

# Terms of Reference

This Iteration is submitted in partial fulfilment of the requirements for Course ASE-9316 Introduction to Industrial Informatics, Laboratory of Automation and Hydraulics, Tampere University of Technology, Finland.

# Iteration Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | Date | UP Phase | Remarks |
| 1 | 28.09.2017 | Inception | * System vision * WBS diagrams with the ProjectLibre * System Architecture |
| 2 | 13.10.2017 | Elaboration | * Refined System Vision * Cost/Benefit Evaluation * UML Diagrams (Use Case, Class, Activity, Sequence) |
| 3 | 31.10.2017 | Elaboration | * Report Restructured * State Diagram added * Refined Class Diagram * Refined Use Cases * Source Code Prototype * Basic GUI Development using SVG |
| 4 | 15.11.2017 |  | * Basic Structure of Knowledge Base Developed * Source Code Integration with the knowledge base. Values were retrieved from the knowledge base * UML diagrams revised * Final GUI Developed |

Contents

[Terms of Reference 2](#_gjdgxs)

[Iteration Version History 3](#_30j0zll)

[1.](#_1fob9te) System vision: 5

[2.](#_3znysh7) System Description: 5

[3.](#_2et92p0) System Architecture: 6

[4.](#_tyjcwt) Explanation of the Block Diagram: 6

[5.](#_3dy6vkm) Explanation of the WBS 7

[Inception phase: 7](#_1t3h5sf)

[Elaboration Phase: 7](#_4d34og8)

[Construction Phase: 8](#_2s8eyo1)

[Transition Phase: 9](#_17dp8vu)

[6.](#_3rdcrjn) COST/BENEFIT ANALYSIS 10

[7.](#_26in1rg) UML Diagrams 11

[Class Diagram 11](#_lnxbz9)

[USE Case Diagram 12](#_35nkun2)

[Activity Diagram 13](#_1ksv4uv)

[Sequence Diagram 14](#_44sinio)

[State Diagram 15](#_2jxsxqh)

[8.](#_z337ya) FRONT-END DEVELOPMENT 17

[9. BACK-END DEVELOPMENT (INITIAL PROTOTYPE) 18](#_3j2qqm3)

# System vision:

To develop a knowledge driven, monitoring and supervisory control system of the FASTory Line using knowledge base (OWL) to make operational decisions for carrying out the required tasks. The implementation of the system must be economically feasible and take the form of iterative and incremental development process following the Unified Process methodology in the following sequential steps; Inception, Elaboration, Construction, and Transition.

**System Capabilities:** The aim of the complete system building is to enable the control of the FASTory system through the web interface so it can work as the basic component of large cyber physical system in big automation environment. The system (KDMOS) is aimed to work using a web interface and execute the tasks (pallets moving and assembling) automatically once commanded and able to respond in sequence of tasks to accomplish a complicated task.

**System Business capabilities:** Business capabilities/ market scope directly depends on the quality of the developed system. So, to see business opportunities, the identification of the potential costumers of the product and customers for the product benefits is mandatory to done in parallel. So, the developed system should deliver value to the customer by replacing manual tasks by automation with developed system.

# System Description:

The FASTory line includes the main components of a production line; robots, transportation system, tools, end effectors, raw material, working stations, loading and unloading stations and a buffer station.

The system consists of the following:

* Conveyor- Each workstation in the FASTory line consist of two conveyors; main and bypass. The main conveyor is used if the pallet requires service from the work cell. Meanwhile, the bypass is used if the work cell is in busy state (another pallet(s) [max 2 pallets] are in the cell) to bypass the pallet to the next work cell. Both Conveyors (Main and Bypass) are controlled by a single RTU.
* Work Stations-

