

Second iteration deliverable

ANDRIANARISOLO Elie

DOS SANTOS Hélène

SARR Seynabou

SERPINET Gaspard

THAIZE Nicolas

TONG An Jun

VILLEROY Billy

Table of contents

Glossary	4
Domain model	7
Use case diagram	8
Structured descriptions of the main use cases	9
1 : Load a city map	9
2 : Load a delivery requests file	10
3 : Create a delivery request	11
4 : Assign a delivery request to a tour	12
5 : Determine an optimized tour	13
6 : Generate Road Map	14
7 : Manage courier	15
State chart diagram	16
MVC Architecture and design patterns	17
MVC	17
Design patterns	18
Commands	18
State	18
Observer	18
Class and package diagrams	19
Differences between these diagrams and first iteration diagrams	24
Planning	25
First iteration	25
Expectation	25
Reality	26
Second iteration	27
Expectation	27
Reality	28
Third iteration	29
Expectation	29
Reality	30
Fourth iteration	31
Expectation	31
Reality	32
Total	33
Expectation	33
Reality	33
Test Report	34
Technical review	38
Human review	38
Environmental and Social issues	39
Personal review and commentary	40
ANDRIANARISOLO Elie	40
	2

DOS SANTOS Hélène	40
SARR Seynabou	40
SERPINET Gaspard	41
THAIZE Nicolas	41
TONG An Jun	41
VILLEROY Billy	42

Glossary

Term	Definition
Arrival time	The moment when a courier reaches a delivery point.
City map	<ul style="list-style-type: none"> Detailed and visual overview of the geographical layout of a city through a 2D plan. It comprises a set of nodes representing the intersections within the city, and the road segments between these nodes. It encompasses the location of the warehouse, which serves as the central starting point for courier tours.
Courier	<ul style="list-style-type: none"> Individuals responsible for transporting and delivering packages within a city using bicycles. Couriers are assigned to a tour, and follow an optimized route visiting each delivery location within its designated time-window. Couriers start and end their delivery tour from the warehouse specified in the city map. Couriers need 5 minutes to perform each delivery. Besides, the speed of all couriers is constant and equal to 15 kilometers per hour. The courier's goal is to complete all each deliveries of the tour efficiently and return to the warehouse while adhering to the time constraints of each delivery: <ul style="list-style-type: none"> If a courier reaches a delivery location before its time-window starts, they must wait for the beginning of the time-window. If they arrive after the end of the time-window, the tour is considered invalid. If a courier can't fulfill a delivery request due to time-window constraints, the delivery planner may be prompted to select another courier or, if no alternative is available, the delivery request may be rejected.
Course / Tour	<ul style="list-style-type: none"> Optimized route taken by a courier to complete a series of deliveries. A tour starts and ends at the warehouse, begins at 8 a.m., and involves visiting each delivery location within their respective time-windows. A courier is assigned to a tour. A tour is made of one courier, the delivery requests assigned to this tour, and the road segments composing the tour. A tour is designed by the application to be efficient and optimized, considering factors like travel speed and delivery time. If a courier arrives after the end of its time-window, the tour is considered invalid.
Delivery	Refer to the process of transporting packages from a known starting point to a delivery location within a city.

Term	Definition
Delivery instructions file	Text-based document generated by the application comprising a detailed set of instructions that guide the courier on the specific actions and routes they should take to successfully complete their assigned deliveries.
Delivery planner	<ul style="list-style-type: none"> • User of the application, responsible for assigning and managing delivery tasks to couriers. • They can interact with the application to allocate specific deliveries to a courier and provide them with their optimized tour for completing the assigned deliveries.
Delivery request	<ul style="list-style-type: none"> • Specific intersection at which a package has to be delivered. • It contains the time-window during which the delivery must occur (starting at 8 a.m. and divided into one-hour intervals).
Departure time	The time at which a courier leaves a delivery point.
Intersection / Node	<ul style="list-style-type: none"> • Geographical points on the city map representing two or more roads meeting or dead ends. • Each intersection is defined by its identifier, its latitude and longitude coordinates. These coordinates are used to accurately position and navigate through the city, ensuring efficient traversal between various locations.
Latitude	<ul style="list-style-type: none"> • Geographical coordinate that specifies a point's north-south position on the Earth's surface. • It is measured in degrees, with values ranging from -90° (representing the South Pole) to $+90^{\circ}$ (representing the North Pole).
Longitude	<ul style="list-style-type: none"> • Geographical coordinate that specifies a point's north-south position on the Earth's surface. • It is measured in degrees, ranging from -180° (representing the International Date Line) to $+180^{\circ}$ (representing the Prime Meridian).
Package	Object to be delivered by the courier.
Ride time	Duration it takes for a courier to perform all their deliveries.

Term	Definition
Roadmap	Output file obtained after saving a tour. This file can then be loaded again in the application to restore the tour.
Road segment	<ul style="list-style-type: none"> • Individual section of a road that connects two intersections. • Each road segment is characterized by an origin intersection, a destination intersection, a name, and a length in meters.
Step	Sequence of one or more road segments between two successive delivery locations of the courier's tour.
Time-window	<ul style="list-style-type: none"> • One hour time interval during which a specific delivery must occur. <ul style="list-style-type: none"> ◦ For instance: timeWindow = 10 means that the courier have to be in the delivery location between 10:00 and 10:55 • If a courier reaches a delivery location before its time-window starts, they must wait for the beginning of its time-window. If they arrive after the end of the time-window, the tour is considered invalid.
Waiting time	The time a courier must wait if they arrive at a delivery point before the beginning of its time window.
Warehouse	<ul style="list-style-type: none"> • Central location from which couriers begin and end their delivery tours. • It is represented by an intersection.
XML File	<ul style="list-style-type: none"> • 3 types of XML files : <ul style="list-style-type: none"> ◦ City map file : file containing information about the city map, including intersections, road segments, and warehouse location. It is used to generate a city map in the application. ◦ Tour file : file containing information about a tour previously saved, including the warehouse location and the delivery requests of this tour. ◦ RoadMap

