ACO CAR TAILOR

A Project Design Report

Submitted by:

AXEL BROULAND

KIRANPREET KAUR

MOHAMED ABDULLAH AMANULLAH

Submitted to:

MARC BOUSSE

in partial fulfillment for the award of the degree

of

MASTER 1

IN

CLOUD COMPUTING AND SERVICES

at

UNIVERSITÉ DE RENNES 1

RENNES (FRANCE)

CAR TAILOR - UML

➢ Car Tailor - Users Stories

Name of the story : Get list of categories
Preconditions : None
Basic scenario :
1) The user wants to get the list of categories.
2) The system prints the list of categories.
3) End of the story

Name of the story : Select part type for given category
Preconditions : Get list of categories, and choses 1 category
Basic scenario :
1) The user wants to get the list of the parts in the category.
2) The system prints the list of parts.
3) User selects 1 part
4) End of the story

Name of the story : Check validity
Preconditions : User has to select 1 part for each category
Basic scenario :
1) The user wants to check the validity of his configuration
2) The user click on the “Validate” button
3) If it’s valid then
4) System prints message to say “Your decision is successful”.
5) Else if it’s not valid then
6) System prints message “Your parts “...” and “...” are incompatible”
7) Else
8) System prints message “Chose one part for each category
9)End if
10) End of the story

Name of the story : Remove Part
Preconditions : Get list of categories
Basic scenario :

1) The user wants to remove a part of his configuration
2) The system delete the part from the configuration
3) End of the story

Name of the story : Edit incompatible required
Preconditions : Login as administrator
Basic scenario :

1) If The administrator wants add new incompatibilities then
2) The system prints a form to add incompatibilities for the piece
3) Administrator fill the form and validate
4) System add the incompatible part to the list.
5) If The administrator wants add new required part then
6) The system prints a form to add required part for the piece
7) Administrator fill the form and validate
8) System add the required part to the list.
9) If The administrator wants delete incompatibilities then
10) The system prints a form to delete incompatibilities for the piece
11) Administrator fill the form and validate
12) System delete the incompatible part to the list.
13) If The administrator wants delete required part then
14) The system prints a form to delete required part for the piece
15) Administrator fill the form and validate
16) System delete the required part to the list.
17) End of the story

Name of the story : HTML Description of Configuration
Preconditions : Configurations complete and Valid
Basic scenario :

1) The user wants to get an HTML description of configuration.
2) The system check the validations of configuration and then display if it's fulfilled.
3) End of the story

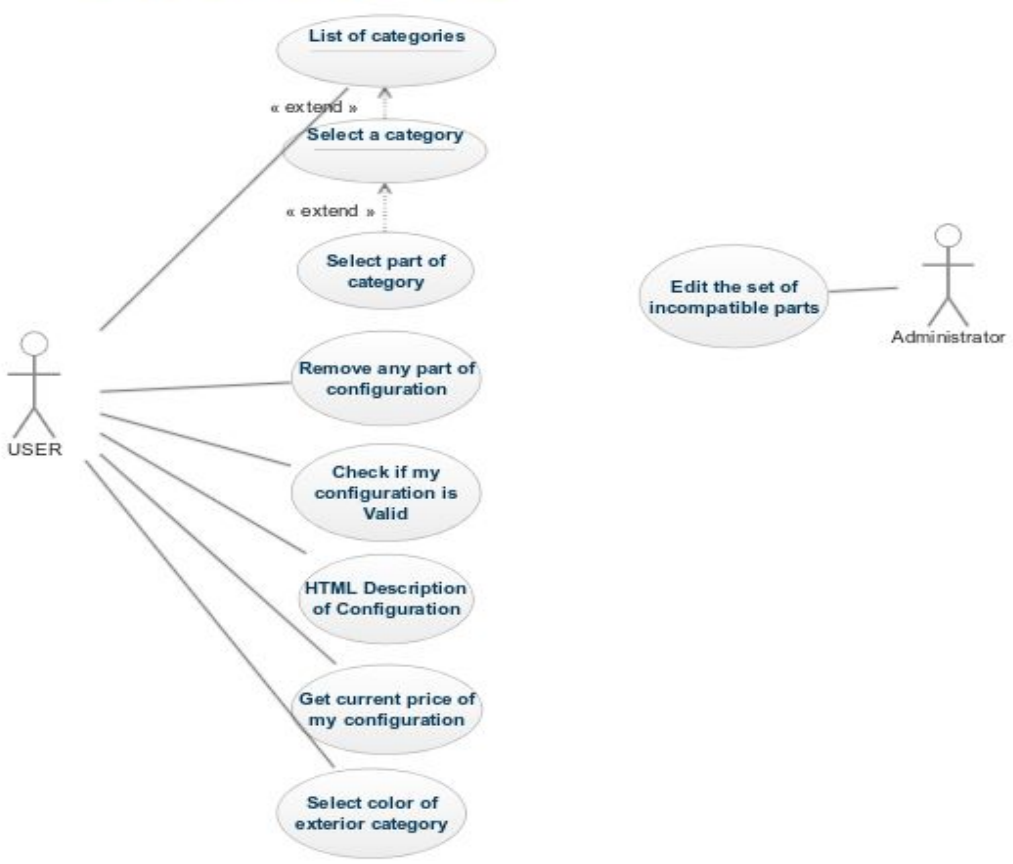
Name of the story : Get current price of configuration
Preconditions : Get list of categories
Basic scenario :

