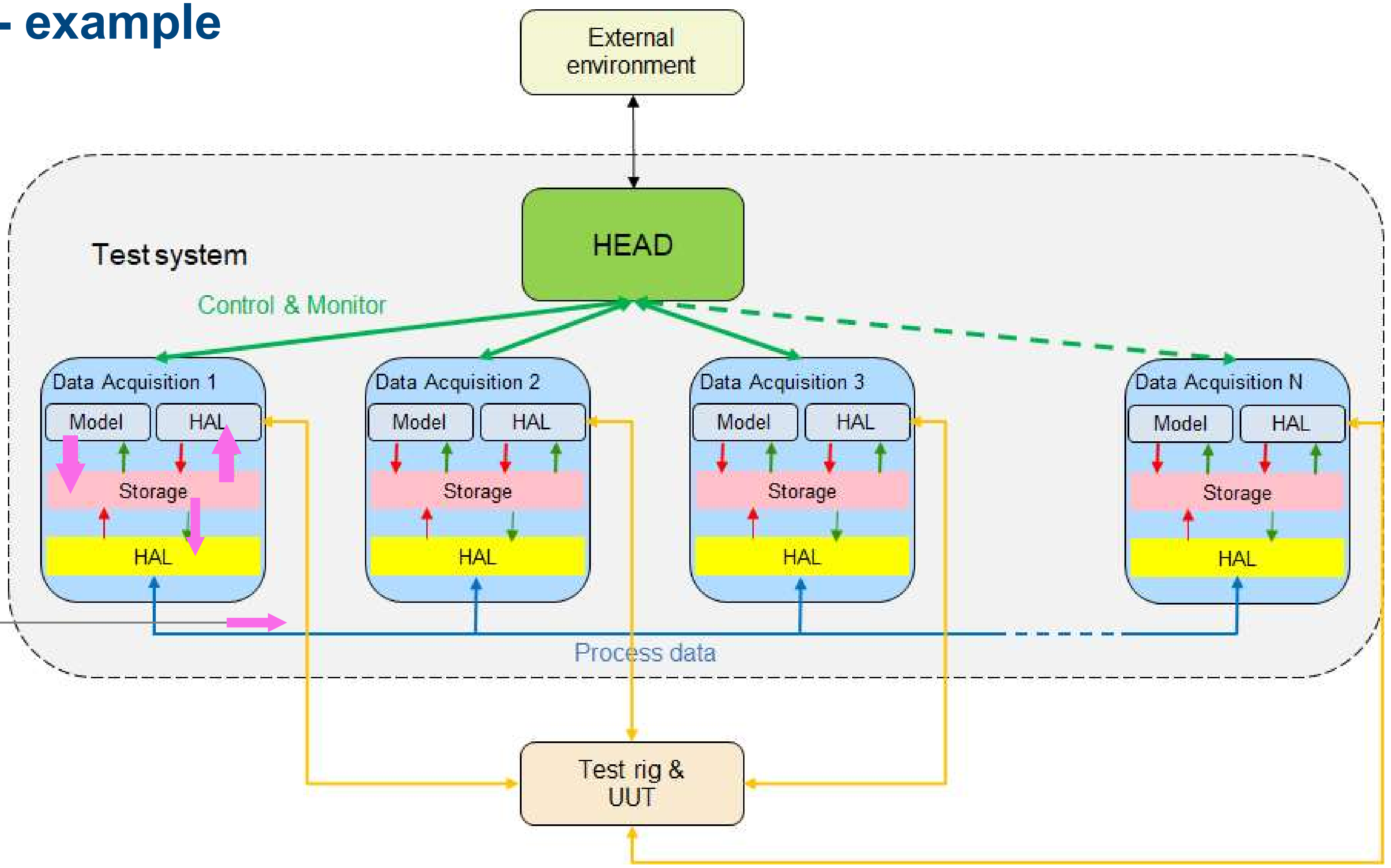
Dataflow - example



Writing outputs

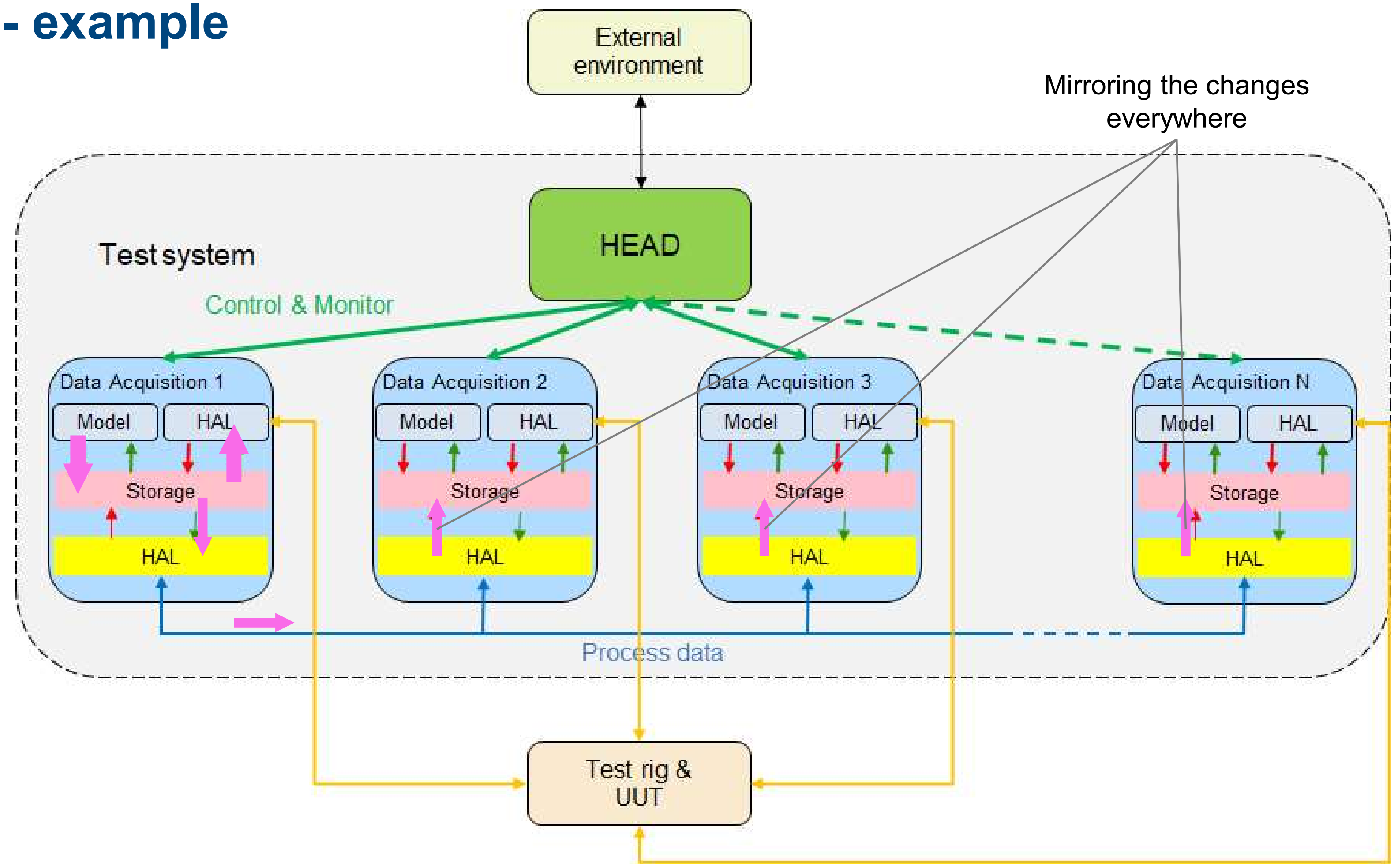
Transfer only changed values

