Industry 4.0

Simulator Documentation

Prof. Dr. Julian Reichwald
julian.reichwald@dhbw-mannheim.de

1 Introduction

To simulate an industrial multistage production process on different levels, an executable jar package is provided. The jar contains all necessary components for the simulation, which are:

1. An underlying message broker based on Apache ActiveMQ.
2. The simulation of an indexed line which incorporates two independent manufacturing stages (milling and drilling). The indexed line is controlled by a simulated Siemens S7-1200 PLC.
3. An OPC server for accessing the sensor / actuator data of the indexed line via the S7-1200 PLC.
4. The (simplified) simulation of manufacturing orders from en ERP system.
5. The (simplified) simulation of a spectral analysis production step (measures if the produced item is ok or not ok).

This document describes the overall process simulated by the software package and gives a detailed description how to use the software.

2 Simulation Process

The overall process can be split into various steps:

1. A work piece equipped with NFC / RFID technology is placed on a production line. The NFC tag is read by the production line and a request for manufacturing order details is sent to an ERP system. The ERP system answers by sending an XML message via the message broker.
2. The work piece now moves through the indexed line and the production stages. Values provided by the PLC:
   - L_1 – L_5 are photo sensors / light barriers which indicate the actual position of the work piece on the line. L_1 and L_2 are at the beginning, L_3 is placed at the milling stage, L_4 is placed on the drilling stage. L_5 is the last sensor at the end of the production line.
   - *MILLING* indicates the status of the milling station
   - *DRILLING* indicates the status of the drilling station
   - *MILLING_SPEED* indicates the actual rotation speed of the milling tool
   - *DRILLING_SPEED* indicates the actual rotation speed if the drilling tool
   - *MILLING_HEAT* indicates the actual temperature of the milling tool
   - *DRILLING_HEAT* indicates the actual temperature of the drilling tool

3. Finally, the work piece checked by a spectral analysis. Since the spectral analysis equipment is not controlled by a PLC and is not able to be integrated in complex IT infrastructures, the only output available is a simple file written in a specific directory. The file itself contains JSON data.

3 Simulation Software

3.1 Start of the simulation

After downloading `I4Simulation.zip`, unpack the archive to your harddisk. In a command line terminal window, enter the following command line:
`java -jar DataAggregator.jar <options>`

Options:

-d n Delay of n milliseconds before simulation starts

-o d Output spectral analysis files to directory d

-m If this option is given, the simulation will run in multimode, i.e. a work piece w_n is placed on the indexed line before the production of work piece w_{n-1} is finished.

-q If this option is given, manufacturing data will not only be provided via the OPC interface but also via the underlying message broker architecture.

The system should be up and running immediately.

3.2 ActiveMQ Message Broker

The ActiveMQ message broker binds to all known network interfaces on port number 61616. Thus, you can connect to ActiveMQ via Java JMS and the following code:

```
1 String conStr = "tcp://localhost:61616";
2 ConnectionFactory connectionFactory =
3   new ActiveMQConnectionFactory(conStr);
```

The `ConnectionFactory` object is a JMS implementation. ERP data is published to the topic `m_orders`. If the -q switch is used, machine data is published to the topic `m_opcitems`.

3.3 OPC Server

The OPC UA server also binds to all known network interfaces using a binary TCP protocol. It is accessible using the following URI:

`opc.tcp://localhost:9999/OPCUA/DHBWTestServer`

Note: no unsecure connections are allowed so use security mode *signed* and security policy *Basic128* or *Basic256*

4 Helpful Tools and Libraries

- Apache Maven
- Git / Github
- ActiveMQ Client (Java Library implementing the JMS API for ActiveMQ)
- Apache Camel
- Stateless4J (Java library for implementing finite state machines)
- EsperTech (Java library for complex event processing)
- D3.js (Javascript library for scientific data visualization)