TAMPERE UNIVERSITY OF TECHNOLOGY

Discrete Automation System Design

Assignment 02

Group member:

Farid Khosravi 267964

Mehdi Mahmoodpour 267958

Palash Halder 267962

ASE – 9426 Winter 2017

Contents

1. Contents 2
2. Objective 3
3. Introduction to system 3
4. Overview 4
5. Tools and environment 4
6. Design 4
7. Specification of production control problem 5
8. Group members' involvement 9

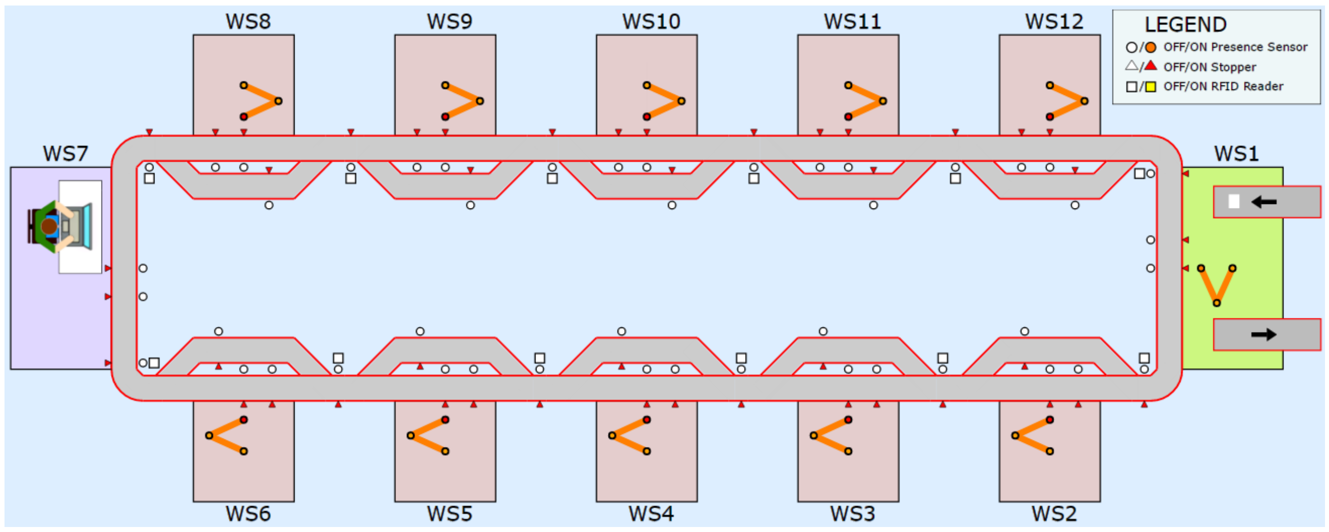
Topic: Agent-based control of a production system

Objective:

- Understanding of **Multi-Agent System (MAS)** concepts.
- Ability to model problem based on **MAS** methodology.
- Ability to implement distributed automation systems using **MAS** methodology.

Introduction to system:

- The factory line was being used to assemble and produce cell phone in Nokia Company. Now it is utilized for education purposes in TUT university.
- It consists of 12 work stations. Pallet loads/unloads to/from the line from work station 7 and the paper loads/unloads on the pallet in work station 1.
- Each work station contains one robot and two conveyors.
- Each pallet has a unique Pallet-ID number.
- The system can produce 729 different types of mobile phones (3 frames * 3 colors * 3 screens * 3 colors * 3 keyboards * 3 colors). In all the cases colors, can be either red, green, or blue.
- The following picture shows the overview of the FASTory line :



Overview:

A multi-agent system (MAS) is a loosely coupled network of agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver.

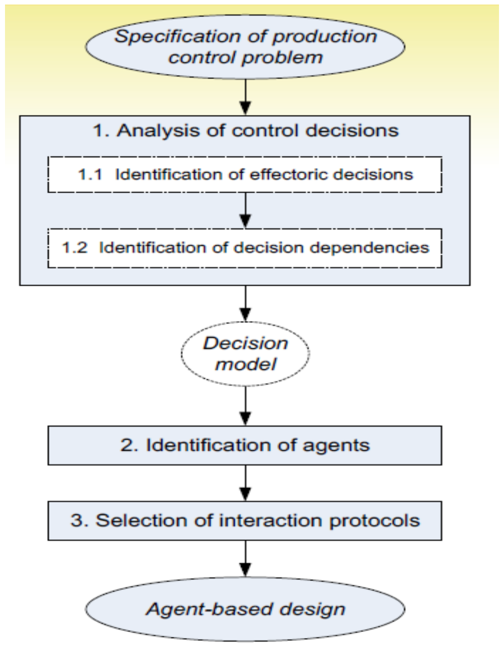
MAS consists of number of intelligent **agents** within the environment which are associated with each other to solve the modeled problem. Each **agent** is a computer system, situated in environment that is capable of flexible autonomous action in order to meet its design objectives. **Agents** can communicate with each other directly. In **MAS**, agents are not able to solve defined problem individually and problem solving can be solved in the presence of several decision-making entities, distributed inside the production system.