Dataflow - example

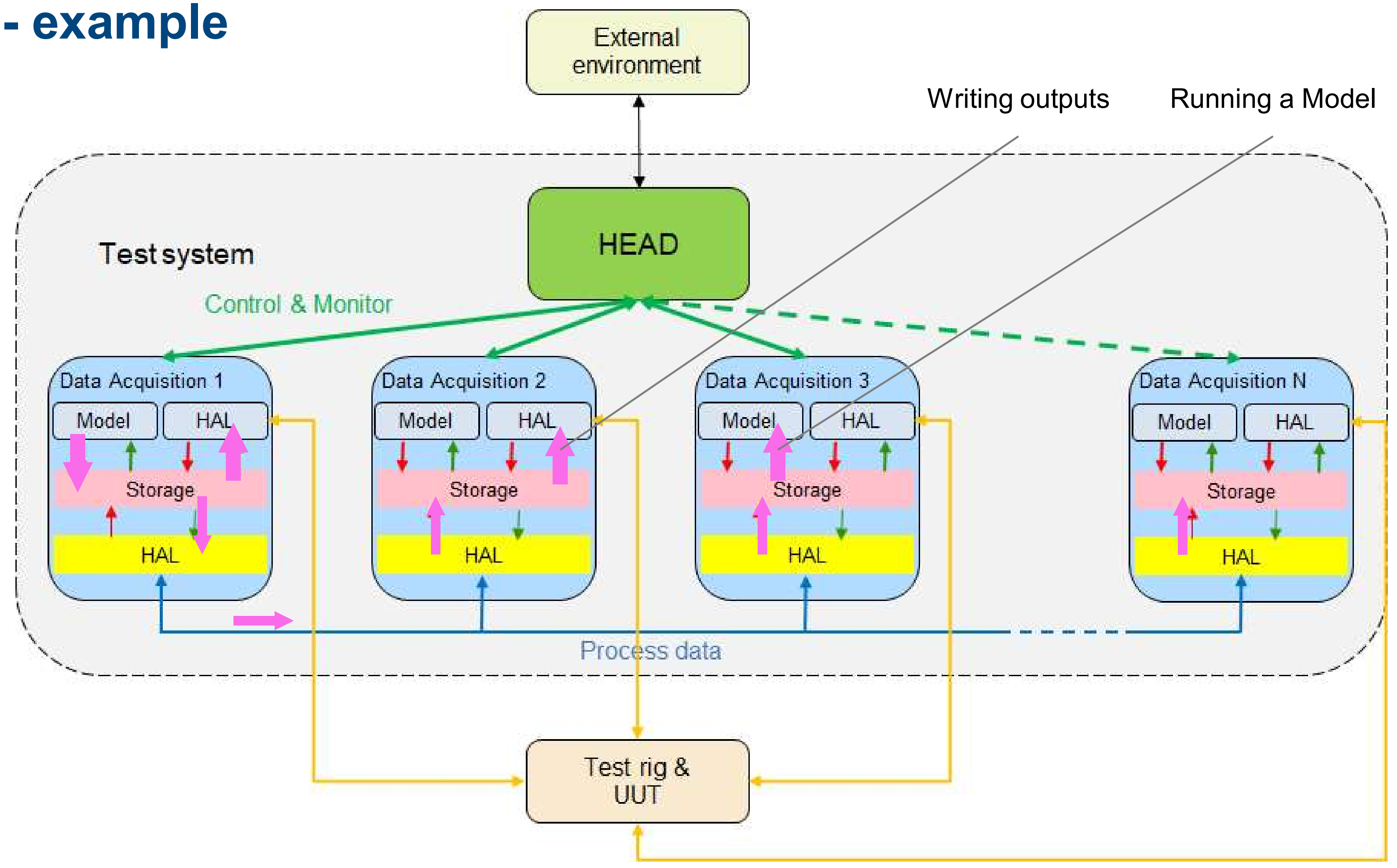


Writing changes to the bus

Dataflow - example

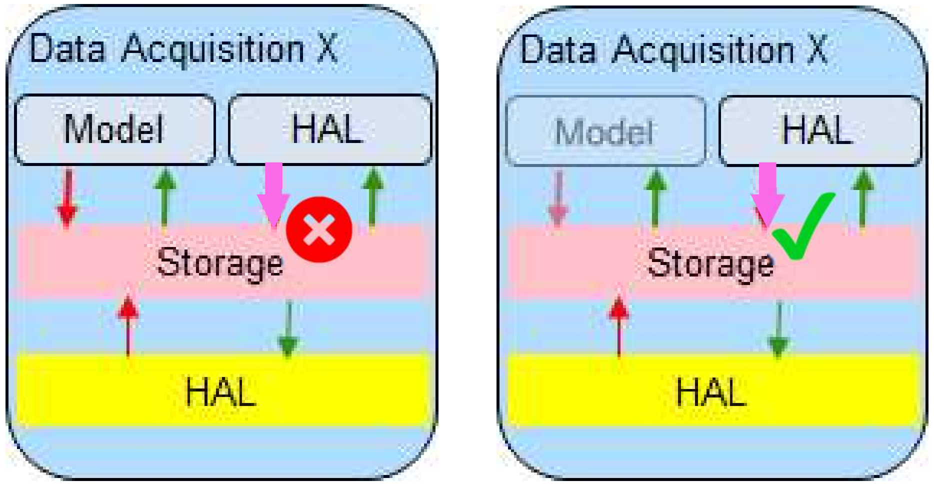


