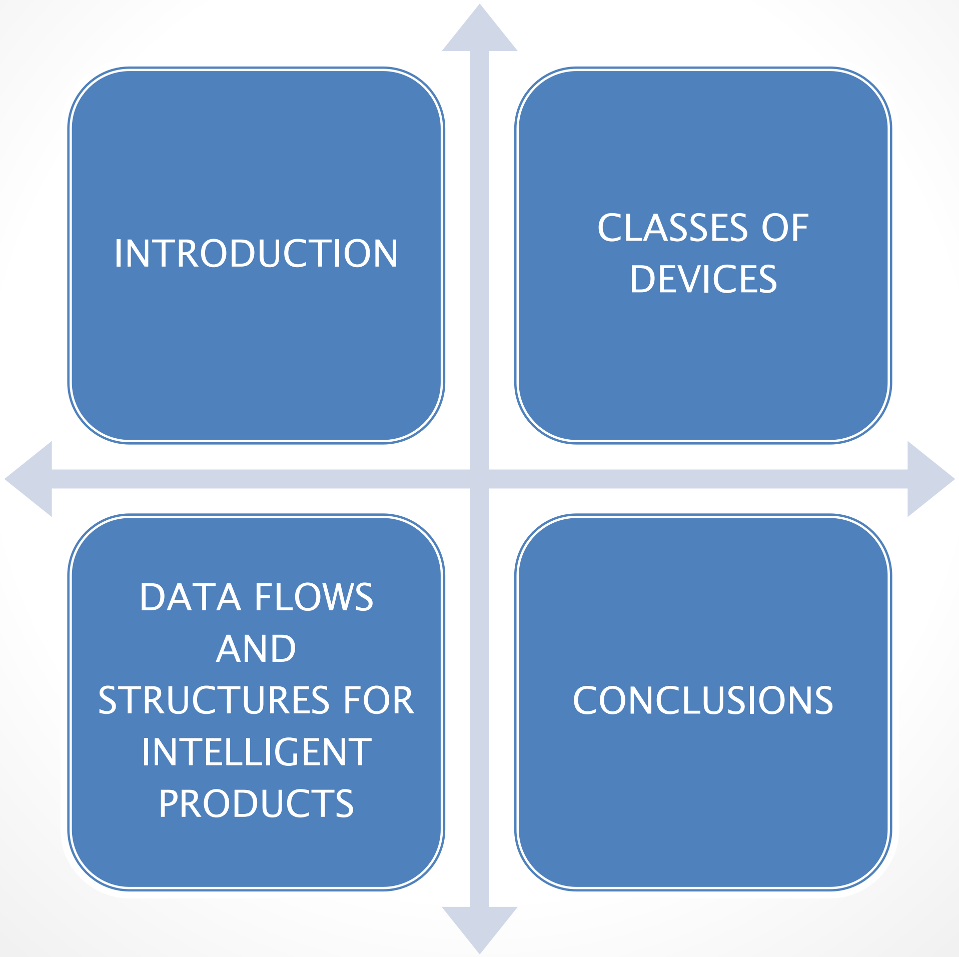
Formalized Information Representation for Intelligent Products in Service-Oriented Manufacturing

Services Development from INSEED in the Manufacturing and Supply Chain Domains

Theodor Borangiu, University Politehnica of Bucharest
theodor.borangiu@cimr.pub.ro