Tools and environment:

- Fastory line simulator.
- Visual Paradigm: In this assignment the "Visual paradigm "software is used to display the diagrams of components of the system based on the Unified Modeling Language (UML).
- Node.js: To implement the agents of production line.
- Webstorm: to run and implement the JavaScript codes.

Design:

In this assignment, the design of system is based on the following MAS methodology:


Specification of production control problem
1. Analysis of control decisions
1.1 Identification of effectoric decisions
1.2 Identification of decision dependencies
Decision model
2. Identification of agents
3. Selection of interaction protocols
Agent-based design

Specification of production control problem:

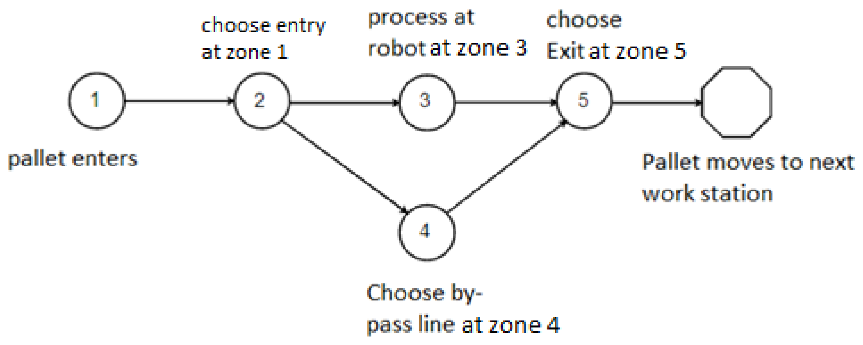
1. Analysis of control decisions:

❖ **1.1 Identification of effectoric decisions:**

The situations where control actions are necessary to be implemented during production process identified in this assignment as following.

- All events have been subscribed to the line. Once the **"Trigger"** subscribed event emits from simulator the work station agent and the pallet agent decides what action should be taken in order to reach the final goal of creating a mobile phone according to customer requirement.
- **Decision Space:** {Main line, Bypass line}
- **ID= "Zone1 Changed":** This is the entry point of work station. If the the workstation is busy it directs the pallet to bypass line. If the pallet needs service and work station is capable of doing the required service, then the pallet enters into work station. Otherwise, the pallet will be passed to by-pass line.
- **ID= "Zone3 Changed":** Here is the point that robot works on pallet. Work station starts to draw, load paper or unload pallet due to status of the pallet.

Following figure depicts the trigger diagram of project:


choose entry at zone 1
process at robot at zone 3
choose Exit at zone 5
1
2
3
5
pallet enters
Pallet moves to next work station
4
Choose by-pass line at zone 4

❖ **1.2 Identification of decision dependencies:**

Dependencies of task between work stations were defined due to capability of each work station and routes between workstations. Each pallet is directed to the nearest work station that is able to process the required service. The decision for next job to be done on the pallet is taken on the basis of pallet status which is a property of the pallet.

2. Identification of agents:

We have implemented discrete independent agents which can interact using HTTP.

We have implemented two types of agents as explained below.

- **Work stations:**

  Each work station including robot and conveyors, considered as an **agent**. Following attributes and methods were defined for each **WS**:

  Attributes:

  - **Work station name**: Work stations were named regarding to the layout of Fastoryline.
  - **Work station capability**: One of three colors (red, blue and green) were determined as a capability of each WS.
  - **Work station port**: A unique port number was given to each WS as a server.

  Methods:

  - **Runserver**: This method opens a server listening to the port and host name defined in attributes. It has own functions defined to set status, to set priority and to receive pallet data.
  - **Getcapability**: This method returns the capability of work stations.
  - **Getname**: This method gives the name of work stations.