Dataflow - example



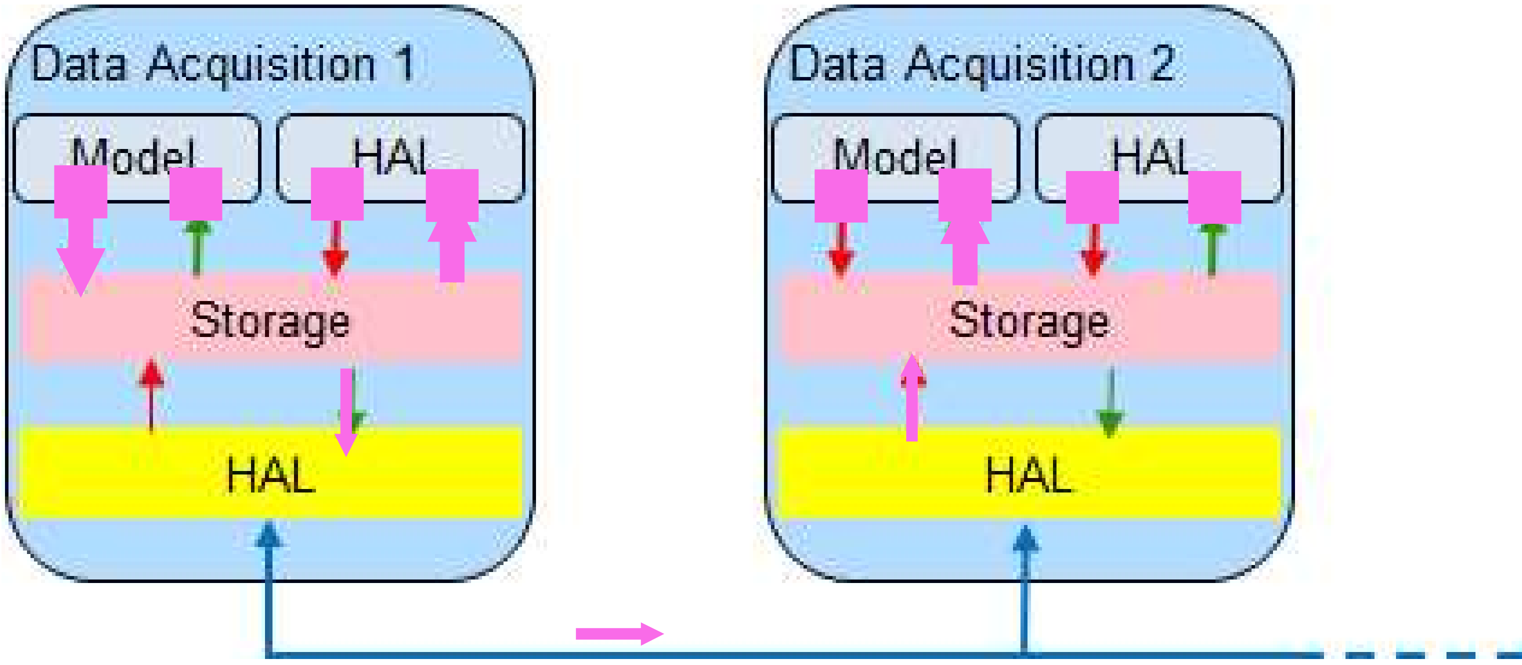
Integrated data protection

- Only one unique producer at the same time for the same data