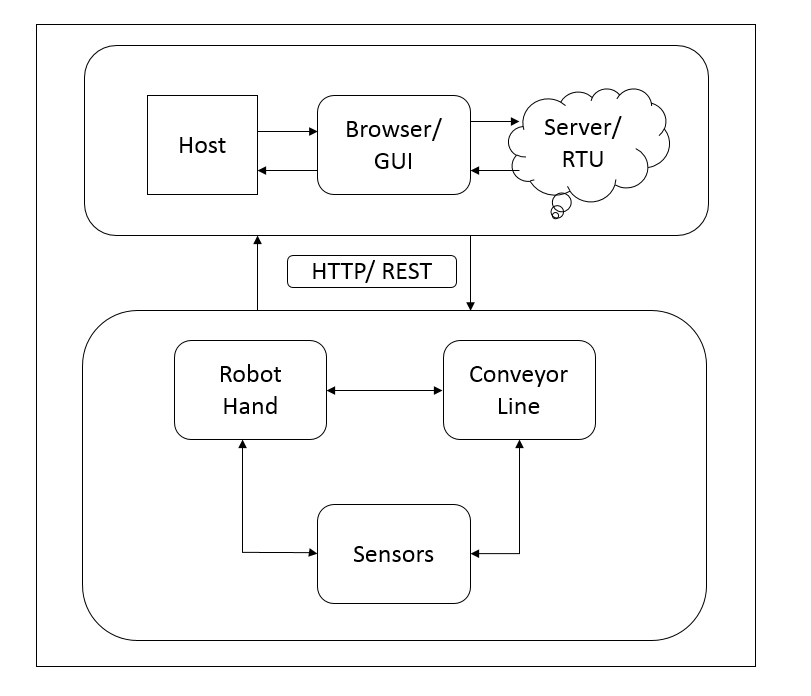
In FASTory there are 12 work stations in total, each work station contains one conveyor and one robot, for each conveyor there are several zones and each zone has one presence sensor to detect the presence of the pallet, one stopper to stop the pallet in the zone and each entrance zone for each work station has an RFID reader for pallet recognition.

There are 10 identical work stations which draw the main parts of three models of a mobile phone (WS 2-6 and 8-12). The remaining 2 workstations are WS1 used for loading raw material and unloading the product, WS7 is used for loading and unloading the pallet.

* Robots- The factory line uses SONY SCARA robots for production. Each robot is represented as an RTU in the line.
* RTUs- The FASTory is equipped with INICO S1000 Remote Terminal Units (RTUs). There are RTUs with unique IP address for each robot and conveyor in a work cell.

# System Architecture:

This following block diagram shows a basic architecture and steps of the intended project. In FASTory



*Figure 1: Block Diagram of System Architecture*

# Explanation of the Block Diagram:

The flow chart above shows the communication and process will be executed in the system architecture. There are multiple workstation that serve various purpose. this flow chart shows how a basic step is done.

The architecture follows server-client based communication. In user computer a host server is setup with a graphical user interface. NodeJs will be used along with HTTP protocol using REST web services to communicate with the INICO RTUs in the FASTory workstations. INICO S1000 remote terminal units located within each workcell is directly connected with robots and conveyors and it can send command and get sensor data to and from the workcells. Workstation then executes the command accordingly and send feedback to the server application which then displays the information to the user via the graphical interface.

# Explanation of the WBS

## Inception phase:

During the inception phase, a case for the system is established and the boundaries of the project scope is determined.

* **Analyze Project Requirement:** The requirements of the project is first studied.
* **Establish System Vision:** The purpose: justify strategic importance of new system
* **Prepare the Work Breakdown Structure:** WBS is subsequently prepared using a Top-Down approach using the scheduling tool Project Libre and the dependencies between the tasks are studied. Time required to carry out each task is studied and entered alongside each task. Key metrics such as Critical path, Slack time and Milestones are identified and a software generated Gantt chart is prepared
* **Initial Risk Assessment:** Risks involved with the project are identified and its technological feasibility (technological proficiency of group members), schedule feasibility (course workload for the semester).
* **Approximate Resource Utilization:** The tasks are allocated to various group members based on the results findings of the assessment in the previous step.
* **Estimate on Budget:** economic feasibility (Cost/benefit analysis) are studied.

At the end of the inception phase is the first major project milestone: the Project Objectives Milestone. The project may be cancelled or considerably re-thought if it fails to pass this milestone.

## Elaboration Phase:

The primary goals of Elaboration are to address known risk factors and to establish and validate the system architecture.

* **Analyze Use Case models:** Prepare Use Case Models for the system
* **Finalizing Architecture:** The architecture of the system to be developed is finalized.
* **Finalizing software requirements:** The integrated development environment for the system is agreed upon. In this case it was WebStorm, Jetbrains**.** Also, a basic pseudo code is designed
* **Finalizing the hardware requirements:** The hardware requirement for the project is next finalized. Most of it are them are available in the Fastory Line.
* **Analyzing Probability of Risk:** Based on the selected architecture, are finalized hardware and software a final risk assessment is carried out before moving to the next phase, i.e Construction.
* **Finalize Budget:** The budget for the project is finalized at the end of this stage.

