Norwegian University of Science and Technology
Knowledge Based Engineering
Knowledge Fusion-Siemens NX
Group: Marco Antonio Merola and Álvaro Ortega

Web-based integration of KBE system

Trucks

Trondheim, November 2022

Parameterization

For this project, we decided to reduce the number of parameters that our DFAs files took to build the models. Furthermore, this approach will ease the creation of the OWL file and the interaction with the saved content so that it will be very easy to get the information about the system through queries. For this solution, we decided to take a parameter that describes the type of truck the user wants to generate, so that by using the same web and the same parameters, it is possible to generate different types of trucks by only changing this mentioned parameter. In broad terms, everything is parameterized, so that the whole structure and details of the selected truck will be changed automatically, which means that all the kinds of trucks receive the same parameters, as for example: element color, details color, truck width, element length, element height, among others. Other necessary calculations to generate the trucks are based on these few parameters.

The end user can use our solution by selecting the parameters on the Web and submitting the result. Furthermore, a couple of instances already exist as objects in the OWL file, therefore through the integration of Fuzeki and OWL, the Truck server will evaluate if the truck already exists by executing a query and checking the received parameters in order to tell the user about it. In other cases, a new truck will be generated and the DFAs files will be updated with the new parameters so that the new model can be created .

Description of the Solution

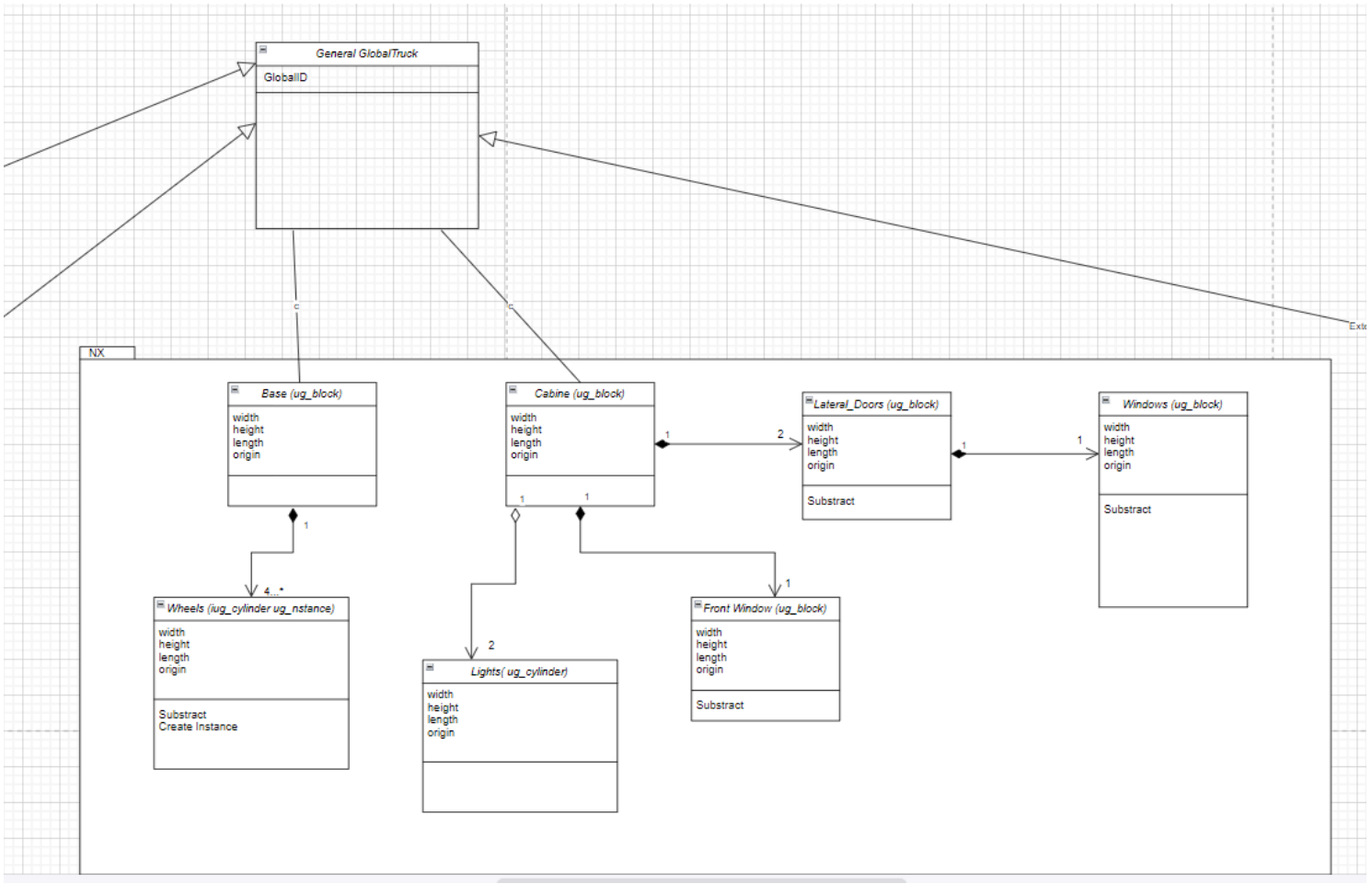
In our solution, the parameters are taken from the Web. We used a very important parameter that describes the type of truck the user wants to generate. This attribute is taken in the Truck Server to check its value and decide which of the DFAs files open. After that, the solution will be generated and written in the corresponding DFAs files that will generate the NX model using the rest of parameters that are sent from the web to the Server. On the other hand, we implemented an OWL database to store existing trucks. In fact, our database corresponds to the solutions that are generated through the DFAs files, which means we used a single class that stores the truck object and saving the type of truck as an attribute of each object, however we also tried to implement this part using three classes, one for each type of truck, but we considered more appropriate to only use a truck class. We connected this database with the Fuzeki server and decided that the connection between the web and the server with Fuzeki and OWL will be done by using constraints to generate trucks. In other words, we generate a request in the form of a SPARQL query from the Truck Server. This query returns the value of the parameters for the existing trucks to check if the new parameters received from the Web would generate an existing truck. In case it already exists the new truck will not be generated, in case it is a new truck it

will be generated by executing an insert SPARQL query.