Agenda


INTRODUCTION
CLASSES OF DEVICES
DATA FLOWS AND STRUCTURES FOR INTELLIGENT PRODUCTS
CONCLUSIONS

Introduction
Intelligent Product

- What is an Intelligent Product?
- Intelligent Product in SOA Context

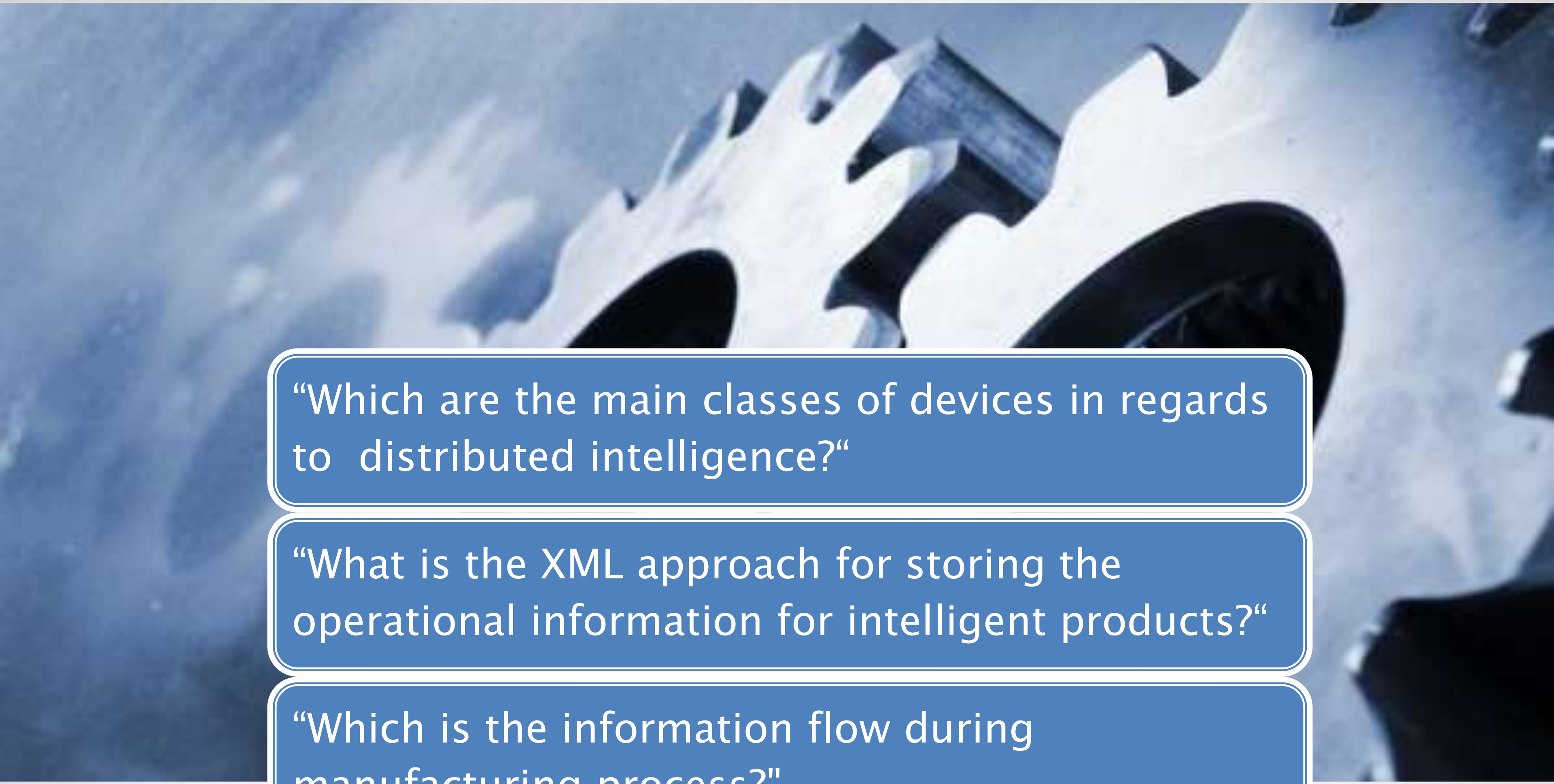
Service Oriented Architecture

Intelligent Product

Introduction
Intelligent Product Classification

Mayer (2009) introduces a classification that positions an intelligent product based on three directions:

- level of intelligence,
- location of intelligence, and
- aggregation level of intelligence



“Which are the main classes of devices in regards to distributed intelligence?“

“What is the XML approach for storing the operational information for intelligent products?“