- Writing access could be changed dinamically



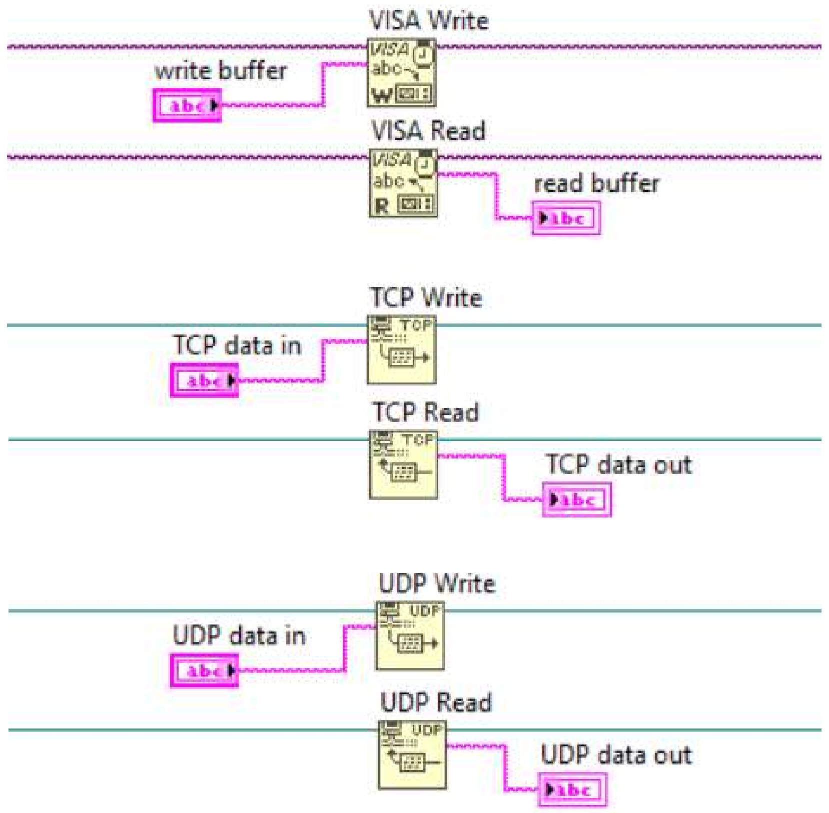
Data integrity and performance

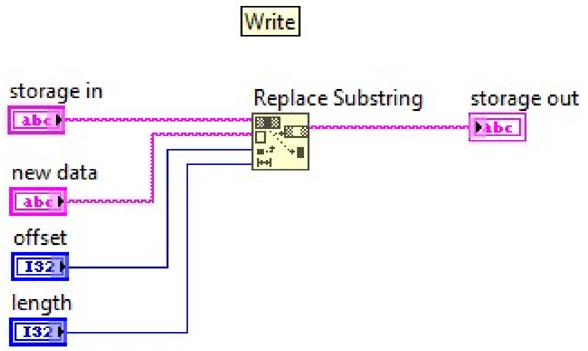
- Data conversion exclusively on the interfaces
- Using the same data type in the whole system without any conversion
- Change-based communication