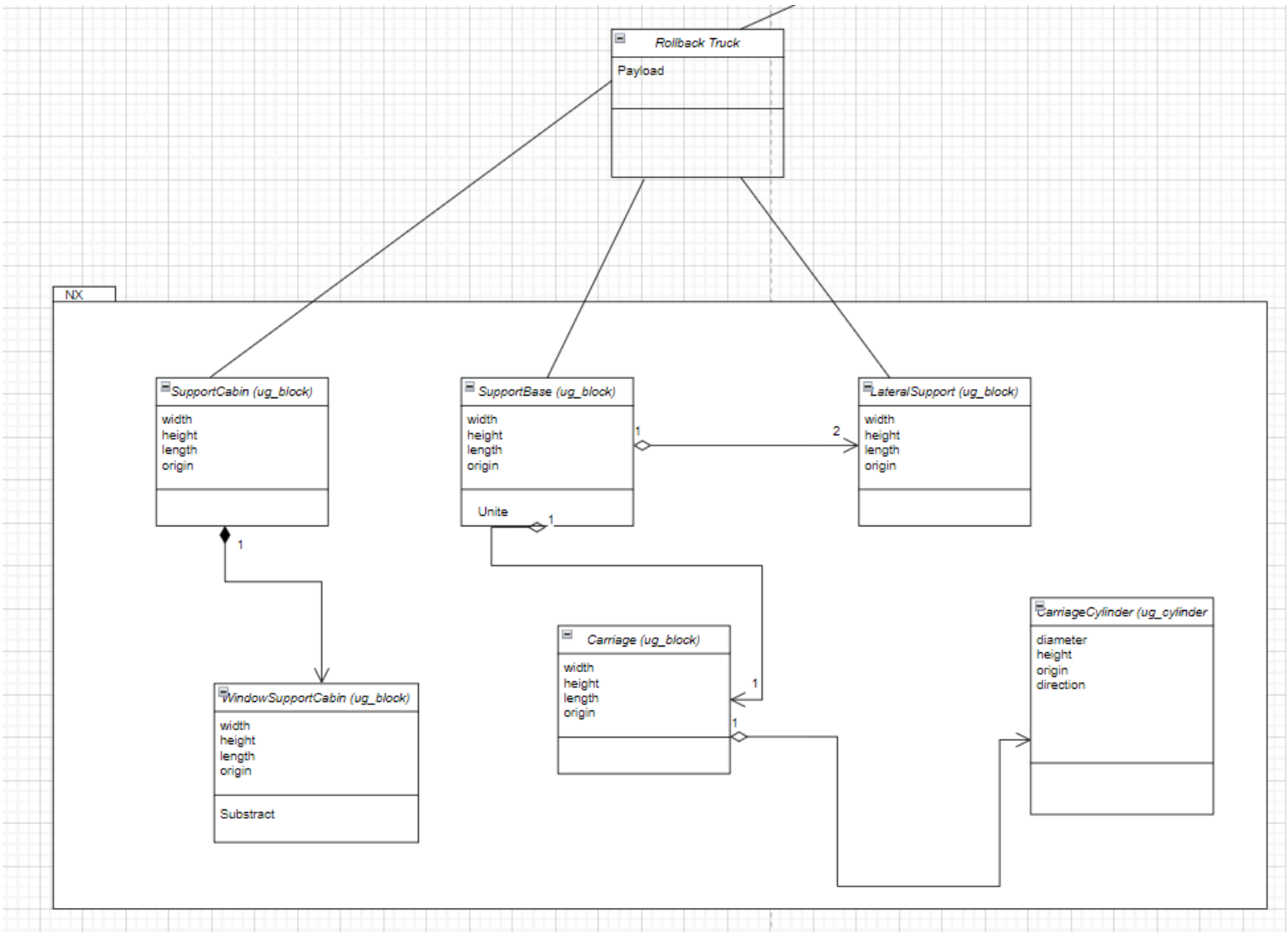
UML Diagrams

Detailed UML Diagram- From Assignment 1

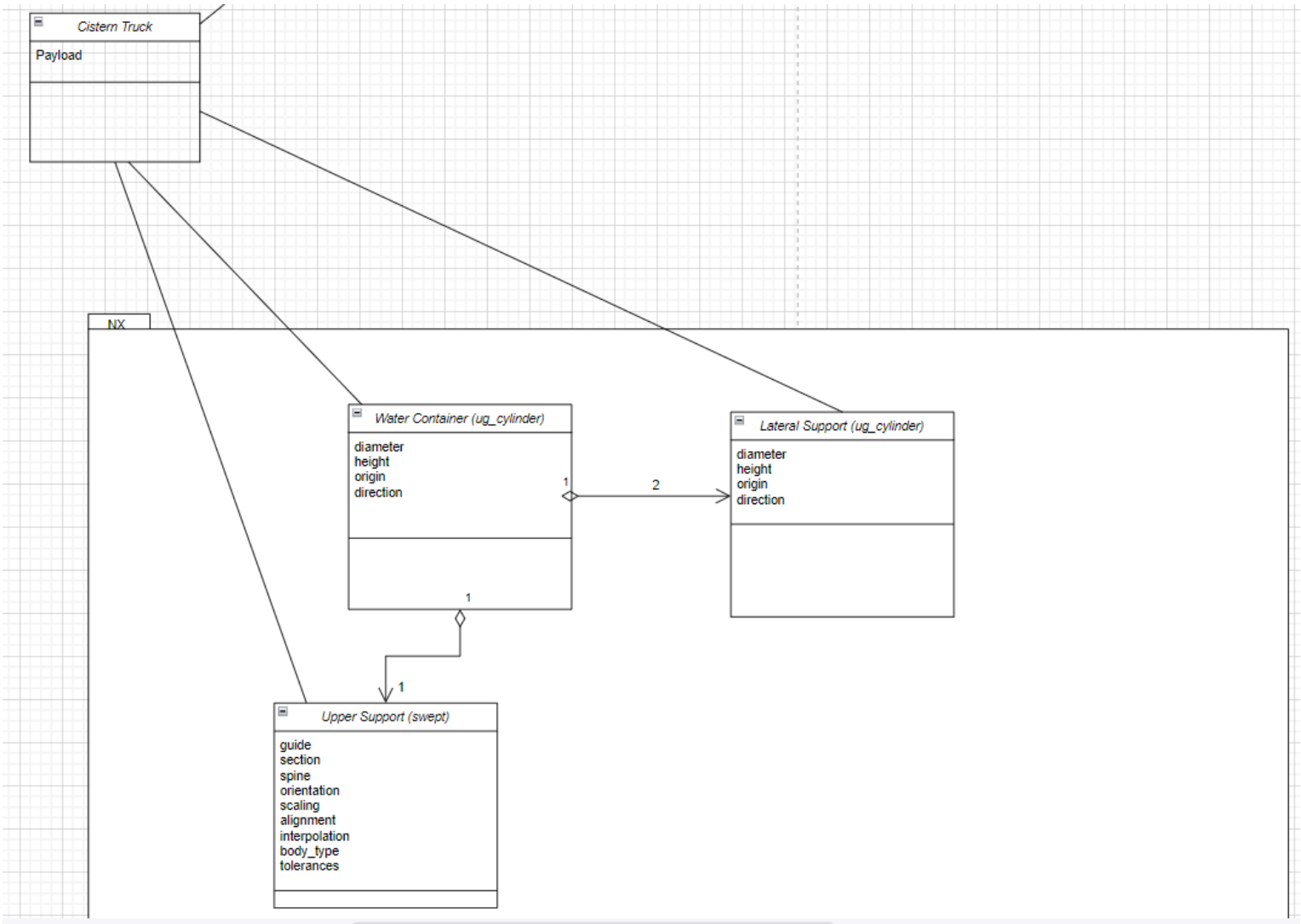