At the end of the elaboration phase is the second important project milestone, the Project Architecture. The project may be aborted or considerably re-thought if it fails to pass this milestone.

## Construction Phase:

During the construction phase, all remaining components and application features are developed and integrated into the product, and all features are thoroughly tested.

* **Software development:** The software components of system is being constructed by using suitable web development tools and platforms. This part also involves development of different communication mediums which allows us to communicate between factory line with front end user interface.
* **Build user Interface:**

This project part consists of building front-end UI and data retrieval implementation. The basic front-end UI will be built first and since it is an iterative activity. The modifications will happen later after inclusion of various operational features and experiencing different user experiences.

* **Introduce knowledge base:**

The processing of data and building OWL Knowledge base will be completed in this phase. In order to develop a pre-knowledge based system an informative set of data collection is also included in this phase.

* **Prototype Testing:** It involves the testing and simulations of different software blocks on the Fastory Simulator.
* **Modification and debugging of Prototype:** Depending on the simulations and prototype testing results, some necessary modification to pre-developed prototype can be done in this span of time.
* **Project Deployment:** The software hardware integration and testing phase is planned for this phase. It includes several iterations since it is different to replicate the simulation results on hardware. Finding of the hardware constrains and implementation restrictions for full scale deployment is the major part of this phase. Other tasks include researching suitable methodology for project integration and successful operational deployment.
* **Project Output Assessment:** A detailed comparison between planned project proposal and achieved targets is carried out. This can be taken as an experience for future projects to avoid setting unrealistic or unclear goals.

At the end of the construction phase is the third major project milestone -Initial Project Operational Capability Milestone. At this point we decide if the project is ready to go operational, without exposing the it to high risks. This release is often called a “beta” release.

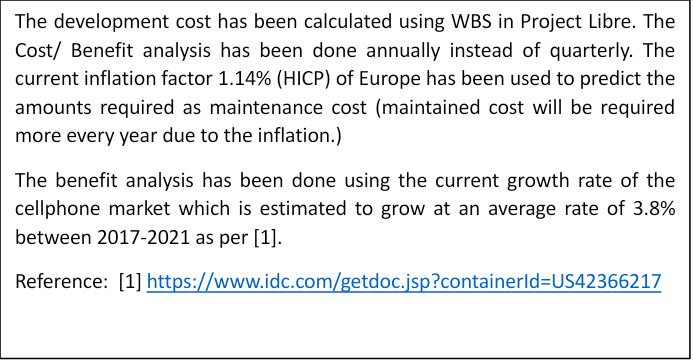
## Transition Phase:

The purpose of the transition phase is to transition the software product to the user community.

* **Detailed Analysis of beta version:** The pre-release version is analyzed.
* **Present Demo for user:** A demo is presented to the user/customer.
* **Analyze user feedback on Demo:** Customer/User feedback is analyzed
* **Make further correction / improvements:** Any necessary corrections stemming from user feedback is incorporated.
* **Submit the final version:** Final version is released/submitted.

At the end of the transition phase is the fourth important project milestone, the Product Release Milestone.

# COST/BENEFIT ANALYSIS

![](data:None;base64,iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAABHNCSVQICAgIfAhkiAAAAAtJREFUCB1jYAACAAAFAAGNu5vzAAAAAElFTkSuQmCC)

# UML Diagrams

## Class Diagram

A screenshot of a cell phone

Description generated with high confidence

*Figure 2: UML Class Diagram*

Figure 2 represents the Class Diagrams. There are 12 workstations so each workstations have their designated number.

At this point, it was thought that the individual capabilities in terms of what colour is to be drawn is given to each of them. Each of them have 4 zones (5 zones but 5th is 1st of adjacent workstation)

It was thought to run each workstations as individual servers with a common url and different port numbers.

A buffer attribute to hold the status of the buffer and status to know the activity of the workstation.

Methods:

* runServer()- to run the individual Servers of the workstations
* getOrder() – to get the order from the operator
* callNext() - to make query to the fuseki server to receive the next url
* request() - to instruct S1000 to perform the job based on the url received from fuseki

## USE Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the ONLY the user is involved.

A close up of a map

Description generated with high confidence

*Figure 3: Use Case Diagram*

The user is able to both place and order and view the current system status after logging in the system. The Logic Use Case include an Authentication Use Case. Back end commands are issued once the Order is placed in the GUI.