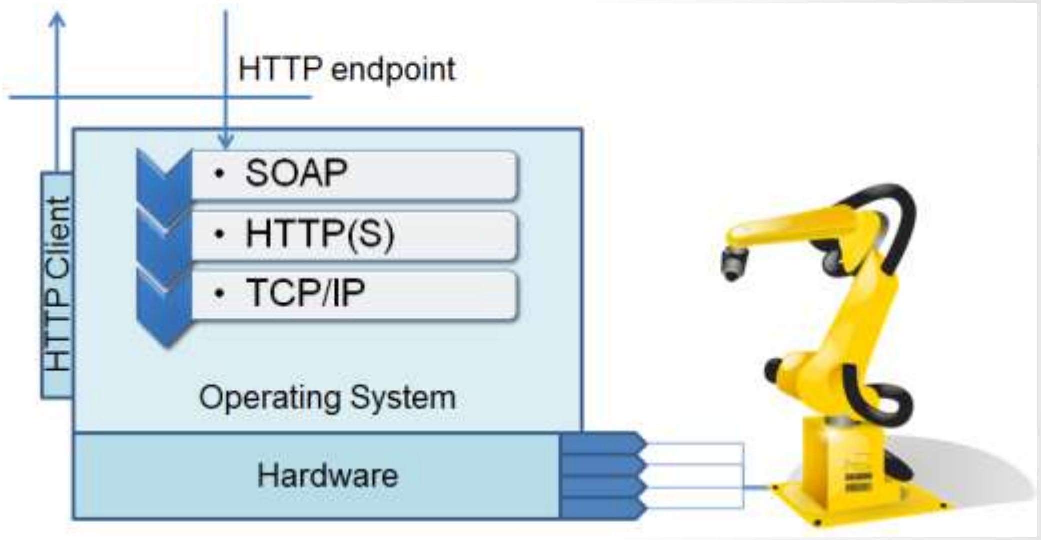
“Which is the information flow during manufacturing process?"

Questions to be answered

Classes of Devices

General Architecture of a SOA Enabled Device

Distributed intelligence and alignment to industry standards are two main prerequisites for organizing shop floor activities based on SOA paradigms



Classes of Devices

- Workstation is a standard computer equipped with a dedicated card for connecting to the device.
- The software is most of the time proprietary and allows programmatic control of the physical device.

Workstation assisted shop-floor device

- Capable to run an embedded Operating System attached to the shop floor physical device.
- Implements a full HTTP stack, capable to run both a HTTP server for hosting web service endpoints and a HTTP client for calling external web services.

