PROJECT REPORT

ON

Deliverable #3

Management Of Cars Manufacturing Resources

Presented By

- Kevin Kim 29781110
- Oussama Jebbar 27750897
- Amandeep Sharma 27260164
- Pilli Sushmita Roy 27676794
- Juhi Arora 27228198

March, 4th, 2016

Introduction

This document provides detailed description of every classes represented in the E-Inventory Class Diagram.

Inventory

categoryDao
+ addItemCat...
+ listCategori...
+ updateCate...
+ removeCate...

ItemDao
+ additemRefN...
+ addprovider()

Warehouse
+ location: String [1]
+ warehouseId: Int...
+ getlocation()
+ setlocation()
+ getwarehouseId()
+ setwarehouseId()

WarehouseDao
+ addWarehouse()
+ listWarehouse()
+ updatewareho...
+ removeWareh...

use
+ warehouse
use

ItemCategory
+ name: String [1]
+ description: Str...
+ dimensions: <...
+ getName()
+ setName()
+ getdescription()
+ setdescription()
+ getdimension()
+ setdimension()

+ itemcategory
use

StorageCapacity
+ capacity: Integer [1]

+ warehouse
+ warehouse

Storage
+ count: Integer [1]

inventoryItem
+ referenceNumbe...

+ inventoryitem

Item
+ itemRefNum: Int...
+ providername: S...
+ getitemRefNum()
+ setitemRefNum()
+ getprovidername()
+ setprovidername()

+ itemcategory
1
1
+ item

+ itemcategory

{?} CapacityUpperBoundConstraint
{{OCL} self.count<(if(self.inventoryitem.isTypeOf(Item)) self.warehouse.storagecapacity.select(c|c.itemcategory=self.inventoryitem.itemcategory).first().capacity else self.productlinestoringcapacity.select(p|p.productline=self.inventoryitem.productline).first().capacity)}

supplyChain

ProductLineDAO
+ addProductLine()
+ removeProductLine()
+ updateProductLine()
+ listProductLine()

Part
+ count: Integer [1]

ProductionUnitDAO
+ addProductionUnit()
+ removeProductionUnit()
+ updateProductionUnit()
+ listProductionUnit()

ProductLineStoringCapacity
+ capacity: Integer [1]

+ productline
use
use

ProductLine
+ productLine: String [1]
+ getProductLine()
+ setProductLine()

+ productline
+ productline
1
*
+ productionunit

ProductionUnit
+ productionDuration: <Undefined> [1]
+ productionDueDate: <Undefined> [1]
+ productionStatus: <Undefined> [1]
+ color: String [1]
+ getProductionDuration()
+ setProductionDuration()
+ getProductionDueDate()
+ setProductionDueDate()
+ getProductionStatus()
+ setProductionStatus()

+ productline

{?} AuthorizedToReportDefect
{{OCL} self.user.role.roleName='PlanningAgent'}

+ productionunit
1..*
+ productionunit
+ productionunit

Defect
+ defectId: Integer [1]
+ defectReportDate: EDate [1]
+ description: String [1]

Batch
+ batchSize: Integer [1]
+ getBatchSize()
+ setBatchSize()

Car

Order
+ dateOfTheOrder: EDate [1]
+ orderId: Integer [1]
+ orderDeadline: EDate [1]

0..1
+ batch

{?} Constraint1
{{OCL} self.productunit.forAll(p|p.productline=self.productline)}

{?} AuthorizedToPlaceOrder
{{OCL}
self.user.role.roleName='PlanningAgent'}

util

+ user
+ user
User
- firstname: <Undefined> [1]
- lastname: <Undefined> [1]
- loginID: <Undefined> [1]
- password: <Undefined> [1]
- empID: <Undefined> [1]
+ getFirstName()
+ setFirstName()
+ getLastName()
+ setLastName()
+ getLoginID()
+ setLoginID()
+ getEmployeeID()
+ setEmployeeID()

Role
+ roleName: String [1]
+ setRoleName()
+ getRoleName()
+ addPermission()
+ getPermissions()

+ role
1
*
+ user

SecurityUtils
+ execute()
+ next
1
1
+

use
1..*
+ role

UserDAO
+ addUser()
+ updateUser()
+ removeUser()
+ listUser()

use

RoleDAO
+ addRole()
+ updateRole()
+ removeRole()

LoggingUtil

AuthorizationUtil

AuthenticationUtil

+ permission
1..*

Permission
+ permissionId: Integer [1]
+ permissionDescription...
+ setPermission()
+ checkPermission()

use

PermissionDAO
+ setAdminPermission()
+ setPlanningAgentPermis...
+ setInventoryAgentPermis...
+ setSupplyChainAgentPer...

Supply Chain Domain

ProductLine

Description
A ProductLine represents a group of related ProductionUnit sharing the same model type. All the elements of the ProductLine are ProductionUnit assembled using the same parts of the same item category with the same quantifies (process in the same manner). A ProductLine usually is associated with the same car model over its lifecycle. Therefore, over this period, the ProductLine will have multiple ProductionUnit of the same productLineName.