Additionally the user can get a report of what was done.

Activity Diagram

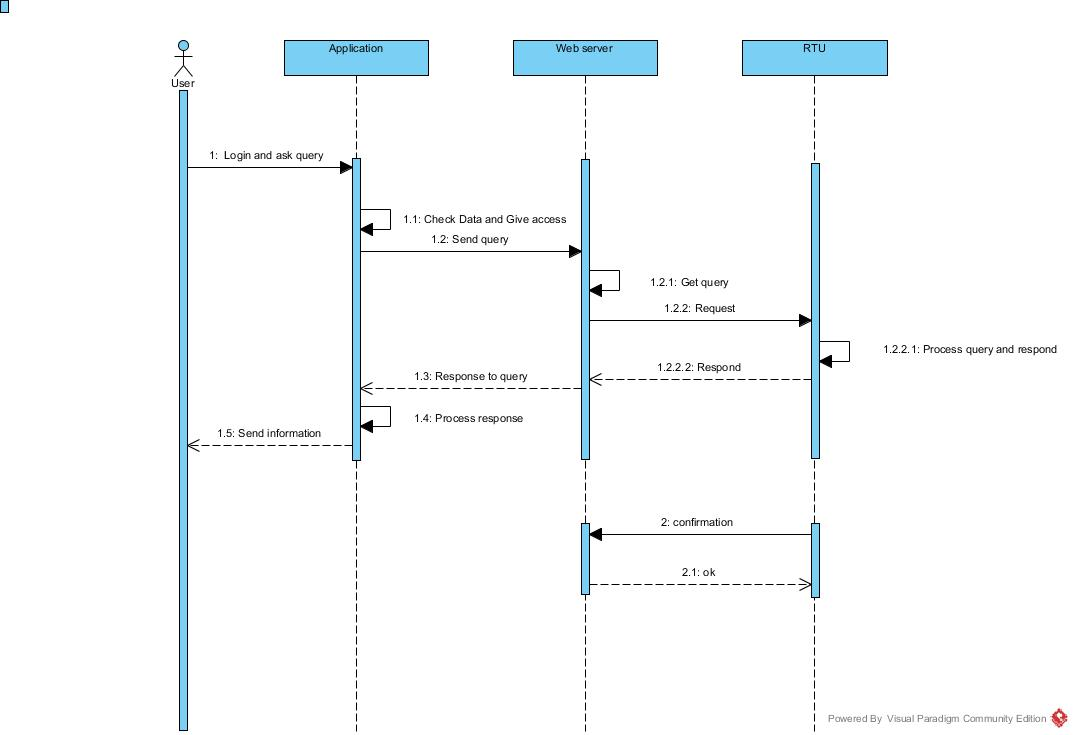
A screenshot of a cell phone

Description generated with high confidence

*Figure 4: Activity Diagram*

The activity diagram represents the workflows of activities and actions that happen in the product manufacturing process after taking specific order from customer in a step by step manner. The solid black circle represents initial node, the starting point of the process. The chamfered rectangle represents individual activity while the diamond shape represents decision making/condition check and at the end is a solid black circle with white edges that is used to represent last step in the process. The process is divided into five major vertical swim lines to represent five major blocks in the system. Two horizontal swim lines are used to divide two different division of process, first, status checking and credentials checking, second, production process in workstation. The black rectangle is used for joining of multiple activities with a single interface. Operator starts the system which is a host computer, which has a web interface that ask for user credentials. All the individual active components of workstation are connected with INICO s1000 RTU which is also connected with the server in real-time. So, status of the components is always updated. After credential verification next process is to produce the item. User select model from various options and place order through web user interface. This information is passed to the server and the production process starts in workstation. After every command execution machine status is updated simultaneously and demonstrated through the GUI in host computer.

Sequence Diagram



*Figure 5: Sequence Diagram*

The user logs in the UI and places an order. This is translated to an order and the query is sent to the RTU. The RTU then sends a response back which is translated to the user as graphics in the user interface. The subsequent actions are visually represented in the GUI.