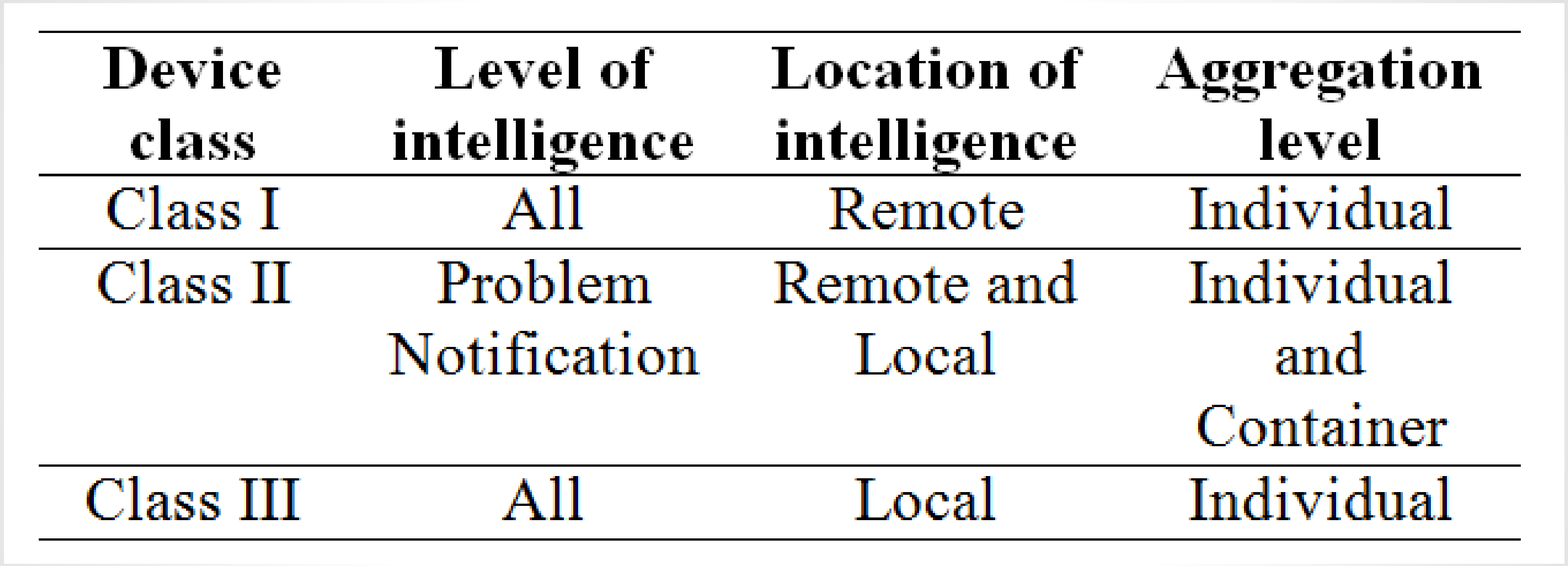
Embedded OS shop-floor device

- Able to run Data and CPU intensive applications in order to implement an intelligent behavior.
- We can include here mostly the Android equipped devices which can execute complex Java based agents (JADE/WADE).

Intelligent shop-floor device

Distributed intelligence and alignment to industry standards are two main prerequisites for organizing shop floor activities based on SOA paradigms.

Classes of Devices



Device class	Level of intelligence	Location of intelligence	Aggregation level
Class I	All	Remote	Individual
Class II	Problem Notification	Remote and Local	Individual and Container
Class III	All	Local	Individual

Based on the classification introduced by Meyer et al. (2009) the capabilities of the device classes are presented in the above table.