1) The user wants to get the current price (in euros) of his configuration
2) If it is valid then
3) The system prints the price of configurations.
4) End of the story

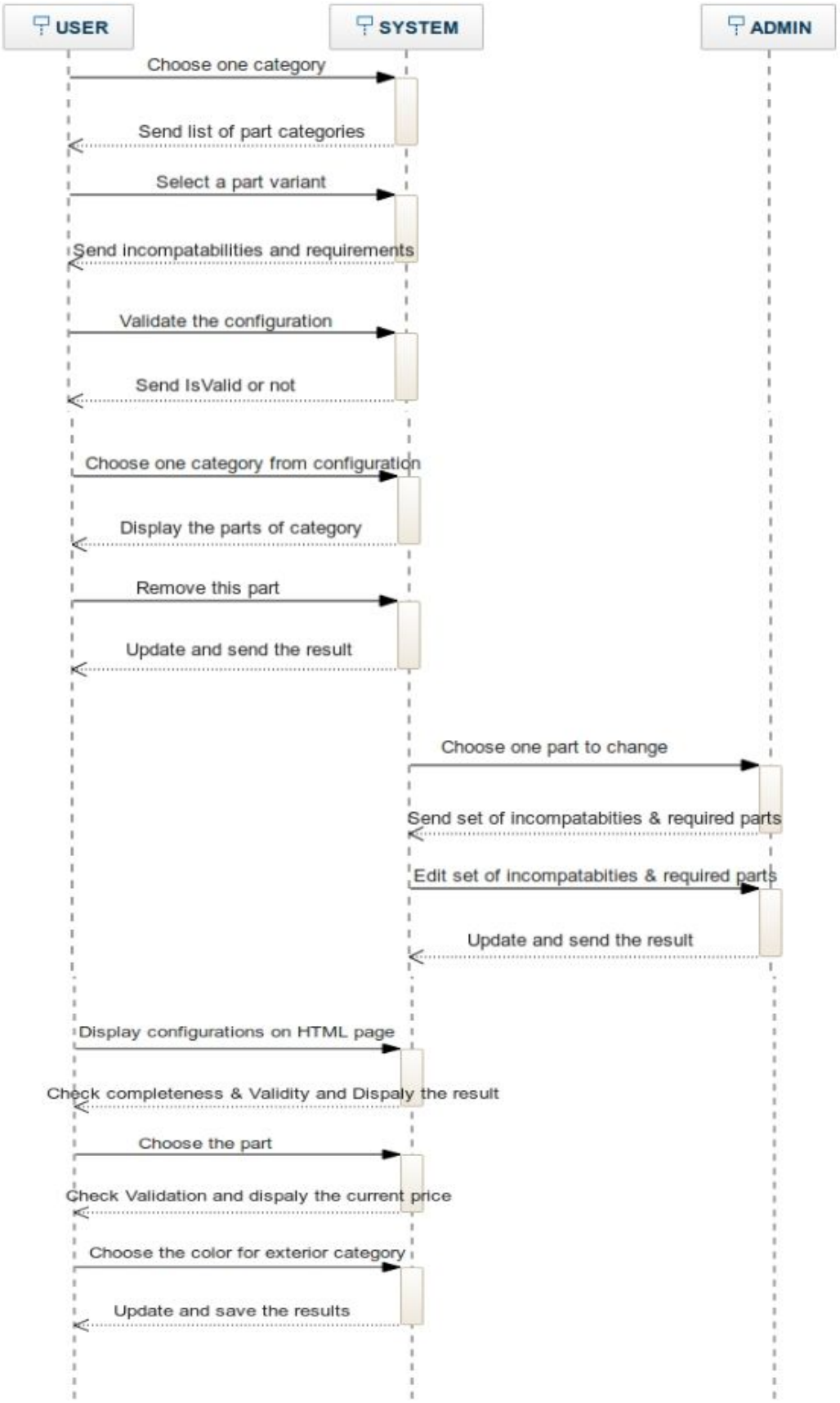
Name of the story : Select color of exterior category
Preconditions : User has to select one part for exterior category
Basic scenario :