State Diagram

A close up of a map

Description generated with high confidence

*Figure 6: State Diagram*

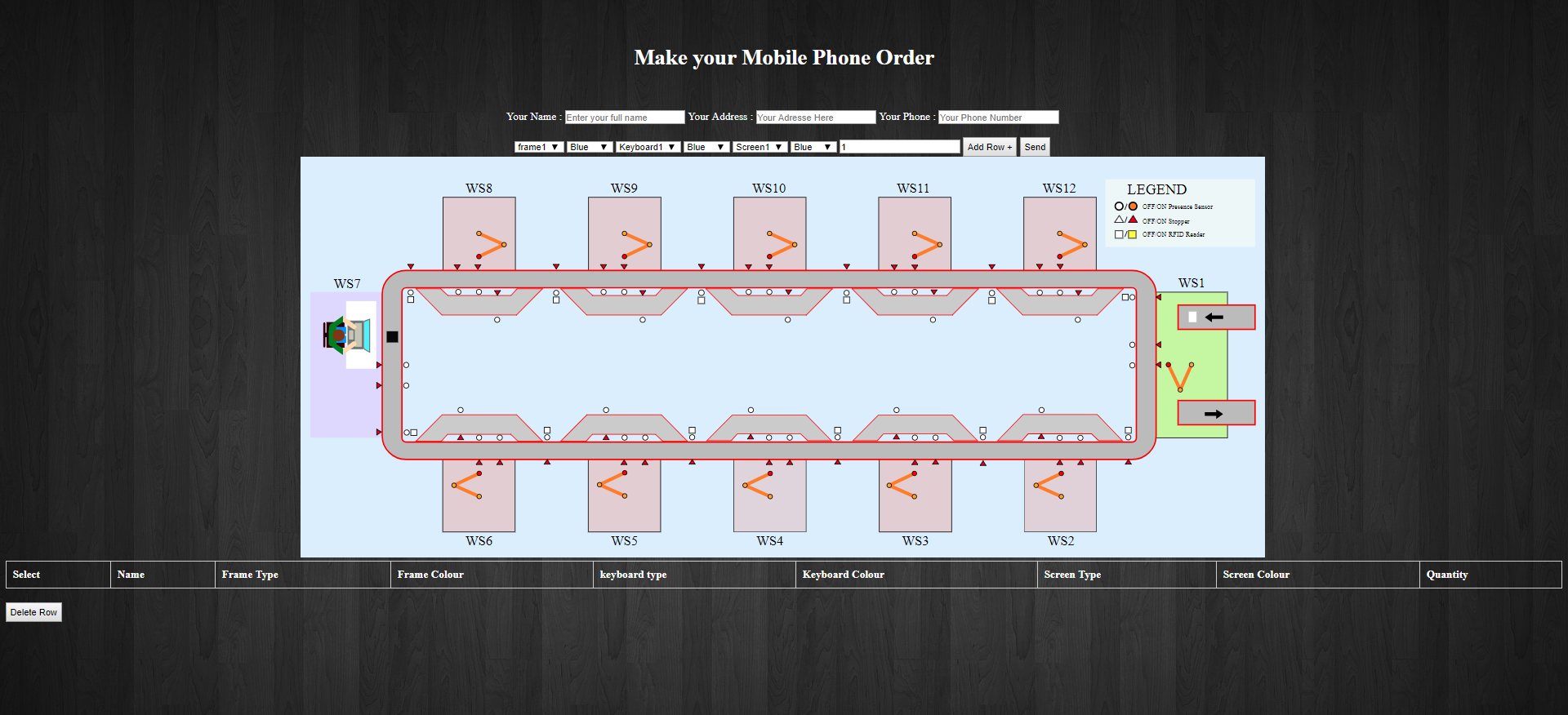
The above Figure (6) shows the State Diagram for the workstation class. It consists of the following states.

1. First the System start state is when the System is powered on and that takes it to the **SETUP** state where configuration (if any) is done on the workstation. It involves deliberate action on the workstation.
2. All the workstations are now in the **STARVED** state in the sense that it is ready to receive an item from an upstream workstation, but no item is available. The workstation’s

working area is available to work but it is not being given anything to build. There is no unfinished work within the workstation and there are no items available to move into the workstation.