Domain model

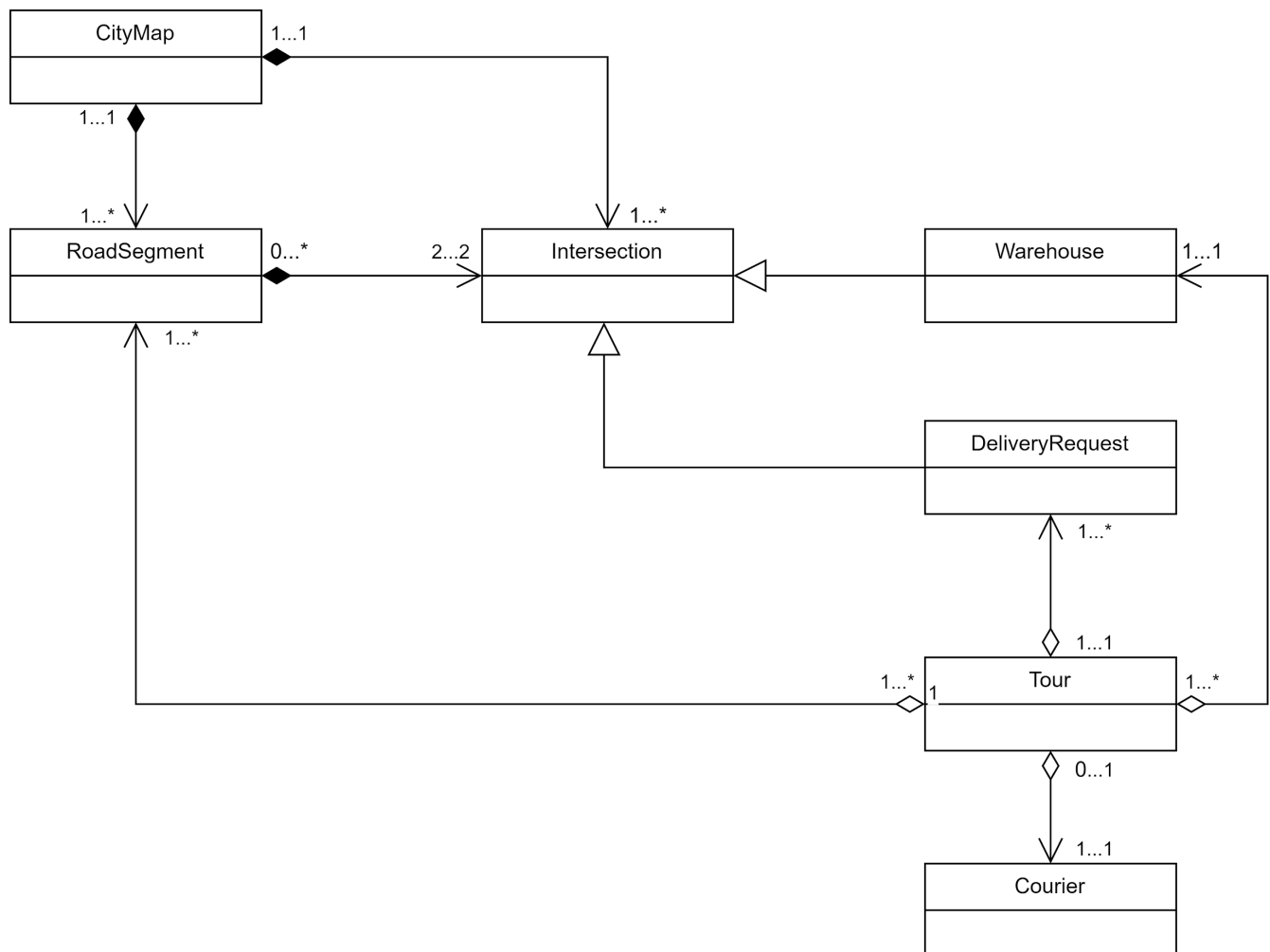


Figure 1. Domain model of the delivery application

Use case diagram



Figure 2. Use case diagram for the first iteration

Structured descriptions of the main use cases

1 : Load a city map

Description : Load into the application an XML city map file that has been chosen by the delivery planner.

Actors : Delivery planner, File System

Preconditions : The application is successfully launched and ready to receive the delivery planner input.

Postconditions : The city map is loaded into the application, providing the necessary geographical data for planning courier's tours.

Main scenario :

1. The delivery planner clicks on the "Load a city map" button.
2. The system opens a file dialog.
3. The delivery planner selects an XML city map file.
4. The system loads the chosen XML file.
5. The system draws the map corresponding to the data described in the XML file.

Extensions :

- 3a. The selected file is not an XML file :
 - The system displays the following error message "The selected file is not an XML File" and goes back to the main menu without loading a city map.
- 3b. The selected file is empty or erroneous :
 - The system displays the following error message "Invalid XML File" and goes back to the main menu without loading a city map.
- 1-3a. The delivery planner clicks on the "Cancel" button :
 - The system aborts its action and goes back to the main menu.

2 : Load a delivery requests file

Description : Load into the application an XML delivery requests file that has been chosen by the delivery planner.

Actors : