The monitoring of the technical infrastructure at CERN relies on the quality of the definition of numerous and heterogeneous data sources. In 2006, we introduced the MoDEStI procedure for the Technical Infrastructure Monitoring (TIM) system to promote data quality. The first step in the data integration process is the standardisation of the declaration of the various data points, whether these are alarms, equipment statuses or analogue measurement values. Users declare their data points and can follow their requests, monitoring personnel ensure the infrastructure is adapted to the new data, and control room operators check that the data points are defined in a consistent and intelligible way. Furthermore, rigorous validations are carried out on input data to ensure correctness as well as optimal integration with other computer systems at CERN (maintenance management, geographical viewing tools etc.). We are now redesigning the MoDEStI procedure in order to provide an intuitive and streamlined Web based tool for managing data definition, as well as reducing the time between data point integration requests and implementation. Additionally, we are introducing a Class-Device-Property data definition model, a standard in the CERN accelerator sector, for a more flexible use of the TIM data points.

<table>
<thead>
<tr>
<th>If data is in error the requestor must start again</th>
<th>Several days can elapse before the datapoints are configured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separate application for registering request.</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>All requests pass by TI operators, who then determine routing. Validation errors halt the process. Request must be restated.</td>
<td></td>
</tr>
<tr>
<td>Data point requestor</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>TI CCC operators</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>Cabling team</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>External Databases</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>Validation</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>Store Data Points</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>TIM reference DB</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>TIM reference DB</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>TIM reference DB</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>CSAM Safety</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>TIM Server</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>LASER Alarms</td>
<td>Configuration launched manually.</td>
</tr>
<tr>
<td>Test points</td>
<td>Configuration launched manually.</td>
</tr>
</tbody>
</table>

Request created directly in web interface. Form generated from Schema. Checks performed against reference DB. All errors immediately reported to user. Workflow only routes alarms to TI ops. Workflow only routes requests with cabled points for addressing. Full validations have been done at the input stage. TIM reference DB is updated. Requests without alarms or cabled points configured immediately after validation. Same interface used for testing configured points.

A few seconds after entering the data (if cabling and alarms are not involved) testing can commence.

### MoDEStI 2 Architecture

- **frontend**
  - Technologies: AngularJS, Bootstrap
  - REST API

- **backend**
  - Technologies: Spring 4 [Boot, Data, REST, MVC], Activiti (BPMN engine)

- **schema**
  - constraints
  - dependencies

- **workflow**
  - validation
  - configuration

- **TIM plugin**
  - CSAM plugin

- **TIM db**
  - CSAM db

- **MoDEStI db**

- Intermediate rapid datastore (MongoDB)

- Backend provides frontend with schemas and validates against them

- Backend orchestrates the workflows, invoking plugin hooks and/or notifying humans as necessary

- Multiple plugins are supported

- Frontend and backend communicate via REST API

- Frontend displays domain-specific creation and search controls based on schemas

- User creates a request and begins to enter data points into the web interface

- User clicks "validate" button, schema constraints and deep database validations are carried out

- If the request is invalid, the user is immediately notified of the errors

- If the TI operators reject the request, they must supply a rejection reason comment

- Addressing team receives an email that the request is ready for addressing

- The request is automatically configured onto the database

- The user and the TI operators must collaborate to make sure the new points are flowing correctly through the system

- If the test fails, the request must be modified. Otherwise, the request is closed.

### MoDEStI 2 Workflow

1. **prepare**
2. **validate**
   - **valid?**
     - **no**
     - **yes**
6. **approve**
   - **approved?**
     - **no**
     - **yes**
7. **address**
8. **configure**
   - **test**
   - **passed?**
     - **no**
     - **yes**