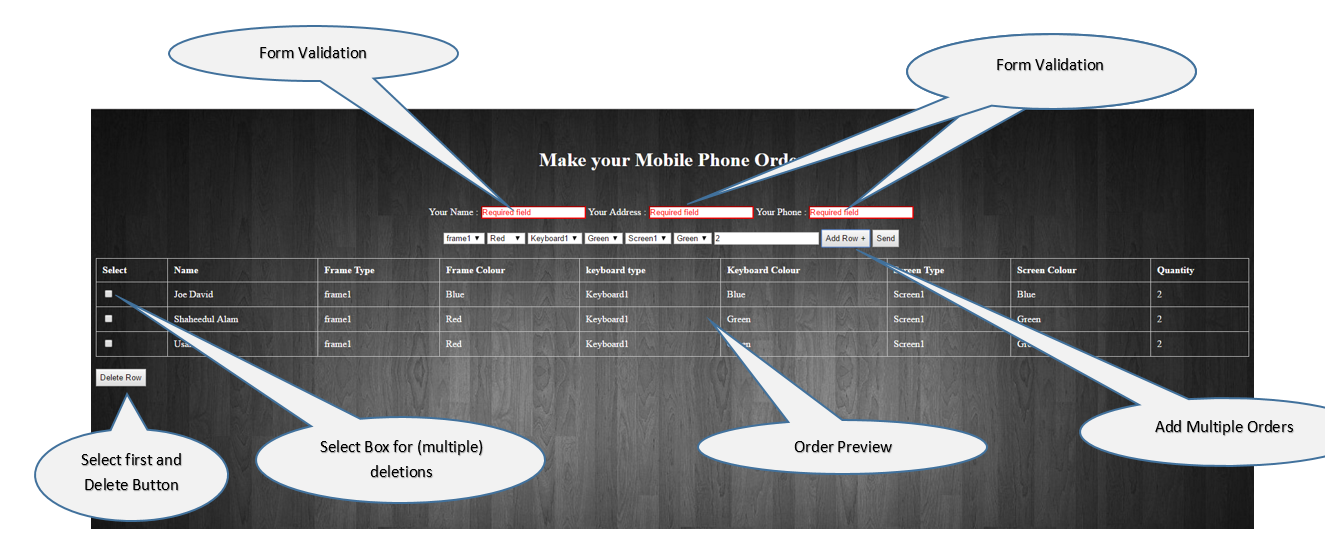
1. When an order is placed, and the pallet arrives at the workstation, it changes to the **ACTIVE** state. In this state, an item is available, but no recipe is being executed. This includes time intervals when items are transferred in and out of the appropriate zones.
2. Once the pallet reaches zone 3 of the workstation, the workstation moves to the **EXECUTING** state where the robot executes the recipe and it continues to do so without external intervention.
3. Once the recipe is executed it can either go to the **STARVED** state if is able to pass on the pallet to the next workstation or to the **BLOCKED** state if it’s not able to do so due to the presence of another pallet in the adjacent zone.
4. In any case, if there is a mechanical failure of any of the equipment in a workstation it goes.

# FRONT-END DEVELOPMENT



*Figure 7: GUI User Interface*

## Features



# 9. BACK-END DEVELOPMENT (INITIAL PROTOTYPE)

This section is to be read in conjunction with the class diagram.

The back-end development is only at the initial stage. Although the attributes for the class has been described its methods have neither been defined nor described. A tentative description of the methods can be found in the class diagram but however as mentioned they have not been implemented in the code as yet. The only method implemented is runServer().

var express = require('express');

*//var app = express();*

var request = require('request');

var bodyParser = require('body-parser');

*//define variables*

var setValue;

var query1 = "query=PREFIX iii:<http://www.manufacturing.com/cellphones.owl#> SELECT\* WHERE{iii:conveyor\_1 iii:transZone45 ?url}";

*//define query for the Knowledge base*

var optionsKB = {

method: 'post',

body: " ",

*//json: true, // Use,If you are sending JSON data*

url: "http://127.0.0.1:3032/iii2017/query",

headers: {

'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8',

'Accept':'application/sparql-results+json,\*/\*;q=0.9'

}

};

*//define query for S1000*

var optionsCntrl = {

method: 'post',

body: {destUrl:''}, *// Javascript object payload*

json: true,

url: '',

headers: {

'Content-Type': 'application/json'

}

};

var workstation= function (wsnumber, capability)

{

this.wsnumber = wsnumber;

this.capability = capability;

this.zone1=false;

this.zone2=false;

this.zone3=false;

this.zone4=false;

this.zone5=false;

this.port = 1234;

this.url = "127.0.0.1";

this.buffer = 'free';

this.status = 'free';

};

workstation.prototype.runServer = function (port)