Global Truck Part



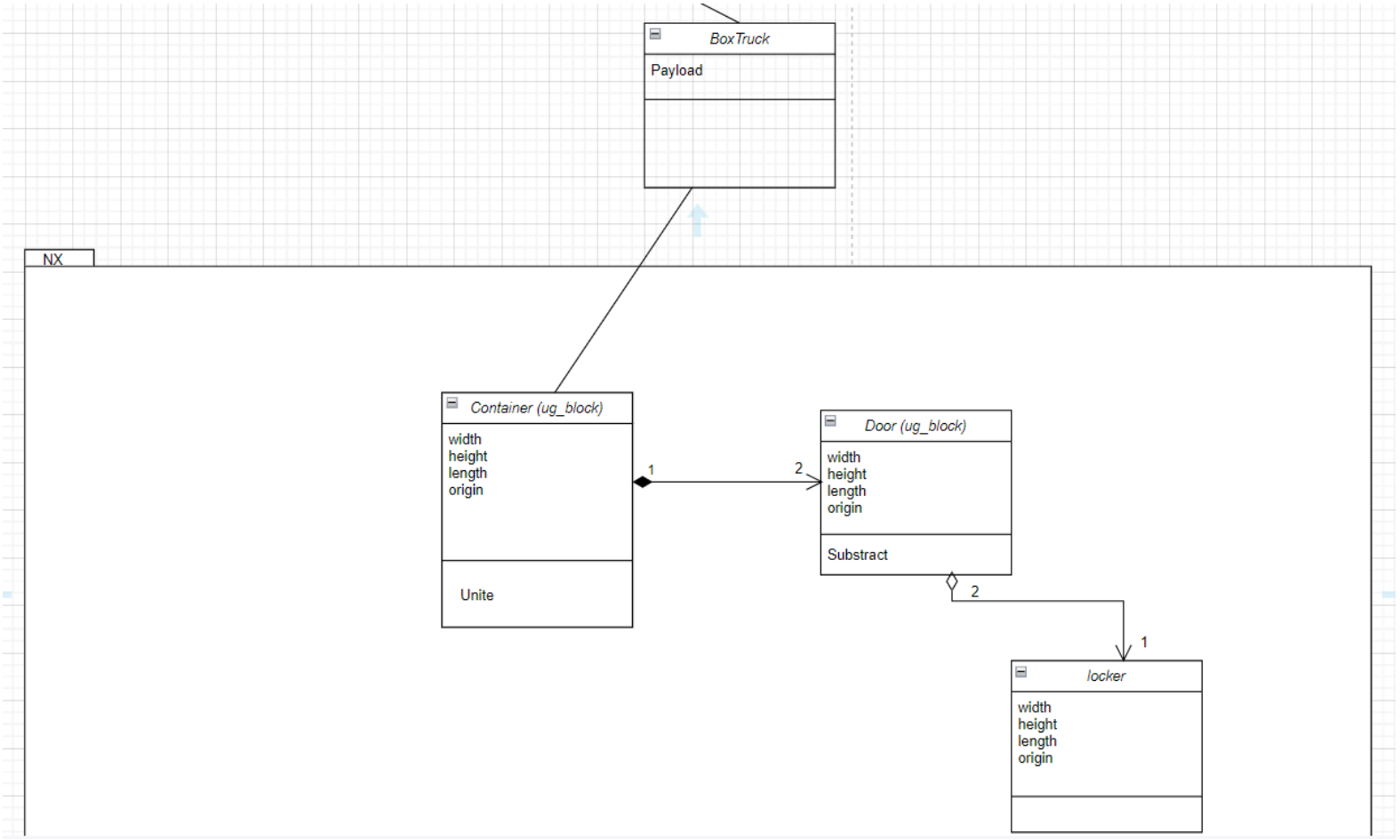
RollBack Truck (extends from Global Truck)



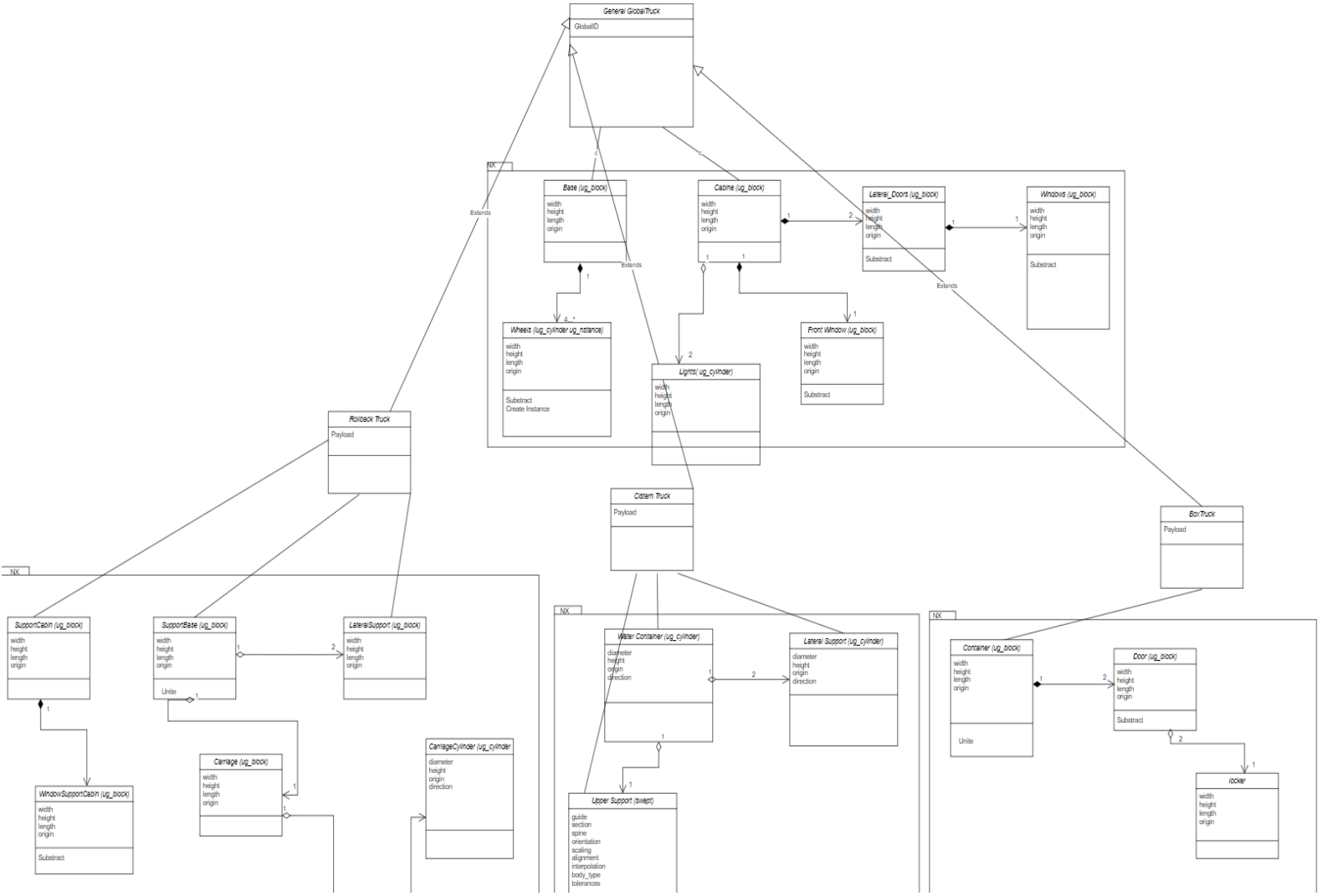
Cistern Truck (extends from Global Truck)



Box Truck (extends from Global Truck)

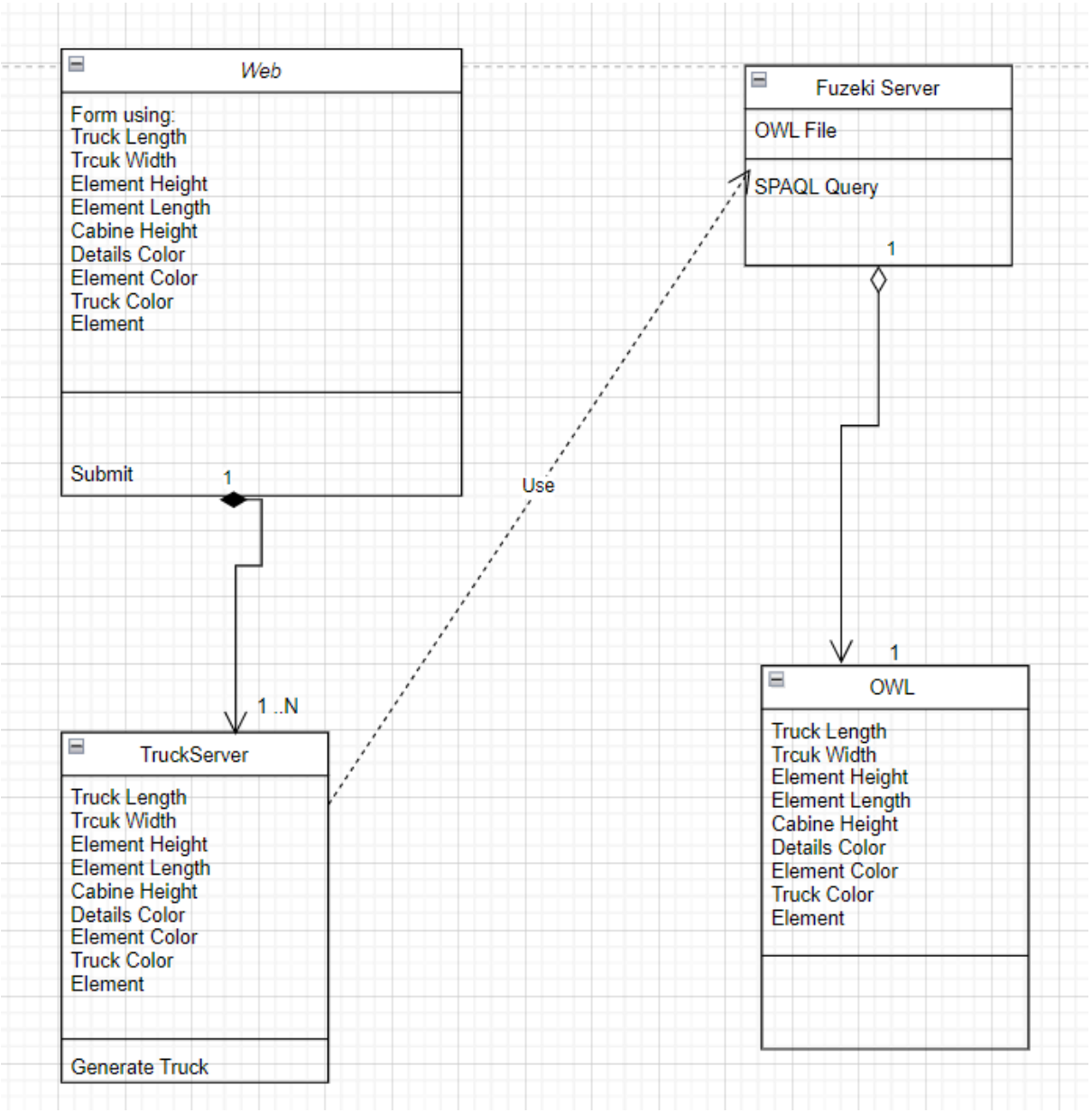


Global Final Structure



UML Diagram With the New Components

To generate an UML diagram representing the structure of the system, we used composition to express that the generated Trucks and the Server cannot exist without the Web that passes the parameters. On the other hand, we have an OWL file that can exist independently of Fuzeki, which is why this relation is an aggregation from one to one. To express the relationship between the Left Structures and the Right Structures in the diagram we decided to represent it as a dependency that exists between the Truck Server and The Fuseki Server, as the first one takes information from Fuzeki to operate correctly

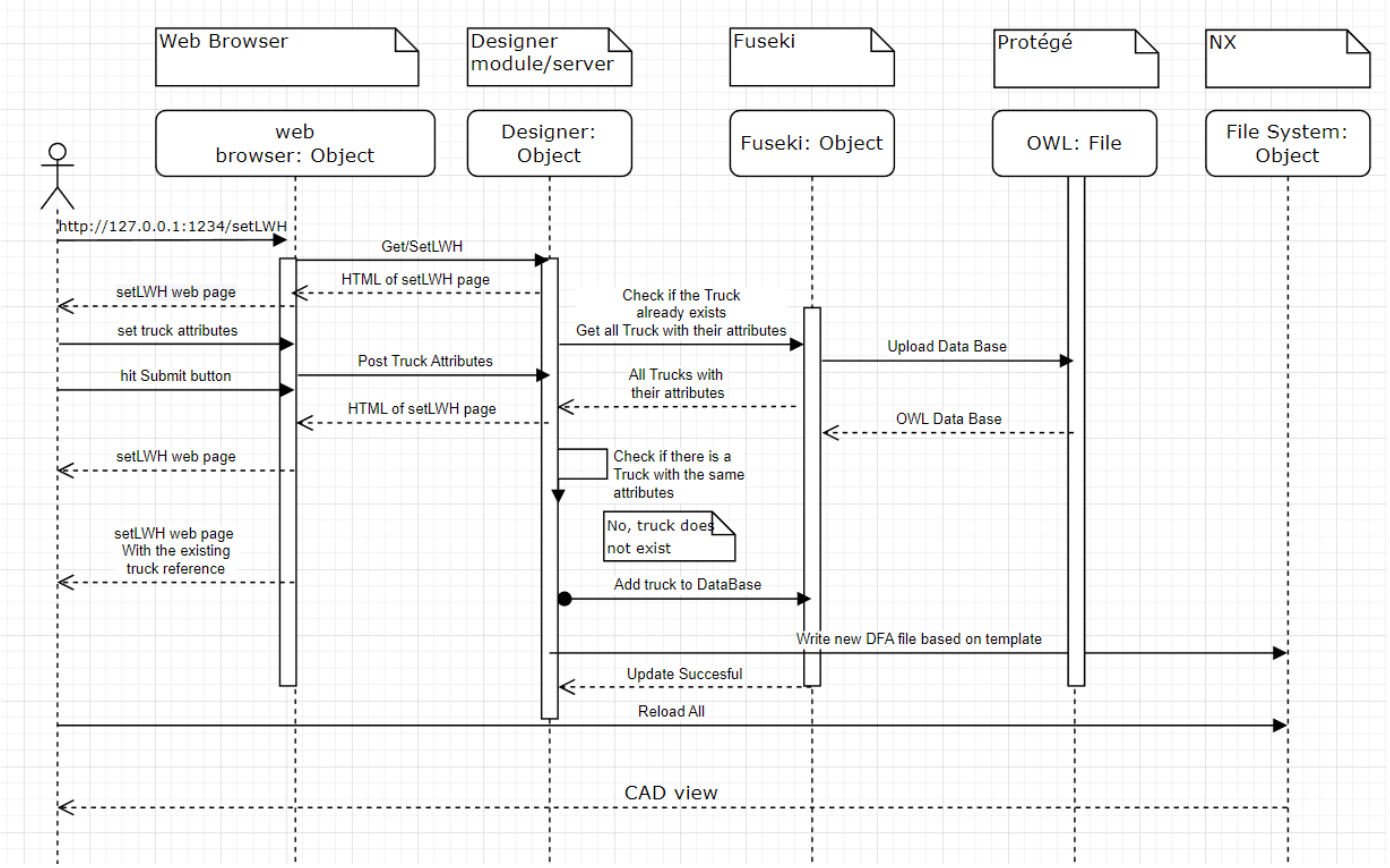


Sequence Diagrams

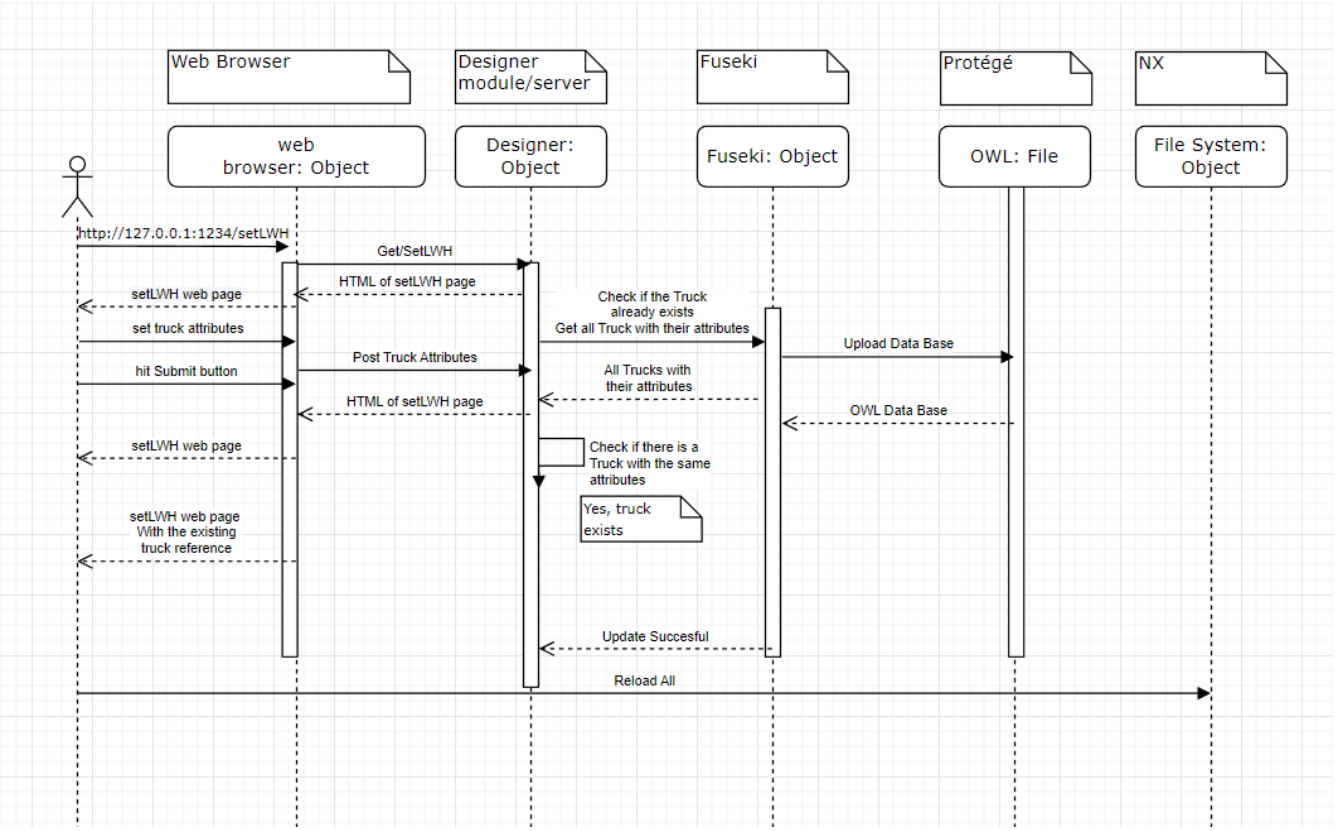
In case the Truck Does not Exist

We have five main components. In broad terms, the user sets the parameters of the desired truck in the website. The website interacts with the Designer Module through get and post methods that reads the parameters from the web in the Designer Module. The Fuzeki server interacts with the OWL file loading the Database to be executed in the server so that already existing trucks are shown. At the same time the Designer Module checks if the new parameters introduced by the user generate a new truck or if the truck is already in the Database. In case the truck does not exist, a new NX will be generated with the solution of the truck and the truck will be stored in the Database

In case the truck does not exist in the Database



In case the truck does not exist in the Database



Reflection and Main Challenges

One of the main challenges in this solution was that we had to take many hours to redefine our truck solution in order to make everything easier in future phases. In fact, we modified many parts of the DFAs files to set rules that calculate details of the trucks and other structures based on very few parameters so that the user felt comfortable and understood the parameters that were being passed from the web to the Truck Server. However, this approach simplified a lot of future implementations, for example with the parameterized new solution, the creation of the database was very easy, because otherwise we would have to calculate each specific part of the truck to make sure it was not stored before.

Furthermore, the SPARQL queries were a bit weird for us and the Fuzeki server caused us problems to open at the beginning. On the other hand, at the beginning we found it hard to understand the relationship between the OWL and Fuzeki modules with the Truck Server, however after understanding the function of the insert queries we completed the functionality of the Knowledge Based System.

In addition, having too many interfaces was complicated to manage, so this assignment required a very good organization among the files, directories, servers and websites. The update function gave us problems, however we found our error to correctly check the dataset and generate the new truck. We tried to display the image of the new generated truck in the website, and we tried to include this part of the code in our solution, however we got many issues trying to display the image and integrate it to our generated solution.

Saving PRT file from KF

```
#! NX/KF 4.0
DefClass: Colored_Block_2022_DYN (ug_base_part);
   (number parameter) my_height: 44;
   (number parameter) my_width: 33;
   (number parameter) my_length: 22;
         (name parameter) color: CYAN;
         (integer parameter) saveOpResult:
save:("C:\Users\andreilo\Desktop\Temp\DFAs\dynBlockTest6.prt");

(child) block1:
   {
      class, ug_block;
      length, my_length:;
      width, my_width:;
      height, my_height:;
   };

# Body colored depending on the volume of the block
(Child) body_colored:
{
  Class, ug_body;
  Feature, {block1:};
  Layer, 1;
  color, ug_askClosestColor(color:);
};

# Methods of the class
(Method Number) getVolume:(Number $length, Number $width, Number $height)
  @{
    $length * $width * $height;
  };

(Method Integer) save:(String $path)
  @{
     ug_savePartAs($path);
  };

(list) DemandValue: {saveOpResult:};
```

Illustrated Solutions

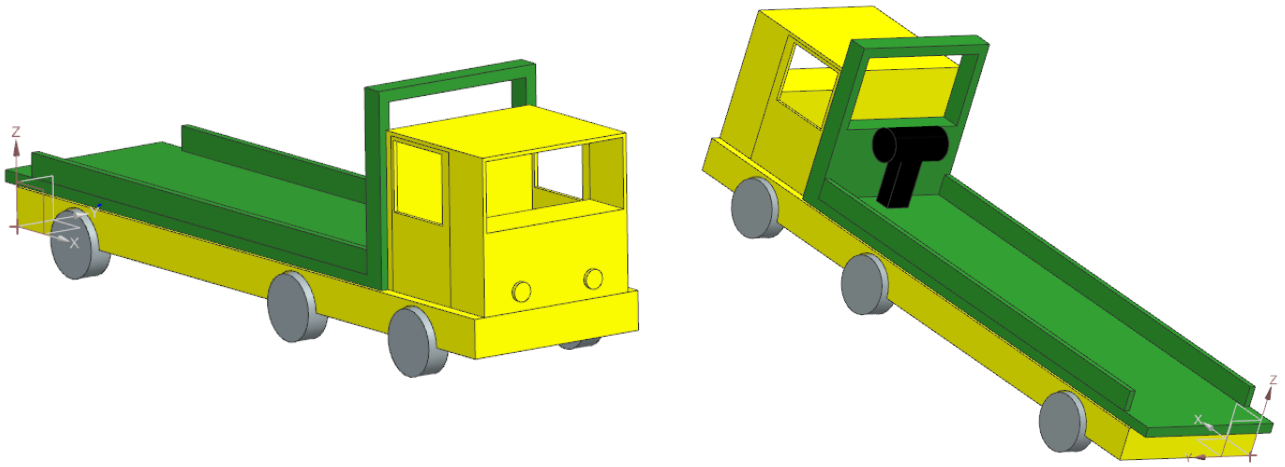
Generate one of the created trucks from the Web and correctly updated in the database. Then, this truck was not found in the Database, so we generated the new one and the values were written in the DFAs correctly

```
?truck a kbe:Truck.
?truck kbe:hasElement ?element.
?truck kbe:hasTruckLength ?truckLength.
?truck kbe:hasTruckWidth ?truckWidth.
?truck kbe:hasElementHeight ?elementHeight.
?truck kbe:hasElementLength ?elementLength.
?truck kbe:hasCabineHeight ?cabineHeight.
?truck kbe:hasDetailsColor ?detailsColor.
?truck kbe:hasElementColor ?elementColor.
?truck kbe:hasTruckColor ?truckColor.
```

Table | Response | 6 results in 0.01 seconds — Simple view ☐ Ellipse ☑ — Filter query results — Page size: 50

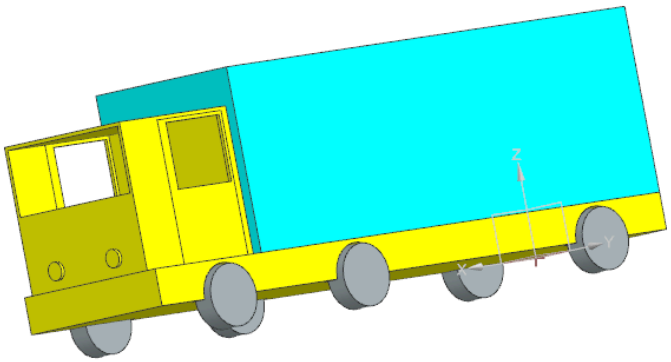
	truck	element	truckLength	truckWidth	elementLength	elementHeight	cabineHeight	detailsColor	elementColor	truckColor
1	<http://www.my-kbe.com/truck.o...	PlatformEle...	"190.0"^^<http://www.w3.org/2001/X...	"50.0"^^<http://www.w3.org/2001/X...	"150.0"^^<http://www.w3.org/2001/X...	"50.0"^^<http://www.w3.org/2001/XM...	"40.0"^^<http://www.w3.org/2001/XM...	BLACK	DARK_DULL_GRE...	RED
2	<http://www.my-kbe.com/truck.o...	PlatformEle...	"190.0"^^<http://www.w3.org/2001/X...	"50.0"^^<http://www.w3.org/2001/X...	"150.0"^^<http://www.w3.org/2001/X...	"50.0"^^<http://www.w3.org/2001/XM...	"40.0"^^<http://www.w3.org/2001/XM...	BLACK	DARK_DULL_GRE...	YELLOW
3	<http://www.my-kbe.com/truck.o...	TankElement	"190.0"^^<http://www.w3.org/2001/X...	"50.0"^^<http://www.w3.org/2001/X...	"150.0"^^<http://www.w3.org/2001/X...	"50.0"^^<http://www.w3.org/2001/XM...	"40.0"^^<http://www.w3.org/2001/XM...	BLACK	WHITE	RED
4	<http://www.my-kbe.com/truck.o...	TankElement	"190.0"^^<http://www.w3.org/2001/X...	"50.0"^^<http://www.w3.org/2001/X...	"150.0"^^<http://www.w3.org/2001/X...	"50.0"^^<http://www.w3.org/2001/XM...	"40.0"^^<http://www.w3.org/2001/XM...	BLACK	DARK_DULL_GRE...	BLACK
5	<http://www.my-kbe.com/truck.o...	BoxElement	"190.0"^^<http://www.w3.org/2001/X...	"50.0"^^<http://www.w3.org/2001/X...	"150.0"^^<http://www.w3.org/2001/X...	"50.0"^^<http://www.w3.org/2001/XM...	"40.0"^^<http://www.w3.org/2001/XM...	BLACK	DARK_DULL_GRE...	BLUE