Attributes
- productLineName : String

  Denotes the ProductLine name. It is an abstraction to a car model. For example, “Civic” can be a productLineName from “Honda” manufacturer

Association
- productionUnit : ProductionUnit[0..*]

  A set a ProductionUnit associated with this ProductLine. A ProductionUnit can be a single car or a batch of many. A ProductionUnit has a single ProductLine. A ProductLine can have multiple ProductionUnit.

- Parts : Item[1..*]

  ProductLines are distinguished by their productLineName. Each ProductLines is characterized by a specific set of parts of the same item category, and quantity because it produces a specific car model. For example, a ProductLine can have 2 doors, 4 windows, etc., while another ProductLine can have 4 doors, 6 windows, etc.

Constraints
No additional constraints

ProductionUnit

Description
A productionUnit is an abstraction to a production iteration in a car manufacturing process. A productionUnit could be a single item produced, which would be a car in this case, or it could be a batch if it is multiple cars produced. The manufacturing of a batch of cars will have a variable production time depending of the batch size. A ProductionUnit needs to be complete by a specific due date. A ProductionUnit may have three states: PENDING, INPROGRESS or COMPLETED.

Attributes
- productionDuration: Time

  Denotes the duration it takes for a ProductionUnit to complete the manufacturing process

- productionDueDate: Date

  Denotes the date a ProductionUnit is due and needs to be COMPLETED.

- productionStatus: String

  An attribute defining the status of a unit at given point of time. A ProductionUnit may have three states: PENDING, INPROGRESS or COMPLETED.

Associations
- ProductLine: [1]

  A many to one association to ProductLine shows that a Single ProductLine may have one or many product units specific to unique car brands.

- Batch and Car: [0..1]

  A production unit may contain a single unit of car or a batch of cars forming a composite pattern which shows a batch can further contain one or more production units.

Constraints
No additional constraints

Car

Description
This Class depicts a single car unit produced in a ProductionUnit

Associations
- ProductionUnit: [1]

  A parent-child relationship denotes a production unit may contain one or many cars sharing similar characteristics or attributes.

Constraints
No additional constraints

Batch

Description
This Class depicts a ProductionUnit which contains a group of cars.

Associations
- ProductionUnit: [1]

  A composite association/relationship showing a batch may contain one or more production units or car units.

Constraints
All production units within a batch belong to the same car product line

Defect

Description
Defect is an association class, which depicts the defective manufactured products within a ProductionUnit and can be reported by a user with permissions to report defects. A ProductionUnit can be defective for multiple reasons. When a defect is reported, details such a description of the defect and a report date are provided.

Attributes
- defectID : Int

  Denotes the Identifier ID for this defect

- defectReportDate : EDate

Denotes the date this defect was reported

- description : String

Denotes a textual description of the defect reported

Associations
This is an association class between ProductionUnit and User.

Constraints
Planning agent is the only one who is allowed to report defects.

Order

Description
This is an association class of product line between user class in the supply chain package. It depicts an order can be placed by an user with the permission to order that goes into the productline to be produced.

Attributes
- dateOfTheOrder : EDate

  Indicates the date of the order placed by the planning agent.

- orderID : Int

Every order has a unique ID described here as an integer value.

- orderDeadline : Edate

Indicates the date before which the product should be manufactured.

Associations
This is an association class between User and ProductLine

Constraints
Only a planning agent is authorized to place an order.

Part

Description
An association class between ProductLine and ItemCategory. This association class is an abstraction to the parts necessary to produce the specific ProductLine car model defined by the productLineName. For each ItemCategory related to a ProductLine, there will be a Parts class defined holding the count of this item. For example, if a specific ProductLine is a hatchback, then it will have a Parts class instantiation holding the count of 3 for the ItemCategory named door.

Attribute
- Count: int

  An attribute defining the number of car parts associating with both ItemCategory and ProductLine classes.

Associations
This class is an association class between ProductLine and ItemCategory

Constraints
No additional constraints

Inventory Domain

ItemCategory

Description
An item category is the type of the part on a car (or an item), it can be a door, wheel, side mirror, etc. Each category has a name, description, and dimensions (length, width and height) these name, description and dimensions falls into the category of attributes in technical terms.

Attributes
- Name: String
  Denotes the name of the item .For Example, Side door of a particular company “X”.

- Description : String
  Denotes the description of an item .For Example, description provides the details about what is included and what is not included in the item called " front end assembly " for the car "Civic " of company "Honda" .
- Dimensions: Integer
  Denotes the dimensions (length, width and height) of an item .For Example, A battery for the car model "Civic" of company "Honda "could be 10, 6 ¾, 7 ¾ (Length, width and height) respectively.

Associations

- itemCategory: Item[1..1]
  An ItemCategory is associated with an Item. An item is a combination of itemRefNum and providername from whom that item is purchased from. For Example, Wheels for the car "Civic" will have an item reference number and dealer details from it was purchased. Now, this item reference number could be same for all the wheels of the car "Civic".