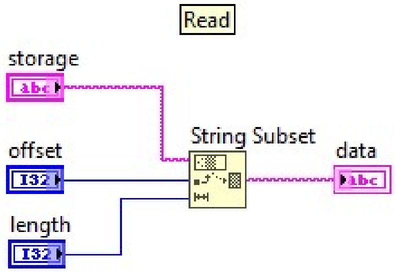


Data integrity and performance

- Data conversion exclusively on the interfaces
- Using the same data type in the whole system without any conversion - **string**
- Change-based communication





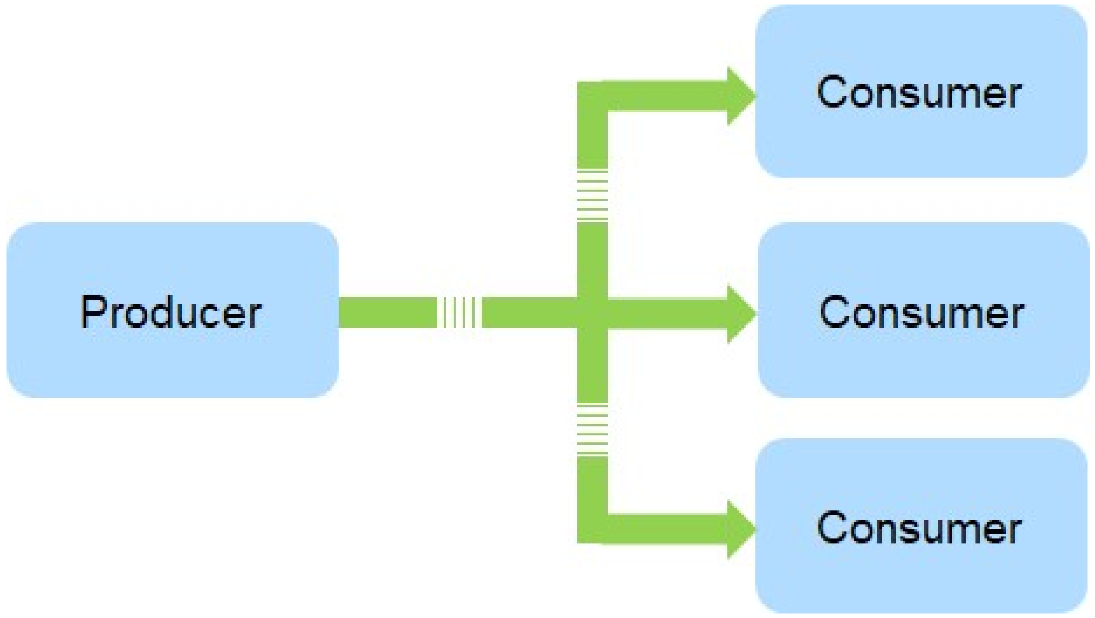


Summary

- Change-based communication
- Automatic event from data change
- No conversion between write and read
- Integrated priority and conflict handling
- Integrated time and validity information for each data
- One single internal data type – any data type could be converted
- Every data are accessible in parallel
- OS and language independent
- Portable
- No special hardware demand
- No special license demand

