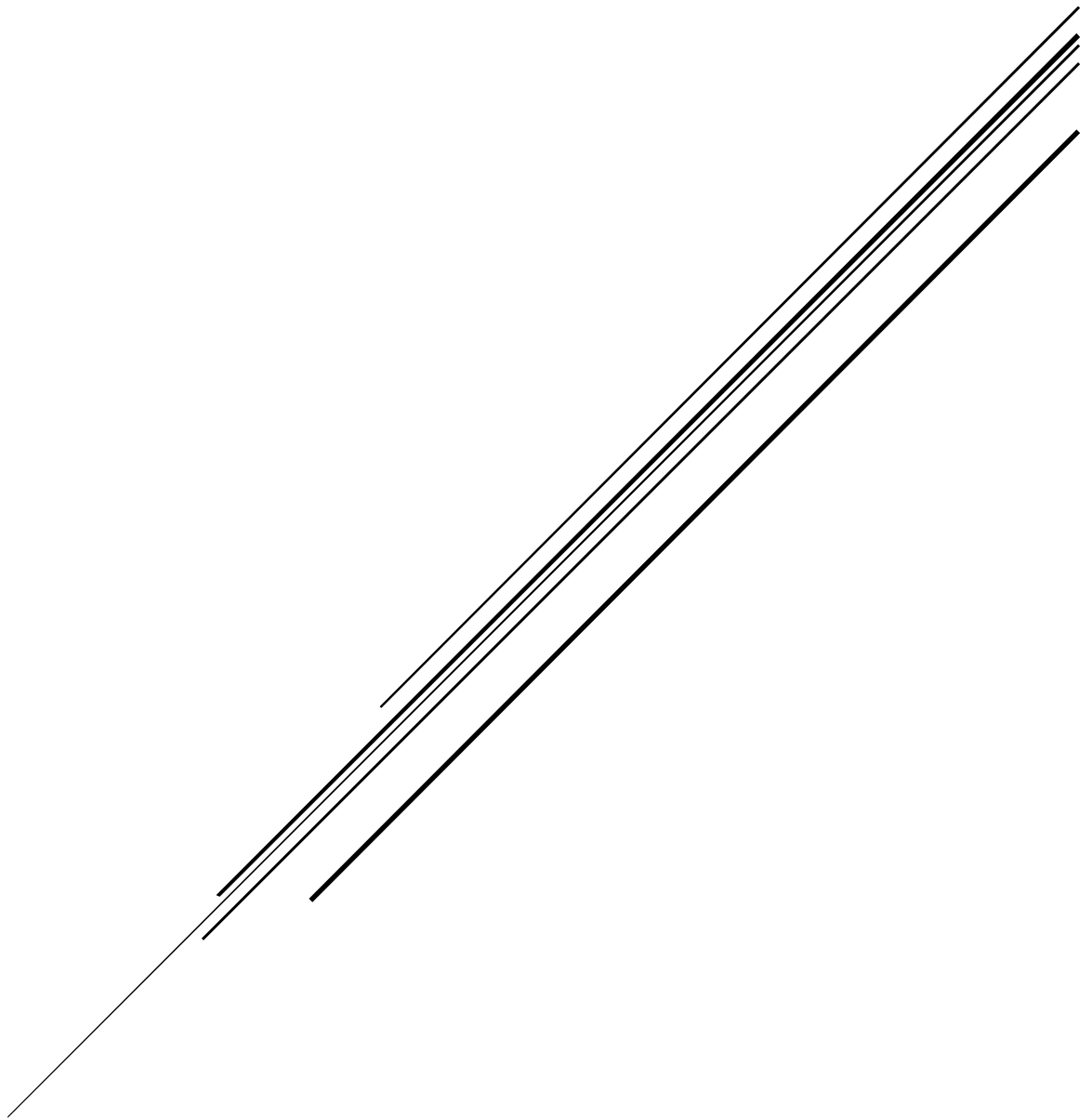




TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ITERATION 6: TRANSITION

ASE-9316: Introduction to Industrial Informatics



Terms of Reference

This Iteration, in addition to the previous 5 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.

Joe David

267598

Prasun Biswas

267948

Iteration Version History

Iteration	Date	UP Phase	Remarks
1	28.09.2017	Inception	<ul style="list-style-type: none"> • System vision • WBS diagrams with the ProjectLibre • System Architecture
2	13.10.2017	Elaboration	<ul style="list-style-type: none"> • Refined System Vision • Cost/Benefit Evaluation • UML Diagrams (Use Case, Class, Activity, Sequence)
3	31.10.2017	Elaboration	<ul style="list-style-type: none"> • Report Restructured • State Diagram added • Refined Class Diagram • Refined Use Cases • Source Code Prototype • Basic GUI Development using SVG
4	15.11.2017	Elaboration/ Construction	<ul style="list-style-type: none"> • 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
5	30.11.2017	Construction	<ul style="list-style-type: none"> • Knowledge Base developed comprehensively • Gateway and Orchestrators components introduced. Newly Developed Classes: Order (sub class: Product), Workstation, Process, SPARQL Generator Function, Another Function to execute commonly executed tasks. • Final Deployment Diagram developed • GUI jQuery Script revised • Functional Progress: Order CL periodically queries the knowledge base for Orders and updates the KB. Order class is simultaneously populated with the orders as products being objects of the Product Class as attributes of the Order Class with each order having the knowledge of the subsequent orders with circular referencing • Source Code Section Removed from

			Report to keep report Concise
6	10.12.2017	Transition	<ul style="list-style-type: none"> • Final Refinement to the Knowledge Base. Developed in completeness. • All Classes developed completely with complete functionalities • UML Diagrams Refined • GUI: Live Product Status Implemented. • System tested to work with complete functionality with multiple product Orders.

Contents

Terms of Reference	2
Iteration Version History	3
1. System vision:	6
2. System Description:	6
3. System Architecture:	7
4. Explanation of the Block Diagram:	7
5. Explanation of the WBS	8
Inception phase:	8
Elaboration Phase:	8
Construction Phase:	9
Transition Phase:	10
6. COST/BENEFIT ANALYSIS	11
7. UML Diagrams	12
Deployment Diagram	12
Class Diagram	12
USE Case Diagram	13
Activity Diagram	14
Sequence Diagram	15
State Diagram	16
8. FRONT-END DEVELOPMENT	18
Features of the UI	19
How to place an Order	19
9. BACK-END DEVELOPMENT	20
KNOWLEDGE BASE:	20
Explanation of Knowledge base:	23
Class Hierarchy:	23
Class Instances and Property Assertions:	23
List of Files and Operation Guidelines	28
List of Files	28
Operation Guidelines.	28

1. 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.