- **Pallets:**

  Each new pallet loaded creates a new pallet agent. Following attribute and methods are defined for each pallet:

  Attributes:

  - **Pallet ID**: Each pallet has pallet id provided by simulator.
  - **Order ID**: Each pallet has unique order id.
  - **Phone requirements**: Defines the specification of customer requirements.
  - **Status**: Indicates what operations have been done on the pallet.
  - **Path**: Defines the path for pallet to visit.
  - **Port**: Unique port number for each pallet agent created.

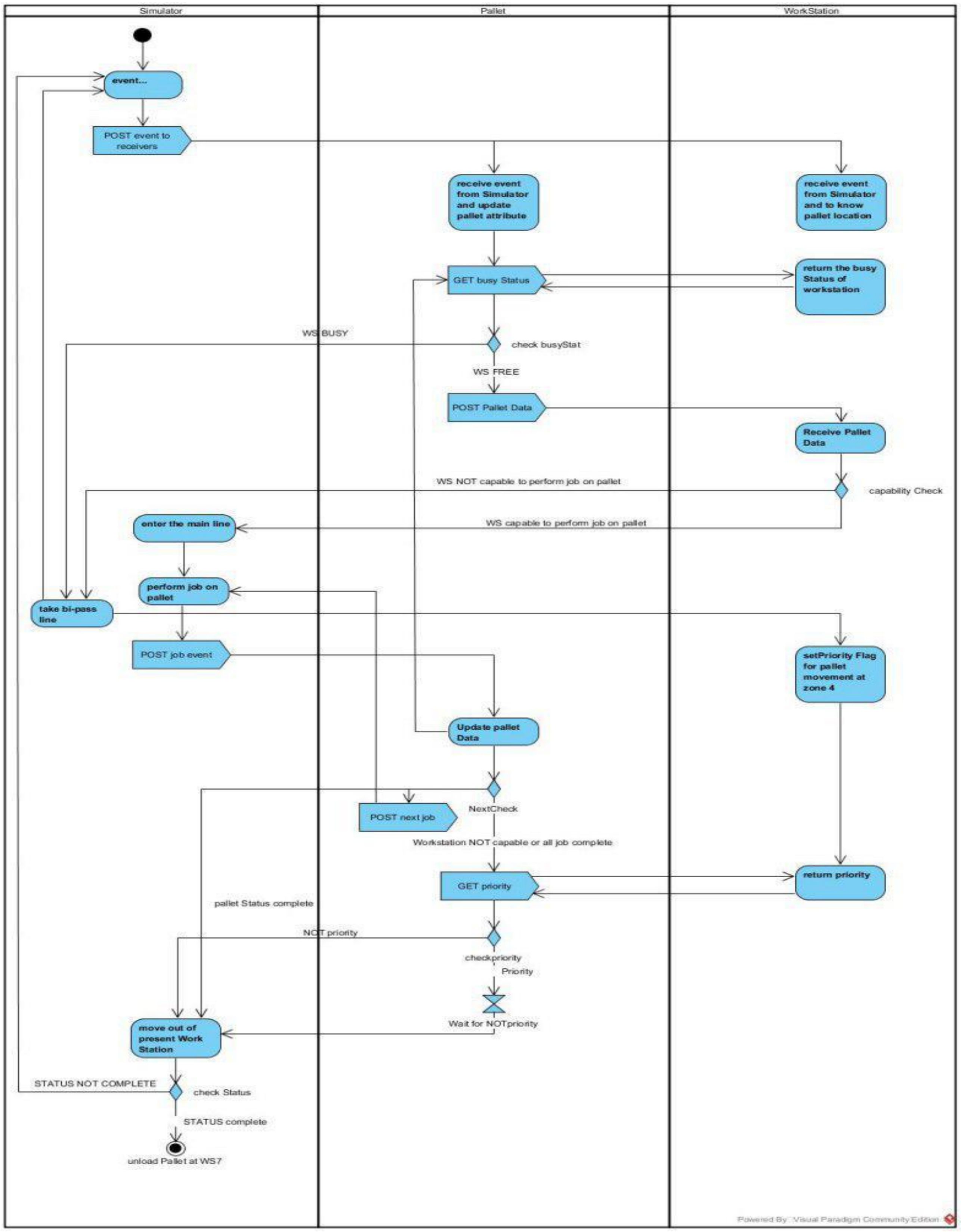
  Methods:

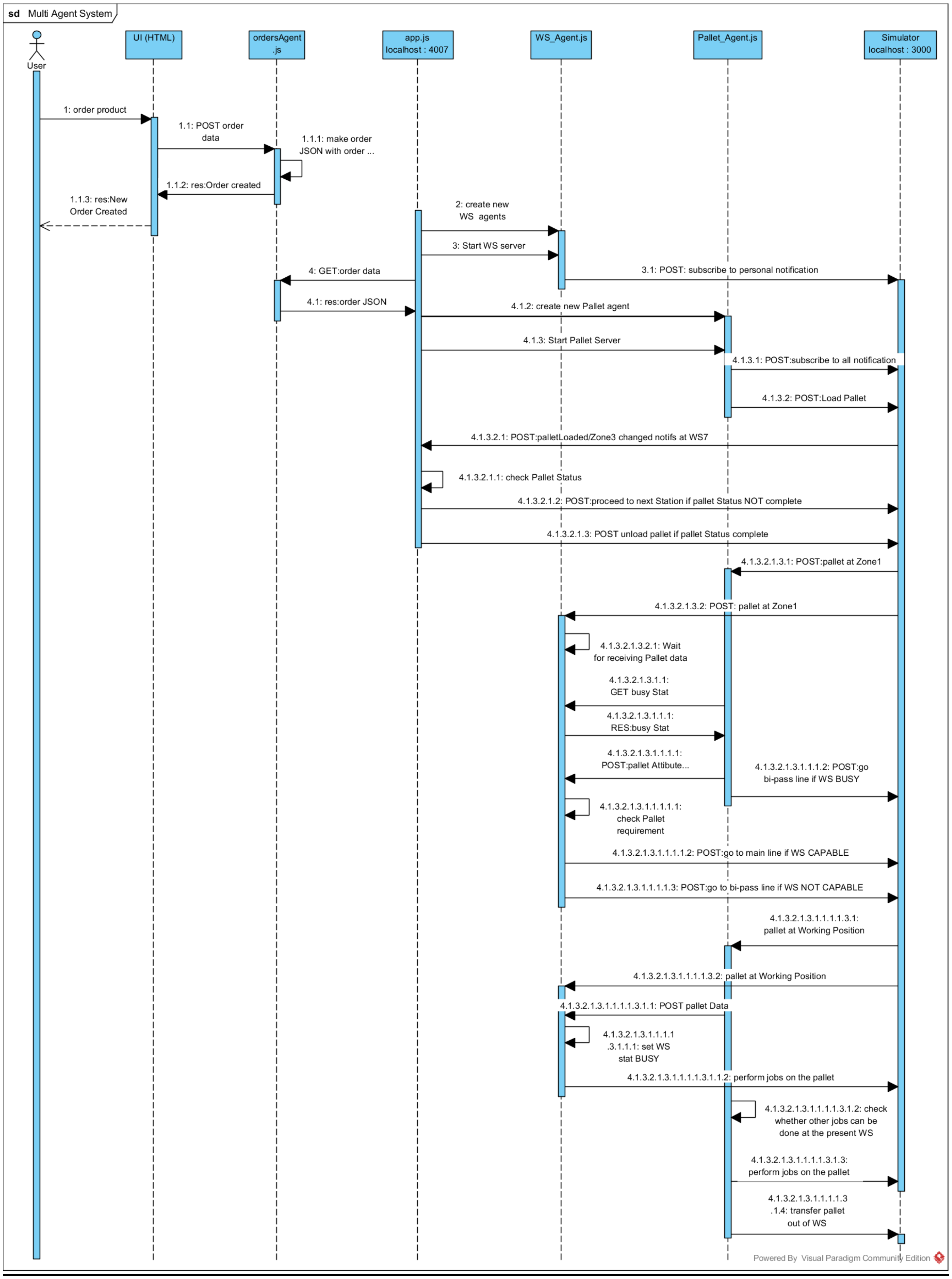
  - **Runserver**: this method opens a server listening to the port and host name defined in attributes. It has own functions defined to set status, to set priority and to send pallet data.

- **Setpath**: This method sets the path for pallets to visit.

3. Selection of interaction protocols:

Following activity diagram and sequence diagram depict the Interaction between agents in this assignment.