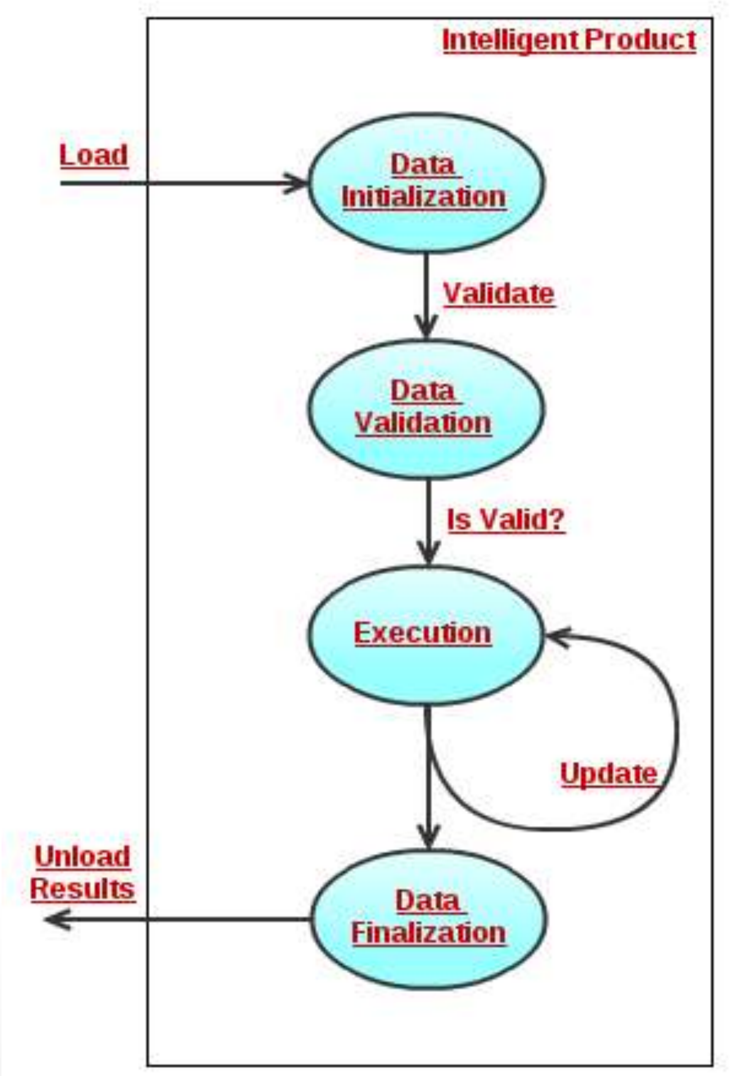
Classes of Devices

- The manufacturing system architecture can be implemented using any combination of the above devices, as all have in common the generic structure consisting in an informational part and the physical system.

- From a SOA perspective, the architecture would tend to be a *point to point choreography* if the lower class devices are used. With the introduction of higher class devices the trend is to use an *orchestrated architecture*, based on BPEL workflows and real time events.

Intelligent Product Datafow

- IP with 4 stage orchestration
- The orchestration can consume externally provided services for data validation and finalization
- Execution stage can be:
  - **Hierarchical mode** – sequential execution of pre-scheduled operations
  - **Heterarchical mode** – negotiation with other IP/resources of next operation



Intelligent Product Dataflow

- The process starts when the intelligent product, represented initially by the pallet carrier equipped with an embedded device, is inserted in the manufacturing line. At this point, the production information is loaded in the memory of the embedded device and <u>initialized</u>.

- The next step is the <u>data validation</u> activity composed from a XSD schema validation and a logical validation against the operations required for the execution of the product. The logical validation is required in order to detect situations like dead-lock scenarios that might occur.

Intelligent Product Dataflow

- Once the validation is complete each operation from the pre-loaded product recipe is executed by the shop-floor resources. The data structure is updated with the result of each operation <u>execution</u>, until all operations are completed.

- When the product pallet exits the manufacturing line, the <u>data finalization</u> phase is executed, where information about each operation execution is consolidated and unloaded from the product pallet.

Intelligent Product Data Type

- ProductIdentity complex type
  - Used to represent one product instance in the product batch. Contains the following items:
    - *Batch_ID – the ID of the product batch as a string*
    - *Product_ID – the internal ID of the product as a string*
    - *RFID_Tag – the RFID tag as a hex string*
    - *Product_Type – the product type of the given product*

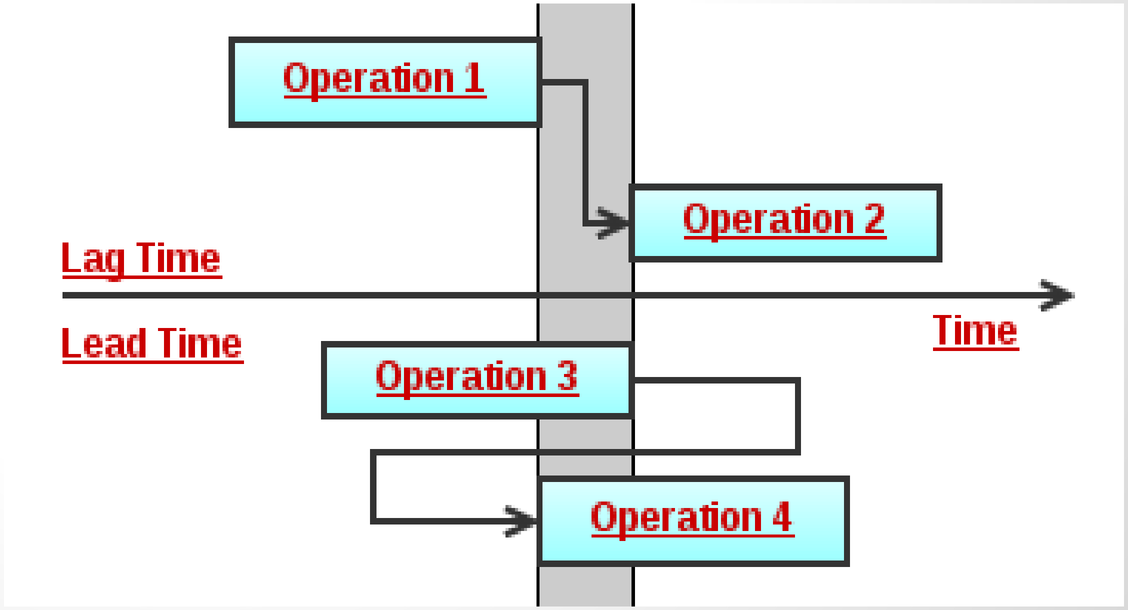
The Product_Type represents a pointer to the manufacturing system knowledge base that stores the recipe (list of operations and precedence) for that specific product.