{ var app = express();

this.port = port;

var ref1 = this; *// explanation?*

var hostname = this.url; *// can write direct url*

app.get('/', function (req, res) {

res.statusCode = 200;

res.setHeader('Content-Type', 'text/plain');

res.write('\nI am workstation ' + ref1.wsnumber);

res.write('\nMy capability is: ' + ref1.capability);

res.end('\nWorkstation ' + ref1.wsnumber + ' is running.');

});

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

*//HANDLE NOTIFICATION FROM THE FASTORY SIMULATOR*

app.post('/notifs/', function (req, res) {

console.log(req.body);

switch (req.body.id) {

case "Z1\_Changed":

console.log(req.body.results);

callNext(query1);

break;

case "Z2\_Changed":

break;

case "Z3\_Changed":

break;

case "Z4\_Changed":

break;

case "Z5\_Changed":

break;

}

});

app.listen(port, hostname, function () {

console.log(' WorkStation Server WS' + ref1.wsnumber +' is running at http://' + hostname + ':' + port);

});

if((ref1.wsnumber>0)&&(ref1.wsnumber<10)) {

request.post(' http://localhost:3000/RTU/SimCNV'+ref1.wsnumber+'/events/Z1\_Changed/notifs', {form: {destUrl: "http://localhost:600"+ref1.wsnumber+"/notifs/"}});

request.post(' http://localhost:3000/RTU/SimCNV'+ref1.wsnumber+'/events/Z2\_Changed/notifs', {form: {destUrl: "http://localhost:600"+ref1.wsnumber+"/notifs/"}});

request.post(' http://localhost:3000/RTU/SimCNV'+ref1.wsnumber+'/events/Z3\_Changed/notifs', {form: {destUrl: "http://localhost:600"+ref1.wsnumber+"/notifs/"}});

*// request.post(' http://localhost:3000/RTU/SimCNV'+ref1.wsnumber+'/events/Z5\_Changed/notifs', {form: {destUrl: "http://localhost:600"+ref1.wsnumber+"/notifs/"}});*

if((ref1.wsnumber!=1)&&(ref1.wsnumber!=7))

{

request.post(' http://localhost:3000/RTU/SimCNV'+ref1.wsnumber+'/events/Z4\_Changed/notifs', {form: {destUrl: "http://localhost:600"+ref1.wsnumber+"/notifs/"}});

request.post('http://localhost:3000/RTU/SimROB'+ref1.wsnumber+'/events/DrawEndExecution/notifs', {form: {destUrl: "http://localhost:600"+ref1.wsnumber+"/notifs/"}});

}

}

if((ref1.wsnumber>9)&&(ref1.wsnumber<13)) {

request.post(' http://localhost:3000/RTU/SimCNV'+ref1.wsnumber+'/events/Z1\_Changed/notifs', {form: {destUrl: "http://localhost:60"+ref1.wsnumber+"/notifs/"}});

request.post(' http://localhost:3000/RTU/SimCNV'+ref1.wsnumber+'/events/Z2\_Changed/notifs', {form: {destUrl: "http://localhost:60"+ref1.wsnumber+"/notifs/"}});

request.post(' http://localhost:3000/RTU/SimCNV'+ref1.wsnumber+'/events/Z3\_Changed/notifs', {form: {destUrl: "http://localhost:60"+ref1.wsnumber+"/notifs/"}});

request.post(' http://localhost:3000/RTU/SimCNV'+ref1.wsnumber+'/events/Z4\_Changed/notifs', {form: {destUrl: "http://localhost:60"+ref1.wsnumber+"/notifs/"}});

*// request.post(' http://localhost:3000/RTU/SimCNV'+ref1.wsnumber+'/events/Z5\_Changed/notifs', {form: {destUrl: "http://localhost:60"+ref1.wsnumber+"/notifs/"}});*

request.post('http://localhost:3000/RTU/SimROB'+ref1.wsnumber+'/events/DrawEndExecution/notifs', {form: {destUrl: "http://localhost:60"+ref1.wsnumber+"/notifs/"}});

}

};

*/\**

*//function for receiving order value from the operator*

*function callOrder(ord){*

*console.log(ord);*

*optionsKB.body = ord; //Assembly of the new query*

*console.log(optionsKB);*

*request(optionsKB, function (err, res, body) {*

*if (err) {*

*console.log('Error :', err);*

*return;*

*}*

*for(var i = 0; i<body.results.bindings.length; i++) {*

*var next = body.results.bindings[i].url.value;*

*setValue = next;*

*console.log(setValue);*

*}*

*});*

*}*

*\*/*

*//function to make query to the fuseki server to receive the next url*

function callNext(curr){

*// console.log(curr);*

optionsKB.body = curr; *//Assembly of the new query*

*// console.log(optionsKB);*

request(optionsKB, function (err, res, body) {

if (err) {

console.log('Error :', err);

return;