- itemCategory : Productline

  Item category imports elements to ProductLine. A ProductLine can import any number of items from any number of categories by identifying the item by using the Name, Description and Dimensions. For Example, Production Line Manufacturing 500 cars of model "Civic" for Company "Honda", then ProductionLine imports 500 wheels,500 right side mirrors and so on.

Constraints

No associated constraints.

Item

Description

An Item is an occurrence of an ItemCategory. An Item is identified with a reference number and with the name of its provider in order to keep track of each Item within the inventory.

Attributes

- itemRefNum: Integer

  Denotes the reference number for each item and belongs to a particular item category. Example: A Windshield belonging to a Company 'X' could have 8 digits reference number to that Windshield.
- providerName: String

  Denotes the name of the provider from whom it was purchased. Example: A tire of Company 'X' could be purchased from a dealer "abc".

Associations

- item: inventoryItem [1]
  This association denotes the association between the Item class and which ItemCategory it belongs. An Item belongs to one ItemCategory.
- item : ProductionUnit[1]
  This association with ProductionUnit denotes all the Items needed for the production of a specific ProductionUnit. A ProductionUnit can be a single car or a batch of many. Therefore this association will denote all the parts necessary for the manufacturing of those cars belonging to this ProductionUnit.

Constraints

No associated constraints

WareHouse

Description

A warehouse is a location with an address that can hold items of different categories with their respective maximum capacities. The management of warehouses falls into the responsibilities of the inventory agent.

Attributes

- location : String

  Denotes the physical address of the Warehouse along with the building number, address postal code city and country. For Example, Honda can have a warehouse in dorval at some physical address and the same Honda company can have another Warehouse in Montreal & both will have different physical address that's how the attribute Location is worked in this context.

- wareHouseId : Integer

  Denotes the numerical identifier for the warehouses. For Example, Company "Honda" have two different warehouses in the city of Montreal but one warehouse stores front end assembly items and the other ware house stores the back end assembly items. So, in order to avoid the confusion each warehouse is given with a unique warehouse Id and that Id must be in numerical format.

Associations

- warehouse : Inventoryitem[0..*]

  Warehouse imports element's from the inventoryItem .Warehouse can import any number of elements from the inventoryItem using the items reference number.This refernce number would be in integer format /Numerical format/integer data type. For Example, A warehouse that belongs to the Company "Honda" will place an inventory which includes the list of items they need along with the items reference number after a certain period of time, warehouse will receive all the items they need in the warehouse.
- warehouse: Capacity[0..*]
  WareHouse is connected with an abstraction class known as "Capacity". It indicates the capacity or the number of items a warehouse can hold. A warehouse can have a different capacity for holding an item. For Example, A warehouse "123"can hold 100 wheels but it can hold 10 doors.
  So, the capacity of holding items varies in number depending on the size of the item or on the physical status of the warehouse building.

Constraints
At any time, a warehouse cannot store more items than its capacity allows.

Utils

User

Description
This class is an abstraction to the different types of users of the system. A user could be a planning agent, supply agent or an admin. Each user has a specific role and permissions according to the type of job they performs and they can be identified with their respective employ ID.

Attributes
- firstName : String Denotes the first name of the User
- lastName: String Denotes the last name of the User
- loginID : Int Denotes the login id of the User
- password : Int Denotes the password of the User
- employeeID : Int Denotes the employee id of the User

Associations
Role[1..1]

This one to one association between user and role shows Each user has a specific role with which they can perform their desired functions.

Constraints
No additional constraints

Role

Description
This class is an abstraction to the different roles a user can be assigned in the system. Each user has some specific role in the system and performs functions according to the assigned role. For example, a user could have a role of supply agent to perform functionalities related to the supply chain domain. Alternately, another user could have a role of planning agent which has access to another set of functionalities.

Attributes
- roleName: String Denotes the identifier of this role

Associations

Permission[*..*]

This many to many association between the two classes represents corresponding functionalities that the user is permitted to perform.

Constraints
No additional constraints

Permission

Description
This class assigns permissions to the different users of the system depending on their roles so that they can perform desired operations. For example, A user with an admin role could be assigned permissions to manage users.

Attributes
- permissionID: Int

Denotes the permission ID assigned to this class

- description: String

Provides a textual description of the functionalities provided by this permission

Associations
- role [*..*]

This many to many association between the two classes represents corresponding functionalities that the user is permitted to perform.

Constraints
No additional constraints

SecurityUtils

Description
This class checks the logging, authorization and authentication whenever the user performs any operation according to the assigned role and permission.

- **LoggingUtil** checks the login details of the user while performing an operation.
- **AuthorizationUtil** checks whether the user is authorized to perform the required operation.
- **AuthenticationUtil** checks the credentials needed to operate the required operations.

Attributes
No additional attributes

Associations

- next[1..1]

  This is a reflexive association that calls the next security operation to be performed until all the three validation steps are performed.

Constraints

No additional constraints