Simulator
Pallet
WorkStation
event...
POST event to receivers
receive event from Simulator and update pallet attribute
receive event from Simulator and to know pallet location
GET busy Status
return the busy Status of workstation
WS BUSY
check busyStat
WS FREE
POST Pallet Data
Receive Pallet Data
WS NOT capable to perform job on pallet
capability Check
WS capable to perform job on pallet
enter the main line
perform job on pallet
take bi-pass line
POST job event
setPriority Flag for pallet movement at zone 4
Update pallet Data
NextCheck
POST next job
Workstation NOT capable or all job complete
GET priority
return priority
pallet Status complete
NOT priority
checkpriority
Priority
Wait for NOTpriority
move out of present Work Station
STATUS NOT COMPLETE
check Status
STATUS complete
unload Pallet at WS7
Powered By: Visual Paradigm Community Edition


sd Multi Agent System
User
UI (HTML)
ordersAgent .js
app.js localhost : 4007
WS_Agent.js
Pallet_Agent.js
Simulator localhost : 3000
1: order product
1.1: POST order data
1.1.1: make order JSON with order ...
1.1.2: res:Order created
1.1.3: res:New Order Created
2: create new WS agents
3: Start WS server
3.1: POST: subscribe to personal notification
4: GET:order data
4.1: res:order JSON
4.1.2: create new Pallet agent
4.1.3: Start Pallet Server
4.1.3.1: POST:subscribe to all notification
4.1.3.2: POST:Load Pallet
4.1.3.2.1: POST:palletLoaded/Zone3 changed notifs at WS7
4.1.3.2.1.1: check Pallet Status
4.1.3.2.1.2: POST:proceed to next Station if pallet Status NOT complete
4.1.3.2.1.3: POST unload pallet if pallet Status complete
4.1.3.2.1.3.1: POST:pallet at Zone1
4.1.3.2.1.3.2: POST: pallet at Zone1
4.1.3.2.1.3.2.1: Wait for receiving Pallet data
4.1.3.2.1.3.1.1: GET busy Stat
4.1.3.2.1.3.1.1.1: RES:busy Stat
4.1.3.2.1.3.1.1.1.1: POST:pallet Attibute...
4.1.3.2.1.3.1.1.1.2: POST:go bi-pass line if WS BUSY
4.1.3.2.1.3.1.1.1.1.1: check Pallet requirement
4.1.3.2.1.3.1.1.1.1.2: POST:go to main line if WS CAPABLE
4.1.3.2.1.3.1.1.1.1.3: POST:go to bi-pass line if WS NOT CAPABLE
4.1.3.2.1.3.1.1.1.1.3.1: pallet at Working Position
4.1.3.2.1.3.1.1.1.1.3.2: pallet at Working Position
4.1.3.2.1.3.1.1.1.1.3.1.1: POST pallet Data
4.1.3.2.1.3.1.1.1.1 .3.1.1.1: set WS stat BUSY
4.1.3.2.1.3.1.1.1.1.3.1.1.2: perform jobs on the pallet
4.1.3.2.1.3.1.1.1.1.3.1.2: check whether other jobs can be done at the present WS
4.1.3.2.1.3.1.1.1.1.3.1.3: perform jobs on the pallet
4.1.3.2.1.3.1.1.1.1.3 .1.4: transfer pallet out of WS
Powered By Visual Paradigm Community Edition

Code:

- Fastory line simulator at **localhost:3000** is simulating the real production line as explained before. The events generated from the simulator represent the state of production and direct to the next steps.
- **app.js** is designed to represent system application at **localhost:4007**. We have used Express app to make our servers. Following functionalities were implemented within **app.js**:
  - Create work stations as an object of **WS_Agent** class: New objects were created using constructor to initialize the object created with a class. Each work station has three attributes: **name**, **capability** and **port number**. Then we run each work station as a server by using **runserver** method of **WS_Agent** class.
  - The HTML form performs a POST method to **localhost:4500/orders**, the **orders-agent.js** file. Here a local variable with the customer specification is created and it is sent to **app.js** using POST method when **app.js** server is running.
  - The pallets are automatically inserted after every 10 seconds till the order amount is reached. The new pallet object of **Pallet_Agent** class will be created and the **path** of pallet will be set using **searchCapability** and **setPath** methods. After all the requirements are fulfilled the pallet will be unloaded at workstation 7.
- **WS_Agent.js** is designed to define capabilities of workstations and allow the movement of pallets between workstations, explanation is provided in the code.
- **Pallet_Agent.js** is implemented to store the order attributes and exchange data with work stations and update its attributes when required. Explanation is provided in the code.

User interface:

- **Index.HTML** form is utilized to get the details of needed product from the user. User can submit a new order. The order contains the information of desired type of product (frame, keyboard, screen, color) and quantity. A post request is made to **localhost:4500/orders**. The post request is received by **orders-agent.js**.

GROUP MEMBERS’ INVOLVEMENT:

- Palash Halder: Design and implementation of Node.js code and making of HTML.
- Farid Khosravi: Design and implementation of Node.js code.
- Mehdi Mahmoodpour: Design and implementation of Node.js code.