}

console.log(body)

*// for(var i = 0; i<body.results.bindings.length; i++) {*

*// var next = body.results.bindings[i].url.value;*

*// //var setValue = next;*

*// console.log(next)*

*// }*

});

}

*/\**

*//function to instruct S1000 to perform the job based on the url received from fuseki*

*function requestOut(num){*

*optionsCntrl.url = num;*

*console.log(optionsCntrl);*

*request(optionsCntrl, function (err, res, body) {*

*if (err) {*

*console.log('Error :', err);*

*return;*

*}*

*console.log(optionsCntrl);*

*console.log(' Body :', body);*

*});*

*}*

*\*/*

var ws1 = new workstation(1,'paper');

var ws2 = new workstation(2,'red');

var ws3 = new workstation(3,'blue');

var ws4 = new workstation(4,'green');

var ws5 = new workstation(5,'red');

var ws6 = new workstation(6, 'blue');

var ws7 = new workstation(7,'loadpallet');

var ws8 = new workstation(8,'green');

var ws9 = new workstation(9, 'red');

var ws10 = new workstation(10, 'blue');

var ws11 = new workstation(11, 'green');

var ws12 = new workstation(12, 'red');

ws1.runServer(6001);

ws2.runServer(6002);

ws3.runServer(6003);

ws4.runServer(6004);

ws5.runServer(6005);

ws6.runServer(6006);

ws7.runServer(6007);

ws8.runServer(6008);

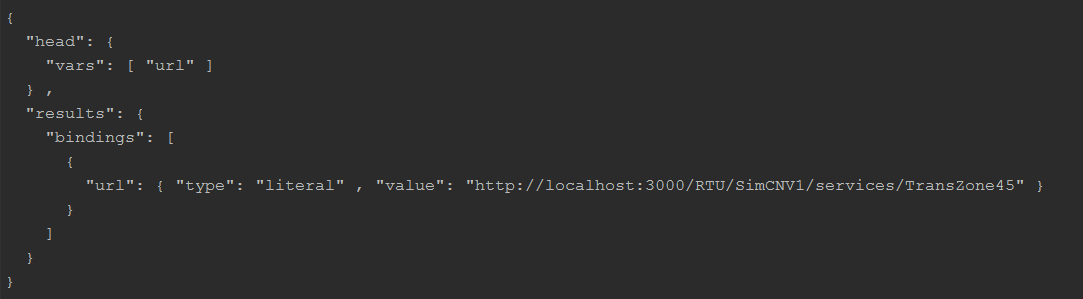
ws9.runServer(6009);

ws10.runServer(6010);

ws11.runServer(6011);

ws12.runServer(6012);

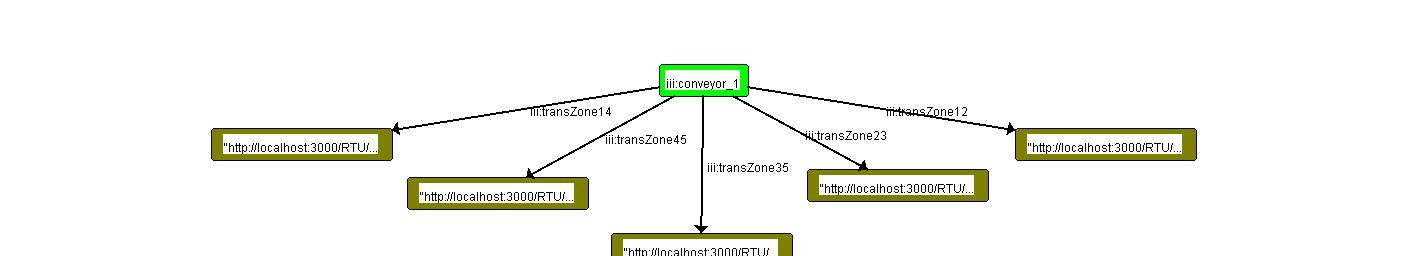
**JSON Response from knowledge base**



## Knowledge Base (class Graphs)

A close up of a device

Description generated with high confidence



A close up of a map

Description generated with high confidence