2. System Description:

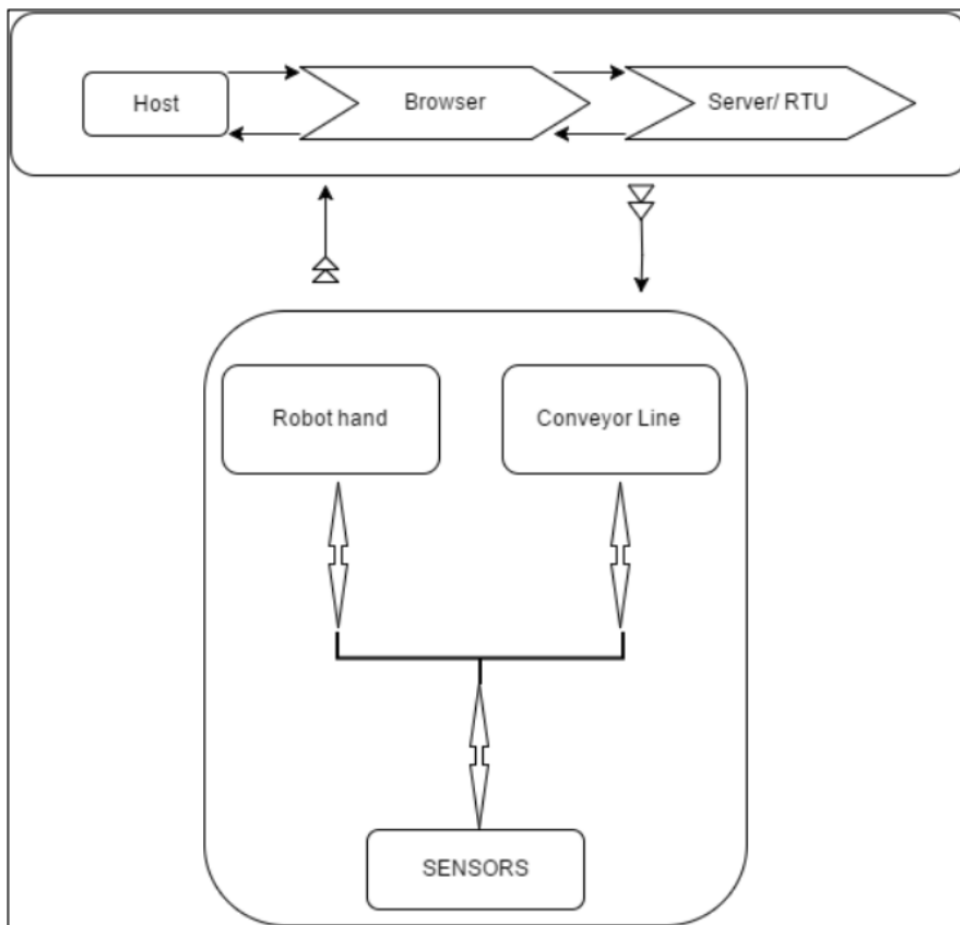
The FASTory line includes the main components of a production line; robots, transportation system, tools, end effectors, raw material, working stations, loading and unloading stations and a buffer station.

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.

3. System Architecture:

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



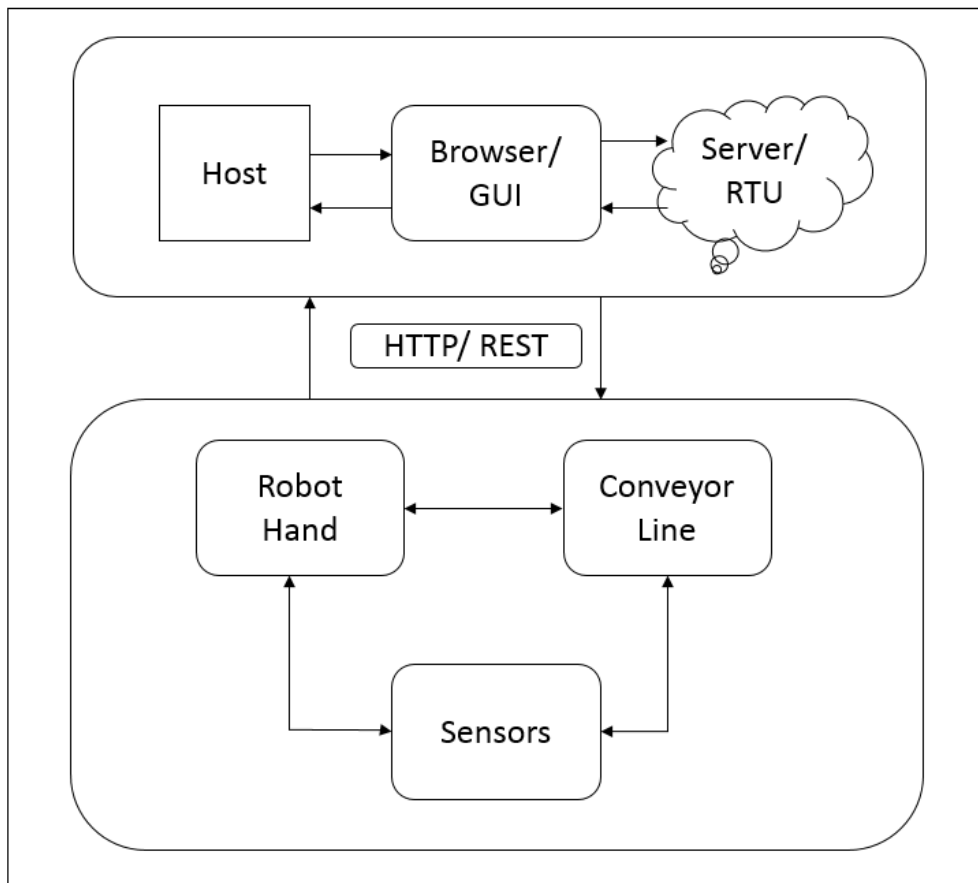


Figure 1: Block Diagram of System Architecture

4. 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.

5. 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 Factory 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 constraints 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.

6. COST/BENEFIT ANALYSIS

Tools/ Components	<p>The development cost has been calculated using WBS in Project Libre. The Cost/ Benefit analysis has been done annually instead of quarterly. The current inflation factor 1.14% (HICP) of Europe has been used to predict the amounts required as maintenance cost (maintained cost will be required more every year due to the inflation.)</p> <p>The benefit analysis has been done using the current growth rate of the cellphone market which is estimated to grow at an average rate of 3.8% between 2017-2021 as per [1].</p> <p>Reference: [1] https://www.idc.com/getdoc.jsp?containerId=US42366217</p>
Fastory Line	
12 manupulators	
Sensors	
Installaion Cost	
Development Cost	
Total Development and Installation Cost	

7. UML Diagrams

Deployment Diagram

Class Diagram

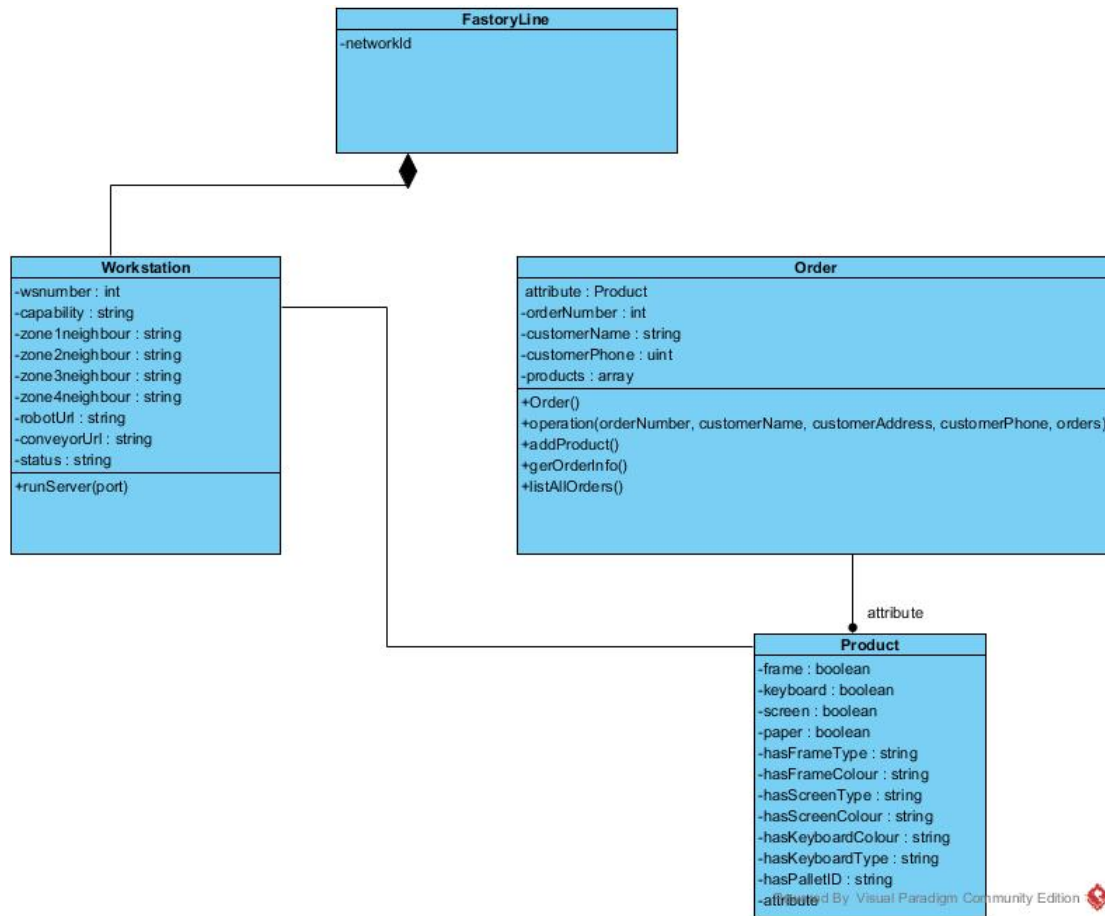


Figure 2: UML Class Diagram

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)

Each workstation is run as individual servers with on the localhost and different port numbers.

Methods:

- `runServer()`- to run the individual Servers of the workstations

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.

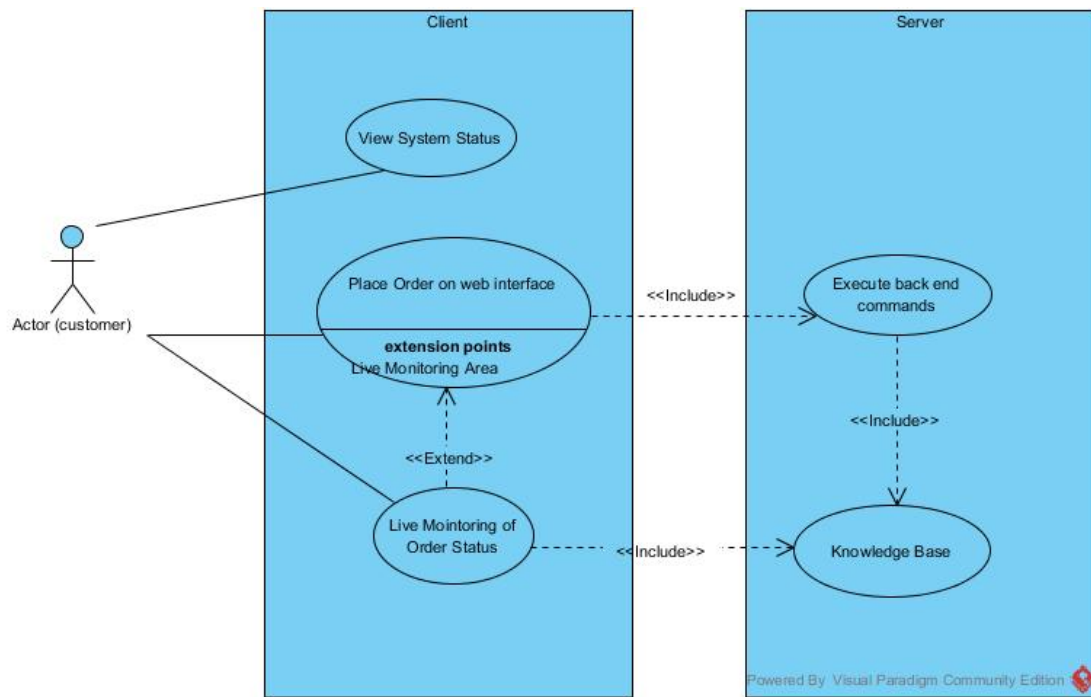


Figure 3: Use Case Diagram

The user is able to both place an order and view the current system status . Back end commands are issued once the Order is placed in the GUI. Additionally the user can get a Live Order Monitoring of the Order status.

Refinement of Use Case diagram:

In this iteration we have omitted the Authentication Use case and plan on using it if time permits

Activity Diagram

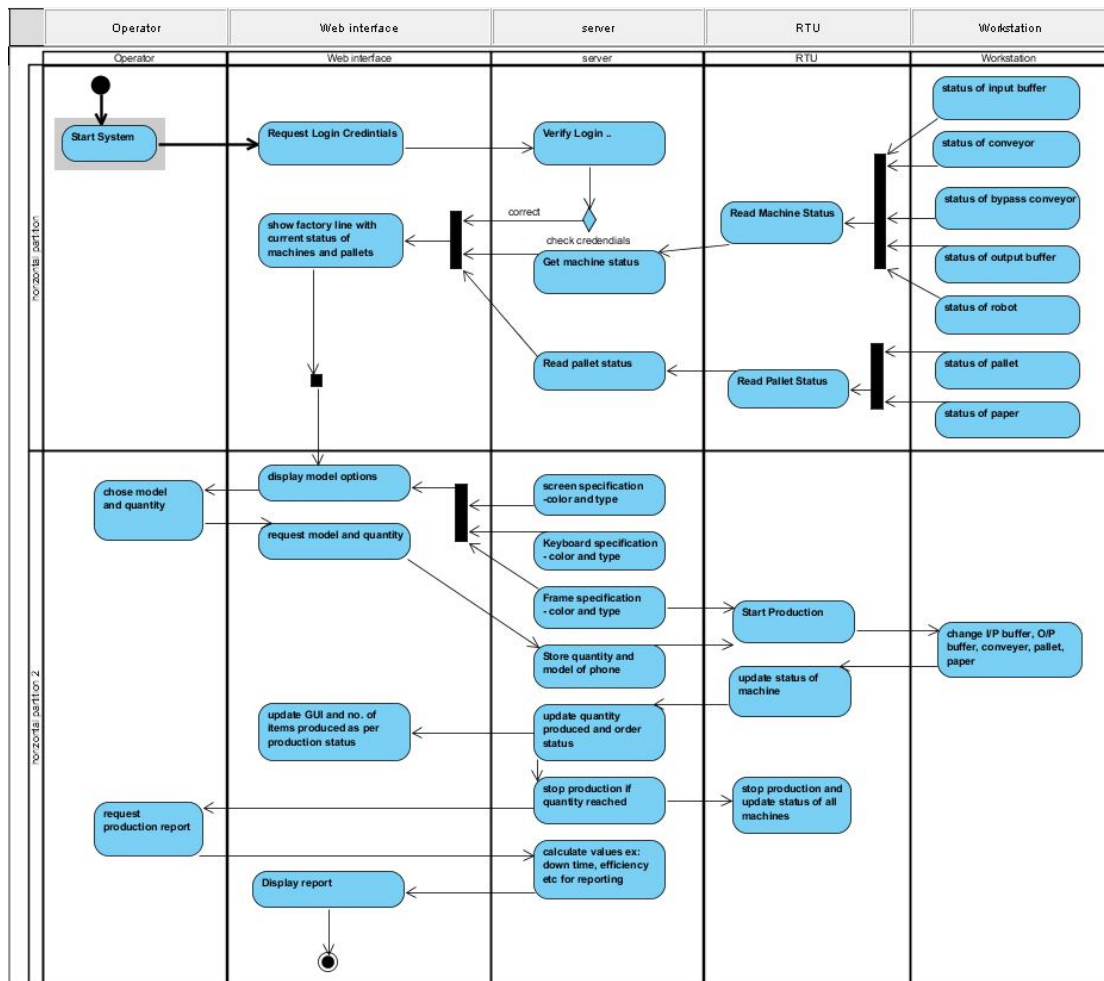


Figure 4: Activity Diagram

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

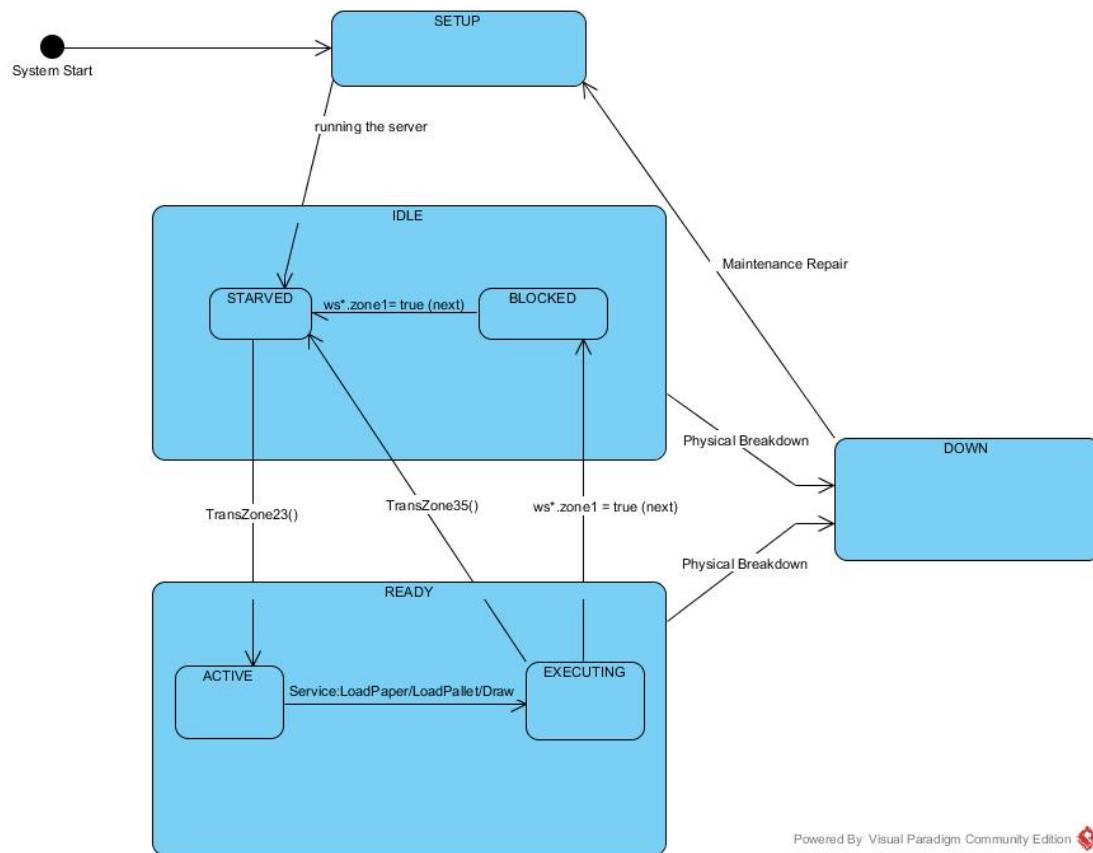


Figure 6: State Diagram

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.
3. 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.
4. 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.
5. 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.
6. In any case, if there is a mechanical failure of any of the equipment in a workstation it goes.

8. FRONT-END DEVELOPMENT

Figure 8 shows a preview of the User Interface.

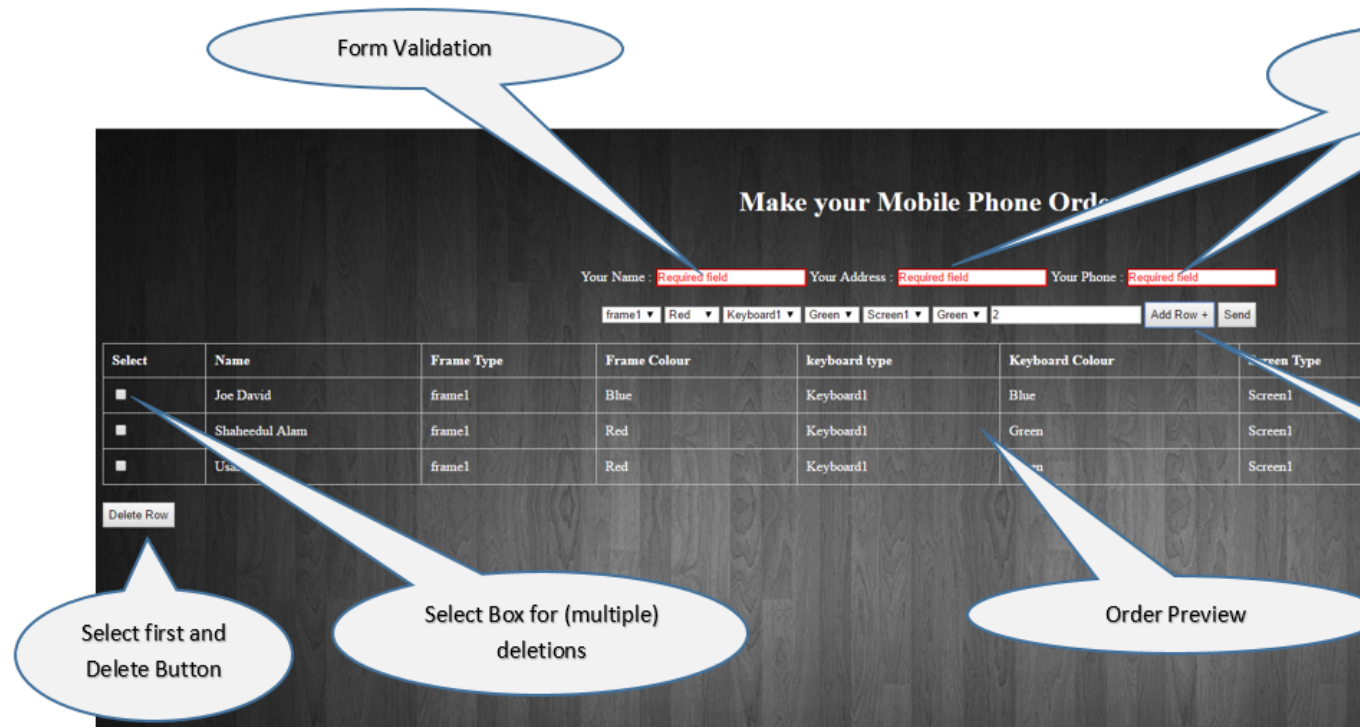


Figure 7: GUI User Interface

Features of the UI

- Uses Form Validation to not allow users to Place Order without entering vital information such as Name, Address and Phone number
- Uses CSS to remind user of the missing information.
- Enables User to enter multiple orders using the Add Order button
- Allows User to delete an order by ticking a check box and pressing the delete row button in case the user inputs an erroneous order.
- Shows a preview of the Order entered so far.

How to place an Order

- Fill User details in the Name, Address and Phone number fields
- Select the Order Variant by the drop down menus
- Add multiple Orders using the Add row Button
- Delete any order if necessary by checking the box adjacent to the order and pressing the “delete order button”
- Press the send button to place the order.

9. BACK-END DEVELOPMENT

KNOWLEDGE BASE:

The following figures shows some of the generated graph from the knowledge base of the class members. It also showing some connections with other members of the classes and its capabilities.

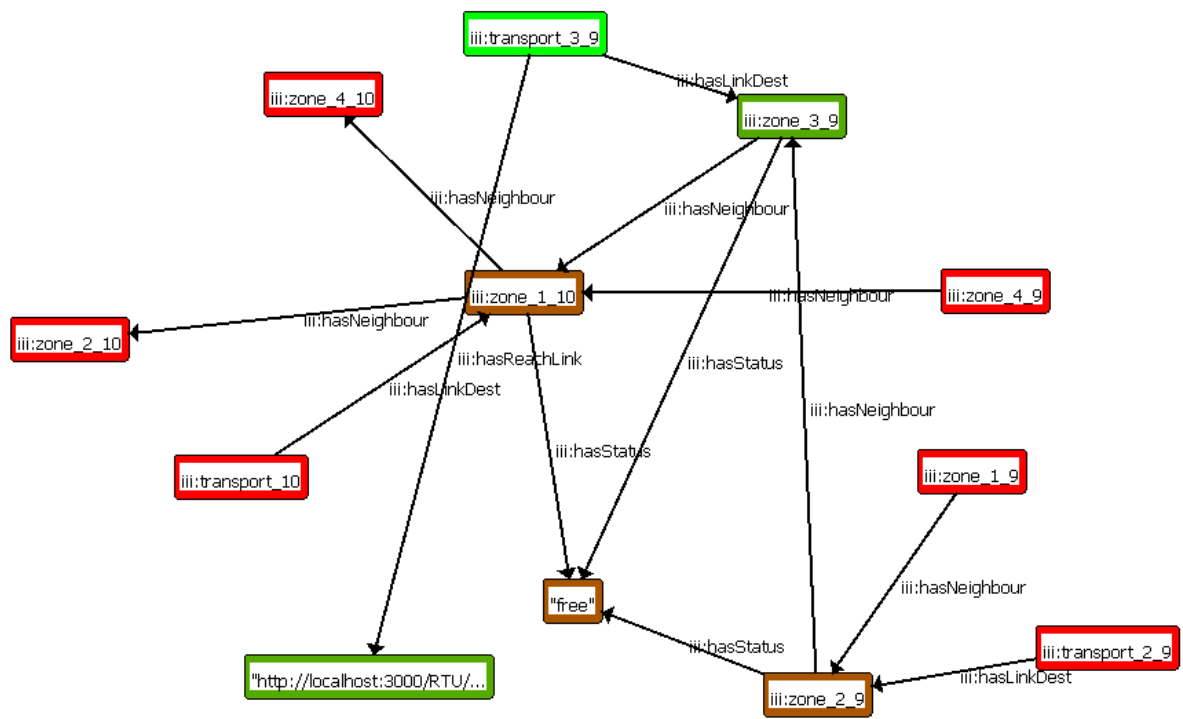


Figure 9: Graph of Transport Class

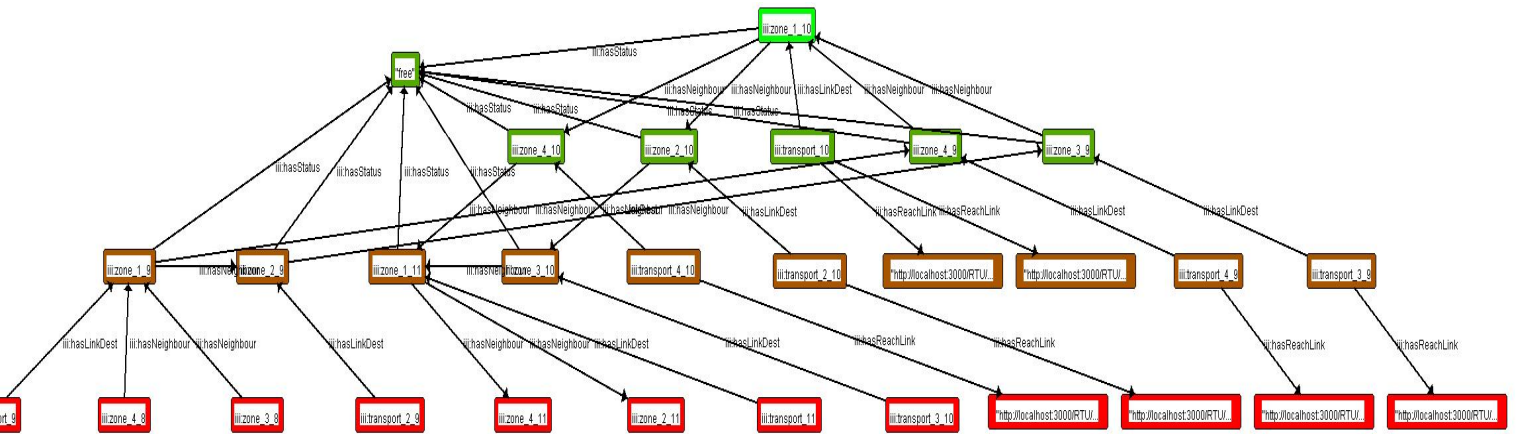


Figure 10: Graph of Zones with `hasLinkDest` and `hasReachLink` Properties

Explanation of Knowledge base:

Class Hierarchy:

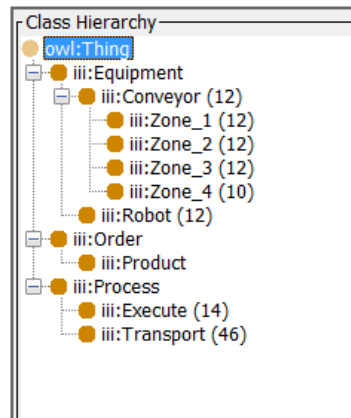


Figure 11: Class Hierarchy

The structure of the Database is such that there are three Main classes (subclasses of Thing) namely Equipment, Order and Process.

Equipment have subclasses Conveyor and Robot. Order has subclass Product(Order is seen as a collection of Products) and Process has subclasses Execute and Transport. Conveyor further has subclasses which are its zones.

Class Instances and Property Assertions:

Conveyor

Zone_1 Instances are as many as the number of zone 1's in the system and so on. "zone_1_10" is read as Zone 1 of workstation 10. Each zone has a transitive property "hasNeighbour" that has both domain and ranges as Zones. **This is the fundamental property of the system that enables it to have a dynamic character with regards to the transport of pallets in the system.** In the case shown in the Figure zone _1_1 has two neighbours zone_2_2 and zone_4_2. The Dynamic Behaviour is explained [in this section](#).

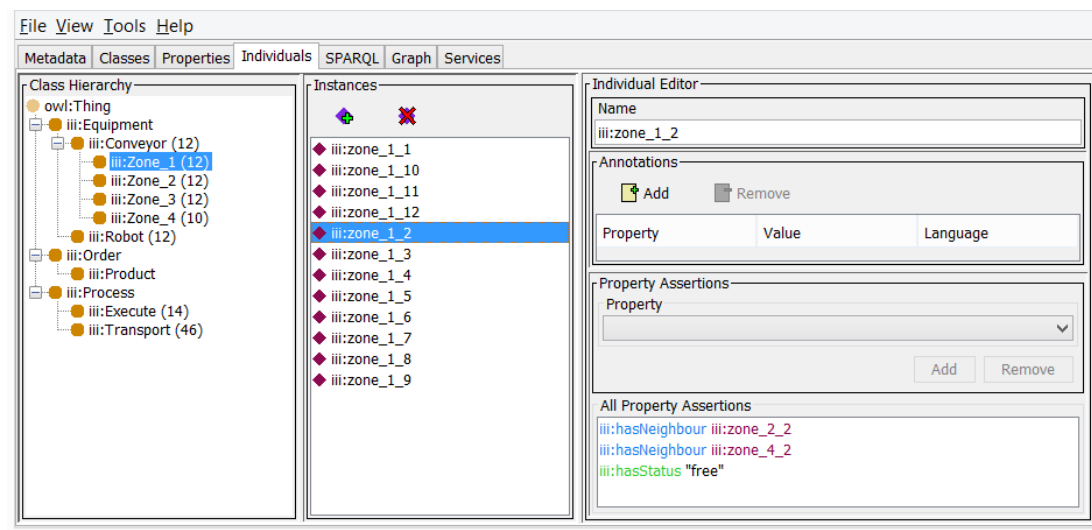


Figure 12: Property Assertions of Class Conveyor

Robot

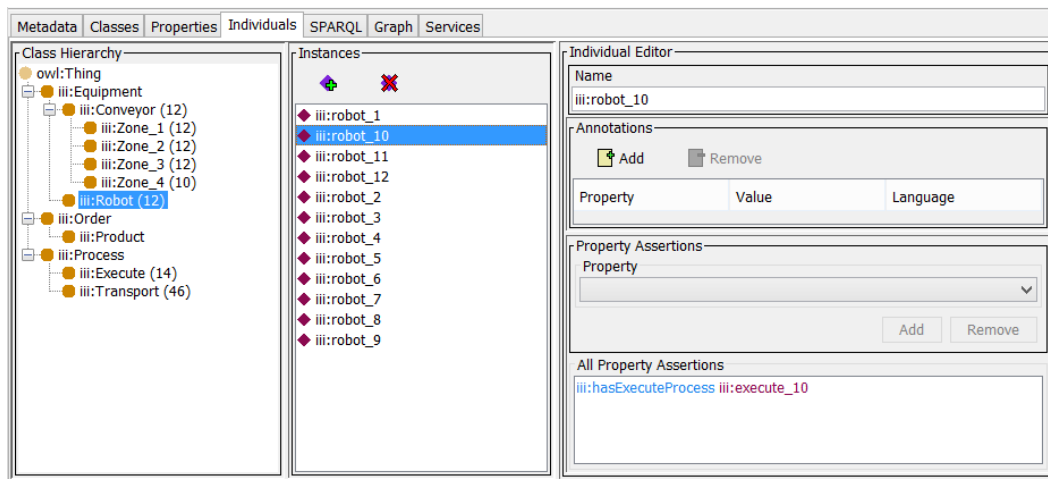


Figure 13: Property Assertions of Class ConveyorRobot

Robot have a property “hasExecuteProcess” that have the domain as “Robot” and range as “Process>Execute”. The intuition is that Robot have the ability that enable them to “execute” certain “Process”. What these processes are will be covered shortly.

Product

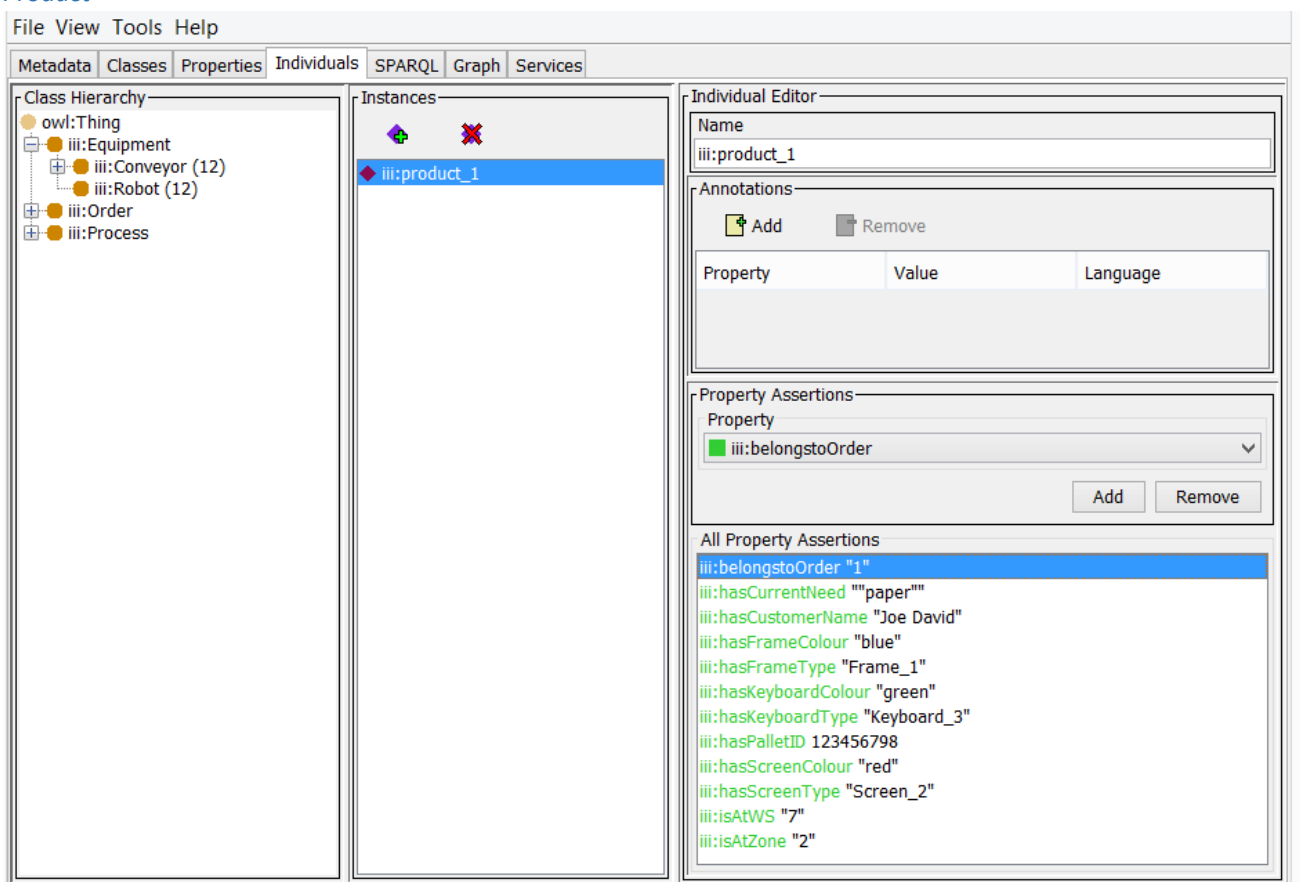


Figure 14: Property Assertions of Class Product

The properties of the Products are shown in the above figure and are self-explanatory. Please note that this is populated dynamically during run time by the Order Class and the above picture is shown just for illustrative purposes.

Execute

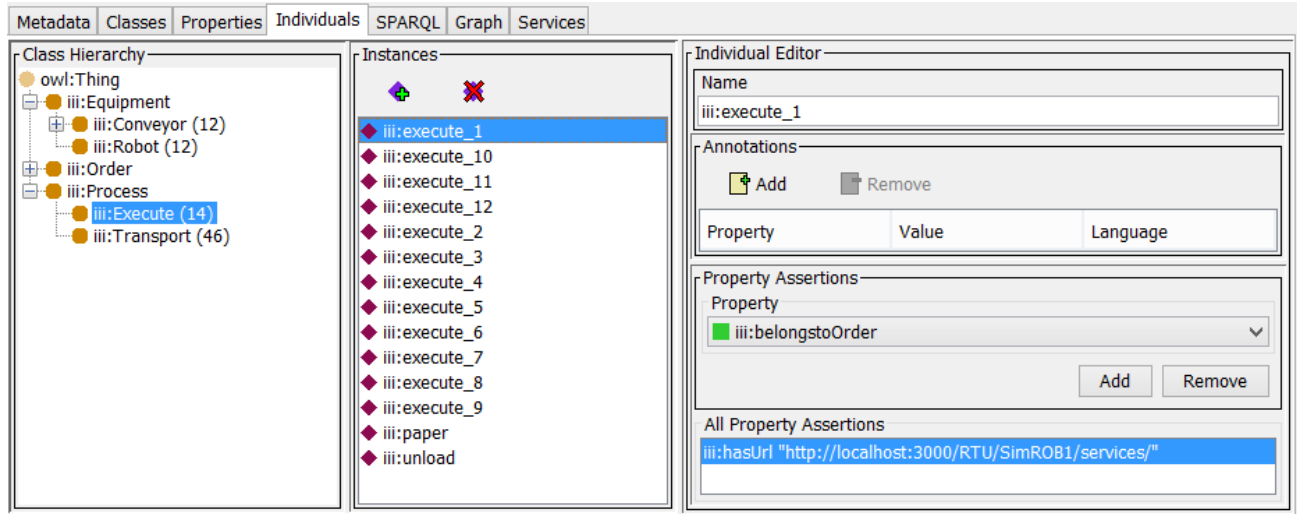


Figure 15: Property Assertions of Class Execute

The execute class contains service execution URLs for each Robot. The Robot first checks its execution capabilities by its “hasExecutionProcess” property which in turn returns the service execution URLs to the Robot. This is how the robot executes its “Process” operations on the pallets. The basic URL is retrieved from the hasUrl property and is merged with the requirement of the Pallet also retrieved from the Knowledge Base from the Product class during runtime.

Transport Class and the Dynamic Transport Mechanism.

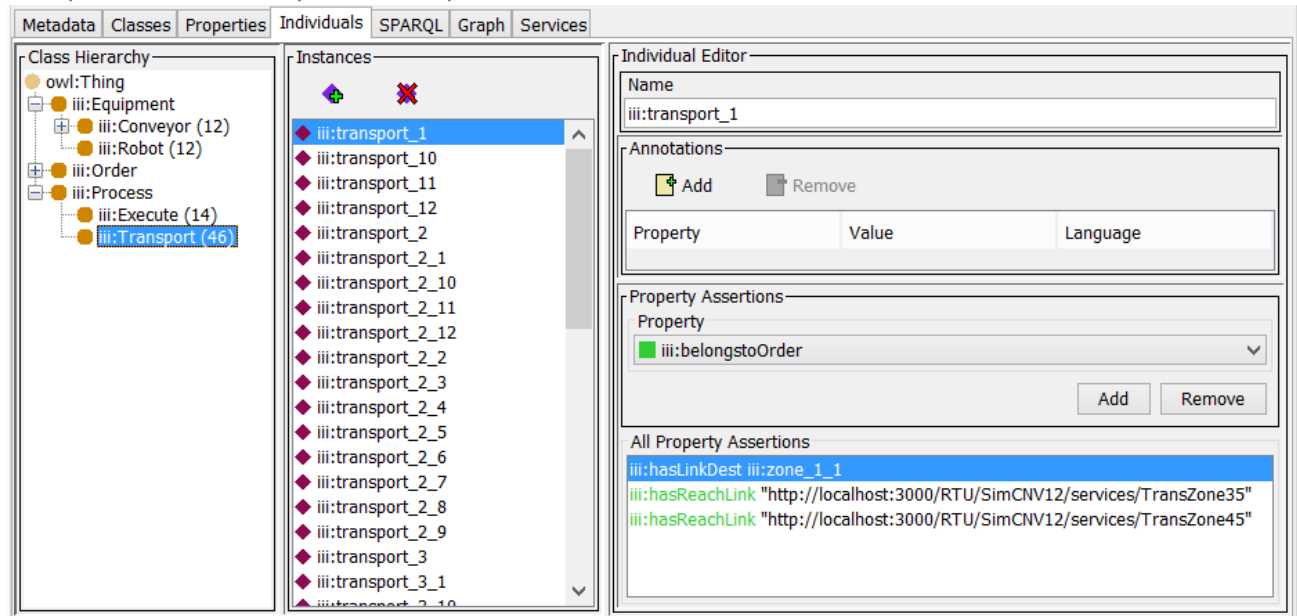


Figure 16: Property Assertions of Class Transport

The Transport class has instances that can take any instances names. In this why we introduce the dynamic capability of transport of pallets/products within the system. Here the names are just labeled for sanity purposes and for easy debugging during the project development phase. Each transport instance, has atleast two properties, hasLinkDest which has an arbitrary zone number and the Urls that can be used to reach it. To understand better, we can consider the case as in the above figure. “zone_1_1” read as zone 1 of workstation one has to Links that can be used to reach it as shown. One link is from the zone 3 of the previous workstation and the other being from zone 4 of the previous workstation, which in this case is Workstation 12.

The Dynamic Transport Mechanism, when each Pallet reaches zone 1 of any given workstation, it first checks if the workstation is busy or not.

In the case that it is not, it pallet is assigned a temporary goal to reach the zone 3 of the workstation. In the case that it is busy, the pallet is assigned a temporary goal to reach the zone 1 of the next workstation. Now the Pallet checks for the Neighbours of the current zone to see if it's the temporary goal assigned, using the "hasNeighbour" property of the zones. This would typically return zones 2 and 4 of the workstation. Next as the goal is not found, the workstation further checks their neighbours (neighbour's neighbour). At this level, the temp goal for the pallet is found and the workstation queries the Knowledge Base for the link that can be used to reach the neighbor whose neighbour is the temporary goal using the "hasReachLink" property of the .

This is not just the case with zone 1. In every zone, the workstation first queries the KB for its neighbouring zone and then queries the Knowledge base for links to reach those zones. The way this is done is that the workstation queries "hasReachLink" Property of a zone that has the "hasLinkDest" of the destination that it wants to reach. **In this way irrespective of the name of the Transport Instance the link for it is always returned correctly. Also if there were a robot that manually picks the pallet and puts it in any random place in the system it would carry on from there by checking its immediate neighbours and so on and so forth.**

List of Files and Operation Guidelines

List of Files

User Interface

1. app.js – The Javascript file that Hosts the User Interface.
2. Form1.html – The html page for the User Interface
3. Script.js – the script that runs on the User Interface that routes the entered order information appropriately

Classes and Other Components

1. Gateway.js – The Gateway for the system. Has the function of updating the Knowledge base with the Order.
2. Orchestrator.js – The Javascript file responsible for invoking services on the Fastory.
3. Order.js – The Order Class the contains the Order and periodically checks for new Orders. Contains an array of Products of the Product Class as one of its attribute
4. Product.js – Product Class that stores Product Information.
5. Process.js The Process Class that has processes of draw and execu
6. Workstation.js – The Workstation Class the subscribes to events on the Fastory and invokes services by requesting the Orchestrator.

Functions

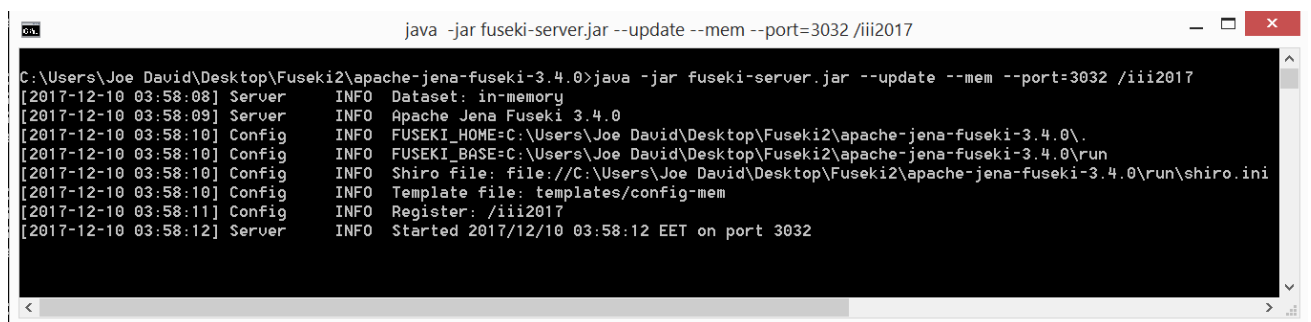
1. SPARQLGEN.js – Contains s that generates SPARQL queries for use by the classes to query the fuseki server.
2. Functions.js – Contains other functions required for working of the system.

Ontology

1. Ontology.owl

Operation Guidelines.

1. Start the Fuseki Server by navigating to the directory in command prompt and using the command “java -jar fuseki-server.jar --update --mem --port=3032 /iii2017” to start on port 3032 on the local host with the dataset iii2017.



```
java -jar fuseki-server.jar --update --mem --port=3032 /iii2017

C:\Users\Joe David\Desktop\Fuseki2\apache-jena-fuseki-3.4.0>java -jar fuseki-server.jar --update --mem --port=3032 /iii2017
[2017-12-10 03:58:08] Server      INFO  Dataset: in-memory
[2017-12-10 03:58:09] Server      INFO  Apache Jena Fuseki 3.4.0
[2017-12-10 03:58:10] Config      INFO  FUSEKI_HOME=C:\Users\Joe David\Desktop\Fuseki2\apache-jena-fuseki-3.4.0\
[2017-12-10 03:58:10] Config      INFO  FUSEKI_BASE=C:\Users\Joe David\Desktop\Fuseki2\apache-jena-fuseki-3.4.0\run
[2017-12-10 03:58:10] Config      INFO  Shiro file: file:///C:\Users\Joe David\Desktop\Fuseki2\apache-jena-fuseki-3.4.0\run\shiro.ini
[2017-12-10 03:58:10] Config      INFO  Template file: templates/config-mem
[2017-12-10 03:58:11] Config      INFO  Register: /iii2017
[2017-12-10 03:58:12] Server      INFO  Started 2017/12/10 03:58:12 EET on port 3032
```

2. Upload the Ontology to the fuseki server by clicking on the “add data button”

Apache Jena Fuseki

Version 3.4.0. Uptime: 0m 47s

Server status: ●

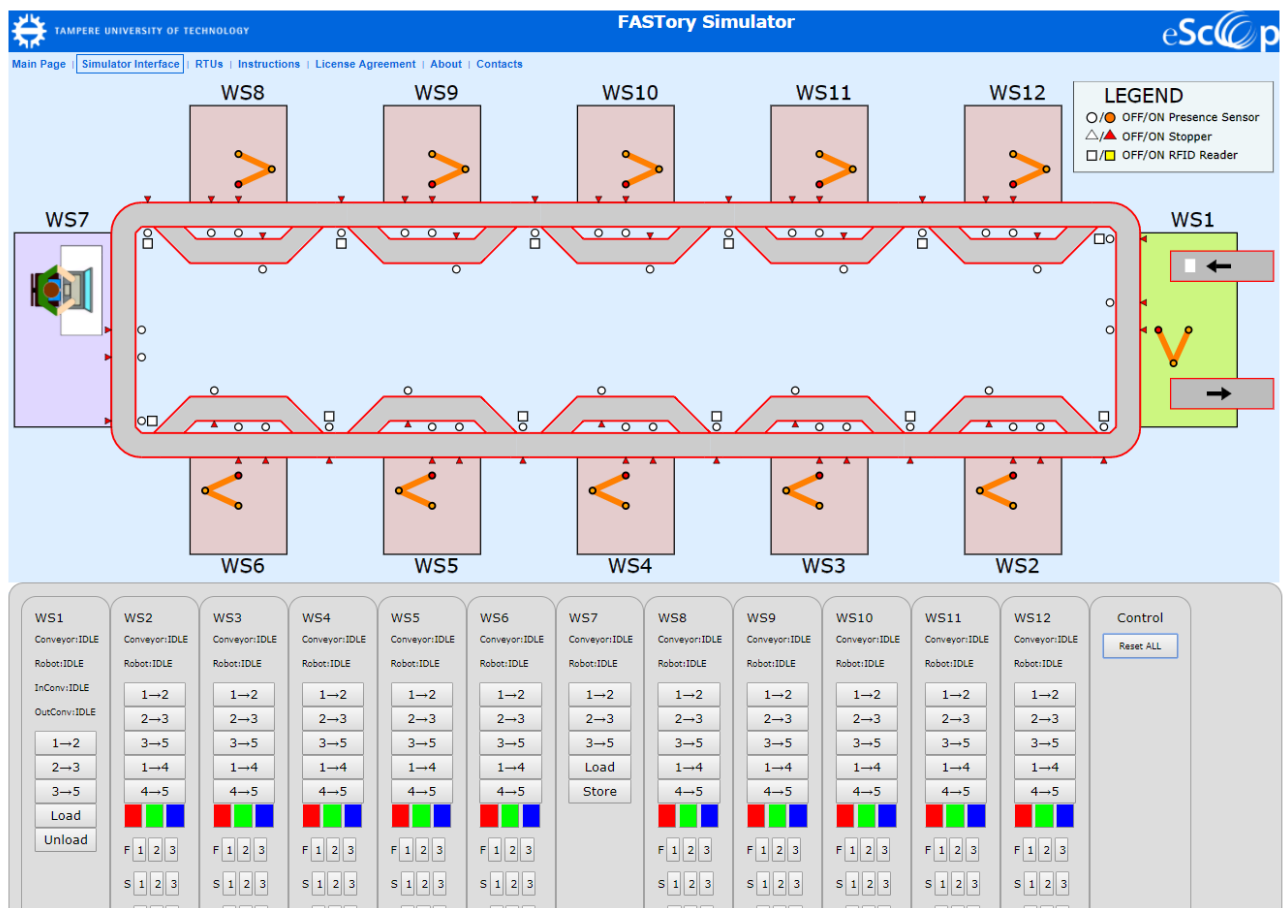
Datasets on this server

dataset name	actions
/iii2017	query add data info

Use the following pages to perform actions or tasks on this server:

- [Dataset](#): Run queries and modify datasets hosted by this server.
- [Manage datasets](#): Administer the datasets on this server, including adding datasets, uploading data and performing backups.
- [Help](#): Summary of commands and links to online documentation.

- Start the Fastory Simulator and navigate to <http://localhost:3000/fmw> to see the Fastory Visualization.



- Run all classes listed above by navigating to the directory and using the node <filename>.js command
- Navigate to the <http://localhost:5000> to place order on the User Interface. Details on how to place order can be found [here](#)