<config>  
</config>

<**context** model= ‘E:\Dokumente\Studium\Master\MDSD\mdsd-factory-project\notation\de.mdelab.languages.factory.instance\VintageCarParts.factory’ element=’{0}’ searchMetamodels='true' importedBundles=’gmf;sirius’ />

<**gendoc** id=’title’>

[self.oclAsType(FactorySpecification).businessTypeName/] Owned By

[self.oclAsType(FactorySpecification).label /]

Table of content

[1. Summary 3](#_Toc455316259)

[1.1 Operational requirements 3](#_Toc455316260)

[1.2 Operational Context 3](#_Toc455316261)

[2. Operational Analysis 4](#_Toc455316262)

[1.3 Operational requirements 4](#_Toc455316263)

[1.4 Operational Context 4](#_Toc455316264)

[3. System Analysis 5](#_Toc455316265)

[3.1 System requirements 5](#_Toc455316266)

[3.2 System Functions 6](#_Toc455316267)

[3.3 [oStateMachine.name/] 6](#_Toc455316268)

[3.4 System Overview 7](#_Toc455316269)

[3.4.1 Behavior of system 7](#_Toc455316270)

[3.4.2 System function 8](#_Toc455316271)

[3.4.3 Data transfer 8](#_Toc455316272)

**Table of Figures**

[Figure 1 : 3](#_Toc420585209)

**</gendoc>**

<**gendoc** id=’summary’><drop/>

# Summary

[if self.oclAsType(FactorySecification).ownedFactories->notEmpty()]<drop/>

Table 1 lists the [self.oclAsType(FactorySpecification).businessTypeName.toLower()/] owned by [self.oclAsType(FactorySpecification).label/]:

<table><drop/>

**Table 1 : The** list of [self.oclAsType(FactorySpecification).businessTypeName.toLower()/] owned by [self.oclAsType(FactorySpecification).label/]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Address | Dimensions (meters) | Employees | Products Made |

[for (factory : factory::Factory | self.oclAsType(FactorySpecification).ownedFactories->sortedBy( label ) )]<drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [factory.name/] | [factory.streetNumber/][factory.streetNumber/]  [factory.postalCode/][factory.city/]  [factory.country/] | [factory.width/] X [factory.length/] | [for ( employee : factory::Employee | factory.employees->sortedBy( label ) )]<drop/>  [employee.label/] | [for ( product : factory::Product | factory.ownedProducts->sortedBy( name ) )]<drop/>  [product.label/] |

[/for]<drop/>

</table><drop/>

Each factory is described in details in the following sections.

[else] <drop/>

No factories.

[/if]

**</gendoc><drop/>**

<**gendoc** id=’operationalAnalysis’><drop/>

# Operational Analysis

## Operational requirements

[if (requirement::RequirementsPkg.allInstances()->select(name = 'Operational Requirements').ownedRequirements)->notEmpty()]<drop/>

The following table shows the different operationals requirements:

<table><drop/>

|  |  |  |
| --- | --- | --- |
| ID | Abstract | Description |

[for (oReq : requirement::SystemUserRequirement | requirement::RequirementsPkg.allInstances()->select(name = 'Operational Requirements').ownedRequirements->filter(requirement::SystemUserRequirement)->sortedBy(requirementId))]<drop/>

|  |  |  |
| --- | --- | --- |
| [oReq.requirementId/] | [oReq.name/] | [oReq.summary/] |

[/for]<drop/>

</table><drop/>

[else] <drop/>

No operational requirement.

[/if]

## Operational Context

The system operational context is illustrated in following diagrams:

[for (element:OperationalContext |OperationalContext.allInstances())]<drop/>

[for (di : notation::Diagram | element.getSiriusDiagrams('my\_diagrams\_file\_name.aird', 'my\_diagram\_name'))]<drop/>

<image object='[di.getDiagramExt('jpg')/]' maxW='true' maxH=’true’> <drop/>

</image>

Figure 1 : [di.getSiriusDiagramName().clean()/]

[/for]<drop/>

[/for]<drop/>

For each system entity, their activities are displayed:

[for (oEntity : Entity | Entity.allInstances()->sortedBy(name))]<drop/>

* [oEntity.name/]

[for (oActivity : OperationalActivity | oEntity.ownedFunctionalAllocation.targetElement->filter(OperationalActivity)->sortedBy(name)->reject(name =’Root Operational Activity’))]<drop/>

*[oActivity.name/]*

[/for]<drop/>

[/for]<drop/>

**</gendoc><drop/>**

<**gendoc** id=’systemAnalysis’><drop/>

# System Analysis

## System requirements

[if (requirement::RequirementsPkg.allInstances()->select(name = ‘System Requirements’).ownedRequirements)->notEmpty()]

The following table shows the different operationals requirements:

<table><drop/>

|  |  |  |
| --- | --- | --- |
| ID | Abstract | Description |

[for (oReq : requirement::SystemUserRequirement |requirement::RequirementsPkg.allInstances()->select(name = ‘System Requirements’).ownedRequirements->filter(requirement::SystemUserRequirement)->sortedBy(requirementId))]<drop/>

|  |  |  |
| --- | --- | --- |
| [oReq.requirementId/] | [oReq.name/] | [oReq.summary/] |

[/for]<drop/>

[for (oReq : requirement::SystemNonFunctionalInterfaceRequirement | requirement::RequirementsPkg.allInstances()->select(name = 'System Requirements').ownedRequirements->filter(requirement::SystemNonFunctionalInterfaceRequirement)->sortedBy(requirementId))]<drop/>

|  |  |  |
| --- | --- | --- |
| [oReq.requirementId/] | [oReq.name/] | [oReq.summary/] |

[/for]<drop/>

</table><drop/>

[else] <drop/>

No system requirement.