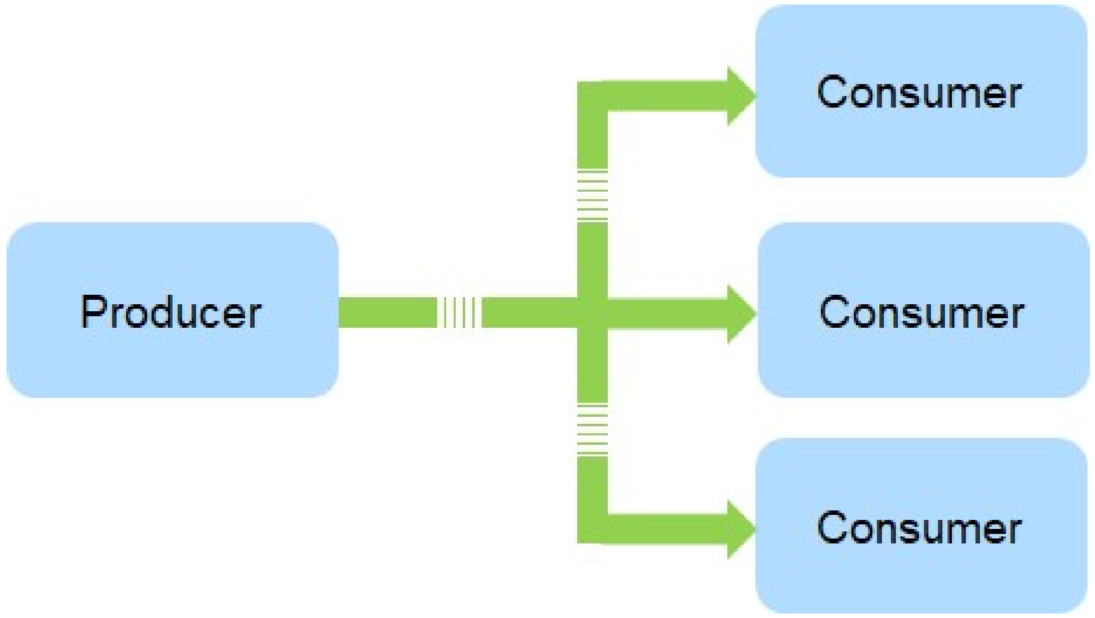
Performance

- Network-dependent
  - 100MB/s
  - 1GB/s
- Protocol-dependent
  - TCP/IP
  - UDP



Performance

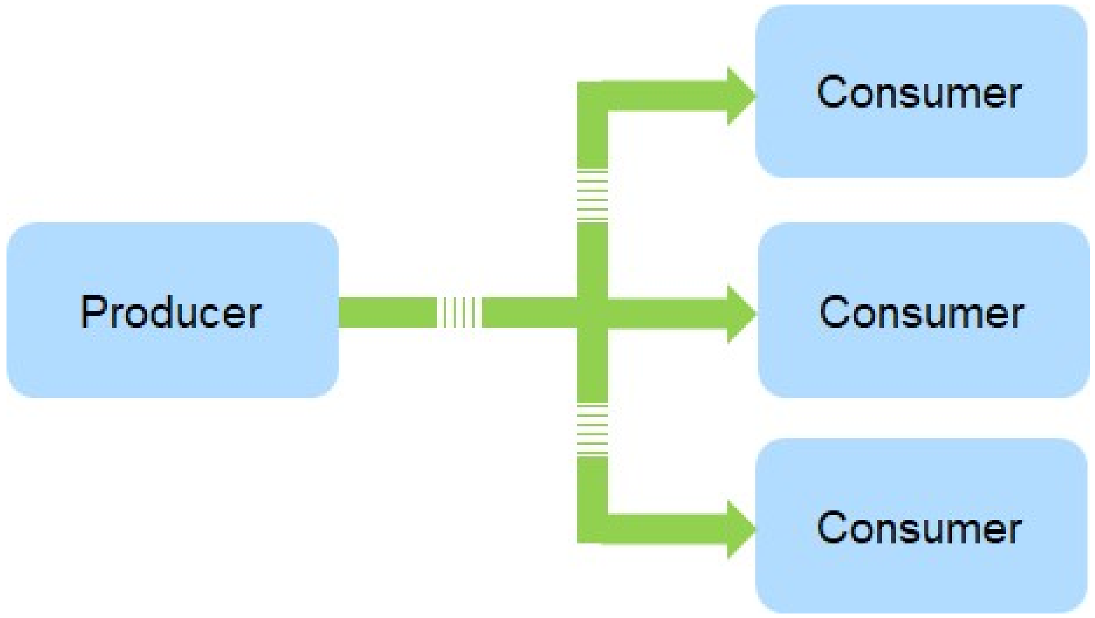
- Network-dependent
  - 100MB/s
  - **1GB/s**
- Protocol-dependent
  - **TCP/IP**
  - UDP



<1ms / 256Byte

Performance

- Network-dependent
  - 100MB/s
  - **1GB/s**
- Protocol-dependent
  - TCP/IP
  - **UDP**



<150us / 256Byte

Question?

Thank you for your attention!