Intelligent Product Data Type

- *ProductStatus* complex type
  - Used to represent the real time status of the product on the production line. Contains the following items:
    - *CriticalPath* – the order of operations executed in regards to Global_EF and Global_EFGlobal_LF
    - *TotalEnergyFootprint* – is computed in real time during product execution by summing the energy footprint recorded in each operation executed by a resource
    - *Global_EF/Global_LF*– global <u>early finish</u> and <u>late finish</u> estimations are updated based on the execution path that has been chosen

Lead and Lag Times for Operations

- **Duration:** represents the number of time units required for the operation to be performed by a resource
- **Lag Time:** represents the delay imposed on the relationship between two consecutive operations
- **Lead Time:** is an acceleration of the successor operation



Early Start and Late Start Estimations

- **ES (early start):**
  - The earliest time expressed in time units when the operation can start
  - Is computed based on the prerequisites tree of the operation by adding the duration of all the operations and the lag time and substantiating the lead time
  - This represents the optimistic start time for the operation
- **LS (late start):**
  - The latest time expressed in time units when the operation can start
  - Is computed form the prerequisite operations and the lead time is subtracted
  - This represents the pessimistic start time for the operation

Early Finish and Late Finish Estimations

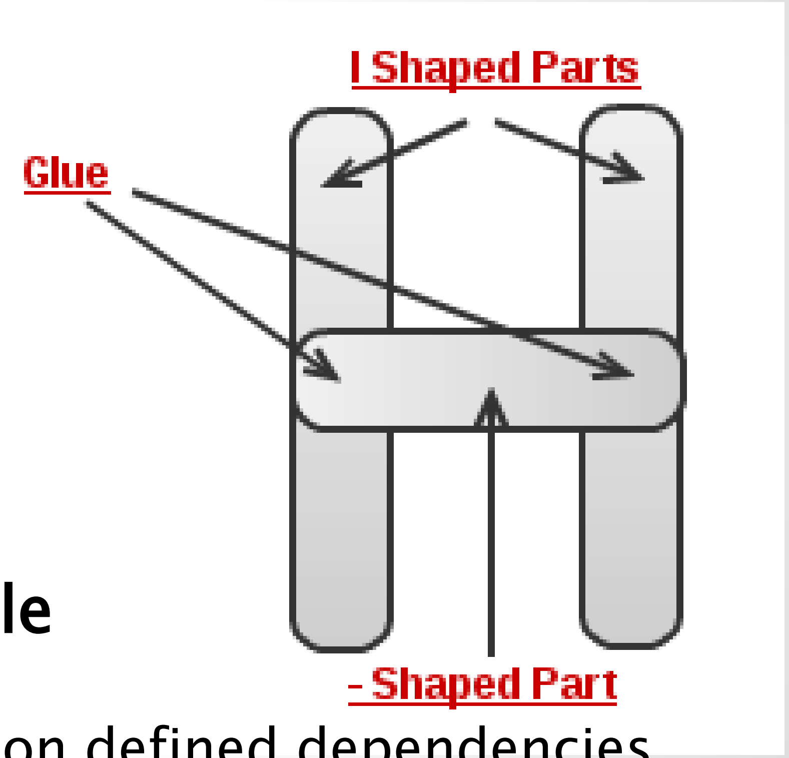
- **EF (early finish):**
  - Is the earliest time when the operation can be finished
  - Computed as ES + duration of the operation
- **LF (late finish):**
  - Is the latest time when the operation can be finished
  - Computed as LS + duration of the operation
- **Global_EF** and **Global_LF** of the product represents a range for the effective makespan (computed by adding EF/LF for each operation).
- **In Hierarchical mode**:
  - The execution order of the operations (critical path) is known in advance
  - EF/LF can be computed directly at the beginning
  - During execution EF/LF are used to validate that the initial schedule is still being followed by the system
- **In Heterarchical mode**:
  - EF/LF computed at run time after each operation
  - Used to estimate the makespan in real time