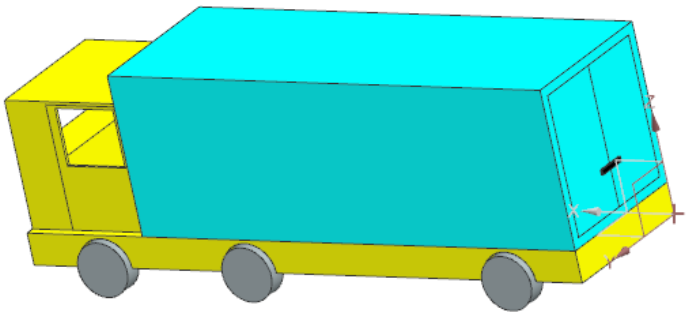
```
Truck found:  False
The UPDATE Query:

        PREFIX kbe:<http://www.my-kbe.com/truck.owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        INSERT
        {
        kbe:truck7446 a kbe:Truck.
kbe:truck7446 kbe:hasElement "PlatformElement"^^<http://www.w3.org/2001/XMLSchema#string>.
 kbe:truck7446  kbe:hasTruckLength "190.0"^^<http://www.w3.org/2001/XMLSchema#float>.
 kbe:truck7446  kbe:hasTruckWidth "50.0"^^<http://www.w3.org/2001/XMLSchema#float>.
 kbe:truck7446  kbe:hasElementHeight "50.0"^^<http://www.w3.org/2001/XMLSchema#float>.
 kbe:truck7446  kbe:hasElementLength "150.0"^^<http://www.w3.org/2001/XMLSchema#float>.
 kbe:truck7446  kbe:hasCabineHeight "40.0"^^<http://www.w3.org/2001/XMLSchema#float>.
 kbe:truck7446  kbe:hasDetailsColor "BLACK"^^<http://www.w3.org/2001/XMLSchema#string>.
 kbe:truck7446  kbe:hasElementColor "DARK_DULL_GREEN"^^<http://www.w3.org/2001/XMLSchema#string>.
 kbe:truck7446  kbe:hasTruckColor "YELLOW"^^<http://www.w3.org/2001/XMLSchema#string>.
} WHERE {}
```

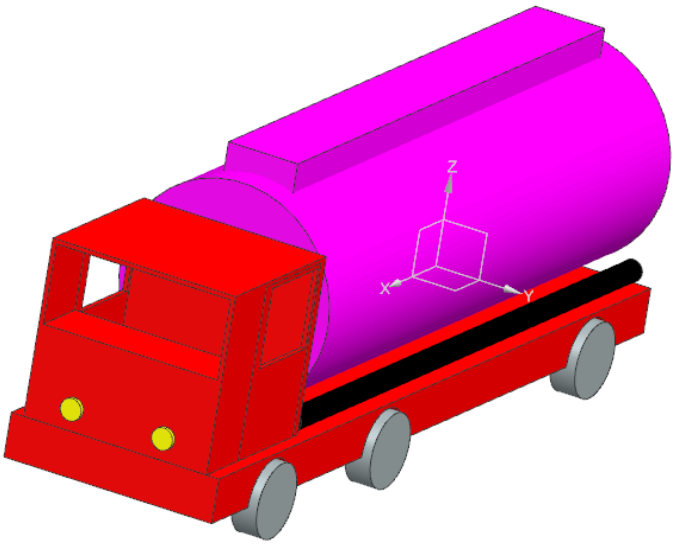


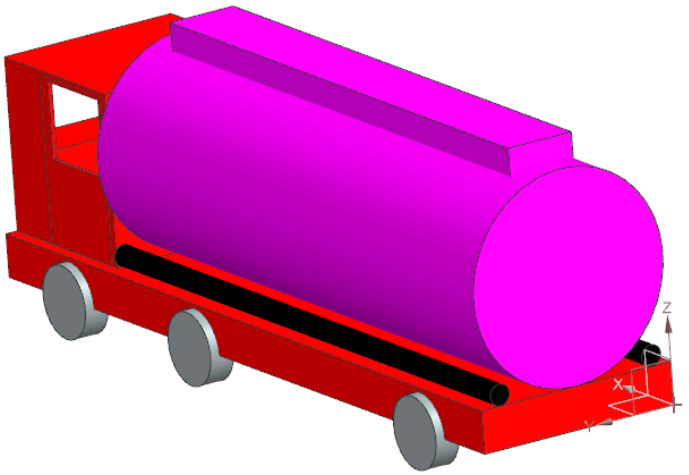
Now, we do the same with other trucks we want to generate

```
Truck found:  False
The UPDATE Query:

        PREFIX kbe:<http://www.my-kbe.com/truck.owl#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
        INSERT
        {
        kbe:truck6509 a kbe:Truck.
kbe:truck6509 kbe:hasElement "BoxElement"^^<http://www.w3.org/2001/XMLSchema#string>.
 kbe:truck6509  kbe:hasTruckLength "190.0"^^<http://www.w3.org/2001/XMLSchema#float>.
 kbe:truck6509  kbe:hasTruckWidth "50.0"^^<http://www.w3.org/2001/XMLSchema#float>.
 kbe:truck6509  kbe:hasElementHeight "50.0"^^<http://www.w3.org/2001/XMLSchema#float>.
 kbe:truck6509  kbe:hasElementLength "150.0"^^<http://www.w3.org/2001/XMLSchema#float>.
 kbe:truck6509  kbe:hasCabineHeight "40.0"^^<http://www.w3.org/2001/XMLSchema#float>.
 kbe:truck6509  kbe:hasDetailsColor "BLACK"^^<http://www.w3.org/2001/XMLSchema#string>.
 kbe:truck6509  kbe:hasElementColor "CYAN"^^<http://www.w3.org/2001/XMLSchema#string>.
 kbe:truck6509  kbe:hasTruckColor "YELLOW"^^<http://www.w3.org/2001/XMLSchema#string>.
} WHERE {}
```









Description of the Roles

We distributed all the parts of the assignment equally so that the hours dedicated will be the same for both.

Person	Marco	Alvaro	Both
Web Browser	X		
Designer Module		X	
Fuseki Server		X	
OWL Database	X		
NX			X
Report			X