[/if]

## System Functions

Below the table of correspondence between the functions and activities of the system:

<table><drop/>

|  |  |
| --- | --- |
| System functions | Operational Activities |

[for (oSytmFunct : SystemFunction | SystemFunction.allInstances()->sortedBy(name)->reject(name = 'Root System Function'))]<drop/>

|  |  |
| --- | --- |
| [oSytmFunct.name/] | [if (oSytmFunct.ownedFunctionRealizations.targetElement->notEmpty())]<drop/>  [for(oActivity : OperationalActivity | oSytmFunct.ownedFunctionRealizations.targetElement->sortedBy(name)->filter(OperationalActivity))]<drop/>  [oActivity.name/]  [/for]<drop/>  [else]  [/if]<drop/> |

[/for]<drop/>

</table><drop/>

[for (oSys : System | System.allInstances())]<drop/>

[for (oStateMachine : StateMachine | oSys.ownedStateMachines->sortedBy(name))]

## [oStateMachine.name/]

The behavior of the System is described in the following state diagram:

[for (element: Region | oStateMachine.ownedRegions)]<drop/>

[for (di: notation::Diagram | element.getSiriusDiagrams('my\_diagrams\_file\_name.aird', 'my\_diagram\_name'))]<drop/>

<image object='[di.getDiagramExt('jpg')/]' maxW='true' maxH=’true’> <drop/>

</image>

Figure 2 : [di.getSiriusDiagramName().clean()/]

[/for]<drop/>

[/for]<drop/> -- For this specification, we are limited to 3 level research

The different states of the system are:

[for (oRegion: Region | oStateMachine.ownedRegions->sortedBy(name))]<drop/>

[for (oStateFirst : State | oRegion.ownedStates->sortedBy(name)->filter(State))]<drop/>

* [oStateFirst.name/]

[for (oStateSecond : State | oStateFirst.referencedStates->sortedBy(name)->filter(State))]<drop/>

*[oStateSecond.name/]*

[for (oStateThird : State | oStateSecond.referencedStates->sortedBy(name)->filter(State))]<drop/>

*[oStateThird.name/]*

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

## System Overview

### Behavior of system

The behavior of system is described in these following diagrams :

[for (element: System | System.allInstances())]<drop/>

[for (di:notation::Diagram | element.getSiriusDiagrams('my\_diagrams\_file\_name.aird', 'my\_diagram\_name'))]<drop/>

<image object='[di.getDiagramExt('jpg')/]' maxW='true' > <drop/>

</image>

Figure 3 : [di.getSiriusDiagramName().clean()/]

[/for]<drop/>

[/for]<drop/>

### System function

For each block illustrated in diagrams, functions are listed below:

[for (oSys : System | System.allInstances()->sortedBy(name)->filter(System))]<drop/>

**Bloc [oSys.name/] :**

<list><drop/>

[for (oFunct : SystemFunction | oSys.ownedFunctionalAllocation.targetElement->sortedBy(name)->filter(SystemFunction))]<drop/>

* + [oFunct.name/]

<richText>[oFunct.description/]</richText>

[/for]<drop/>

</list><drop/>

[/for]<drop/>

[for (oAct : Actor | Actor.allInstances()->filter(Actor))]<drop/>

**Bloc [oAct.name/] :**

<list><drop/>

[for (oFunct : SystemFunction | oAct.ownedFunctionalAllocation.targetElement->filter(SystemFunction))]<drop/>

* + [oFunct.name/]

<richText>[oFunct.description/]</richText>

[/for]<drop/>

</list><drop/>

[/for]<drop/>

### Data transfer

For each exchange illustrated in diagrams, here the data transfered:

[for (oExch : ComponentExchange |ComponentExchange.allInstances()->sortedBy(name))]<drop/>

[if (oExch.ownedComponentExchangeFunctionalExchangeAllocations.targetElement.oclAsType(FunctionalExchange)->notEmpty())]<drop/>

**Link [oExch.name/] :**

[for ( oFunctExch : FunctionalExchange|oExch.ownedComponentExchangeFunctionalExchangeAllocations.targetElement.oclAsType(FunctionalExchange)->sortedBy(name))]<drop/>

* + [oFunctExch.name/]

[/for]<drop/>

[/if]<drop/>

[/for]<drop/>

**</gendoc><drop/>**