Intelligent Product Data Structure

- The example of a “H”-shaped mechanical product assembly is considered to demonstrate the modeling of the operation sequence using the presented data structure.

Intelligent Product Data Structure

- **Operations**:
  - Place_I_Left
  - Place_I_Right
  - Apply_Glue_Left
  - Apply_Glue_Right
  - Place_Center_Piece
  - Paint_Product



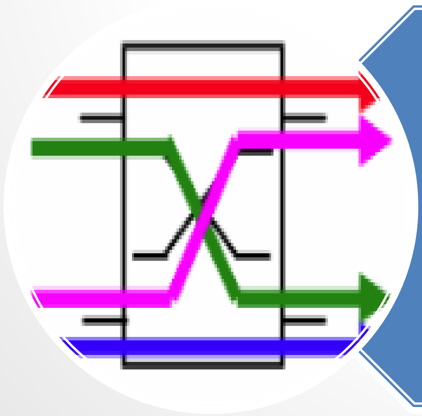
- **Several critical paths possible**
  - 1,2,3,4,5,6 or 1,3,2,4,5,6
  - Others are also possible based on defined dependencies (see next slide)

Intelligent Product Data Structure

```
<tns:Operations>
  <tns:Operation>
    <tns:ID>1</tns:ID>
    <tns:Code>Place_I_Left</tns:Code>
    <tns:Duration>10</tns:Duration>
    <tns:Prerequisites></tns:Prerequisites>
  </tns:Operation>
  <tns:Operation>
    <tns:ID>2</tns:ID>
    <tns:Code>Place_I_Right</tns:Code>
    <tns:Duration>10</tns:Duration>
    <tns:Prerequisites></tns:Prerequisites>
  </tns:Operation>
  <tns:Operation>
    <tns:ID>3</tns:ID>
    <tns:Code>Apply_Glue_Left</tns:Code>
    <tns:Duration>5</tns:Duration>
    <tns:Prerequisites>1</tns:Prerequisites>
  </tns:Operation>
  <tns:Operation>
    <tns:ID>4</tns:ID>
    <tns:Code>Apply_Glue_Right</tns:Code>
    <tns:Duration>5</tns:Duration>
    <tns:Prerequisites>2</tns:Prerequisites>
  </tns:Operation>
  <tns:Operation>
    <tns:ID>5</tns:ID>
    <tns:Code>Place_Center_Piece</tns:Code>
    <tns:Duration>10</tns:Duration>
    <tns:Prerequisites>3,4</tns:Prerequisites>
    <tns:LagTime>5</tns:Prerequisites>
  </tns:Operation>
  <tns:Operation>
    <tns:ID>6</tns:ID>
    <tns:Code>Paint_Product</tns:Code>
    <tns:Duration>15</tns:Duration>
    <tns:Prerequisites>5</tns:Prerequisites>
    <tns:LagTime>10</tns:Prerequisites>
  </tns:Operation>
</tns:Operations>
```

Conclusions

A formalization of the data structure stored in an intelligent product during the manufacturing process has been presented. Development in INSEED (service orientation & SOA in the manufacturing domain)



Future work on the XSD schema is aimed at reducing the size of the XML document. The size can be reduced especially when many operations of the same type are repeated many times.

Thank you!