1) The user wants to choose the color of exterior part
2) The system prints the list of colors.
3) User selects one color
4) The system update the color of part exterior from the configuration
5) End of the story

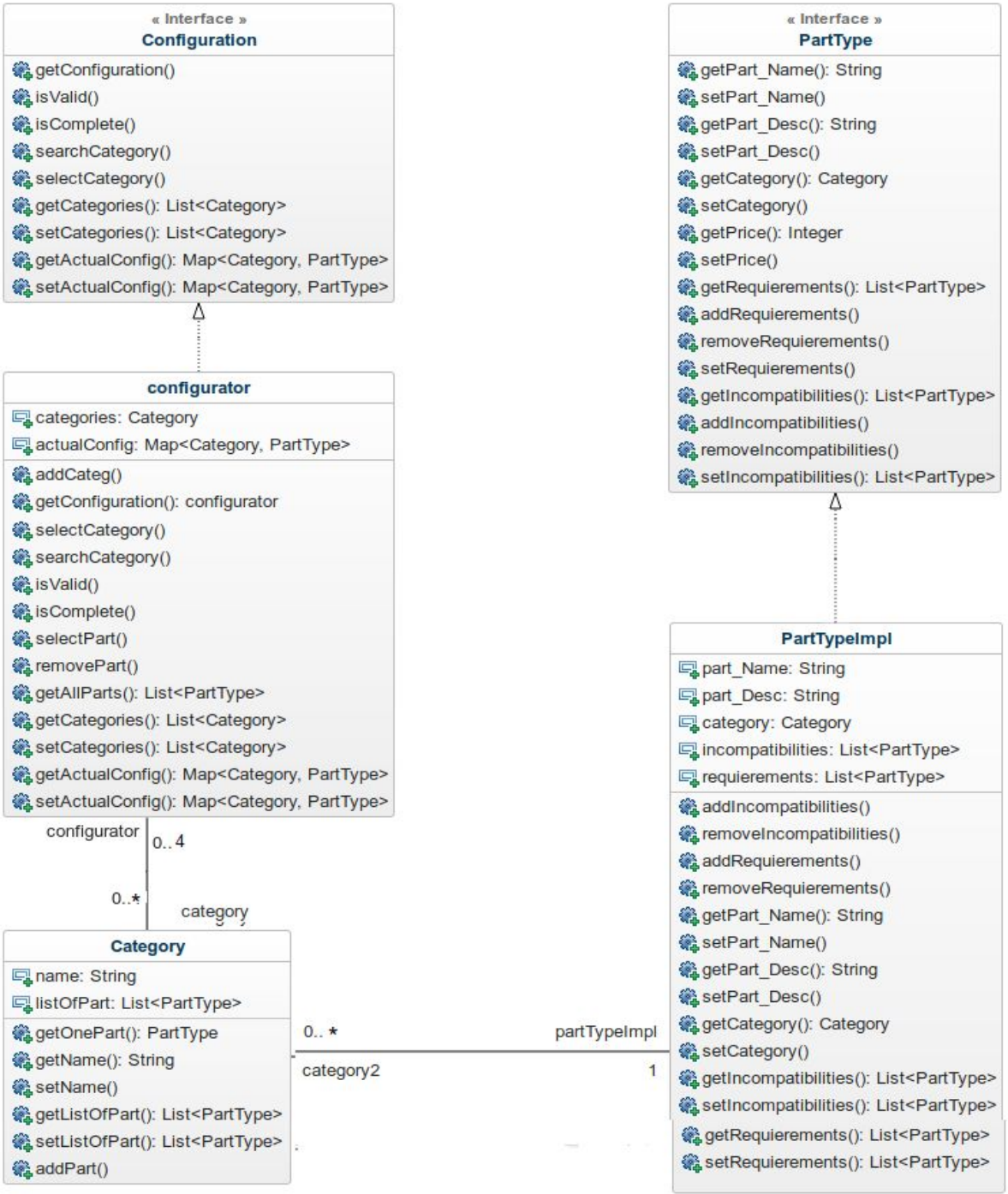
➢ Car Tailor - Use Case Diagram



➢ Car Tailor - Sequence Diagram



➢ <u>VERSION 1</u>: Car Tailor - Class Diagram



➢ VERSION 2 : Car Tailor - Class Diagram

➢ TEST CASE RESULTS

Name	Description	Expected Result	Actual Result	Pass/Fail
userStartsConfigurator	Initially check the state of configuration	1. Empty configuration must be valid 2. Empty configuration is not complete.	1. isValid()= true 2. isComplete = false	Pass
creationOfCategory	Add new category in the configuration	Category should be well recuperate	Category is added	Pass
selectAPart	Select the part of category, check validity and completeness	The selected Part should be validated.	Part is selected and Validated	Pass
configIsCompleteAnd Valid	To check if the Configuration is Valid.	It must be valid and complete	Configuration is validated and complete	Pass
partNeedRequirement	Add a part without its requirements, add the requirement after	First it's not valid and valid after the add	False then true then ok	Pass
select2Incompatible Part	Check incompatibilities of parts	Incompatibilities is Checked Configuration isn't valid	verify the Incompatible PartIncompatible parts. Configuration isn't valid	Pass
Remove a Part	Selected part is removed Remove part from configuration	Part is removed Part should be deleted	Part is removed	Pass
priceOfValidConfig	Get current price of configuration	Display current price if it is valid	Display current price and is valid	Pass
changeColor	Add color to exterior then change it	Color is added, changed and checked	At the end the correct color is on the part	Pass
creationHTML	Create the file of the HTML description in the folder	File is created	File is created	Pass
changeRequirements	Edit the requirements list (add and delete)	Add one part and delete it after	Test during the add and the delete are working	Pass
changeIncompatibilities	Edit the incompatibilities list (add and delete)	Add one part and delete it after	Test during the add and the delete are working	Pass

CONCLUSION

The project “Car Tailor” was really rewarding for us, and this for different reasons. First we learn a lot about the design pattern. We try different design pattern as the MVP or the Observer. But finally we chose the Composite. It was interesting to see how design pattern could influence the way of working of a little project like this one and make it totally different in the way of executing.

In a second time, we learnt a lot about the team work, because we had to divide the tasks in order to make it more easy and faster to do it.

During the development we encounter different problems but one of them was really remarkable. We had to search the best way to implement the possibility to change the color of an Exterior. We decided to create a PartColor which is a subtype of PartType to make it easier to use Exterior as a normal part and as a PartColor when we need to change the color, then we didn’t have to change all the code from the V1.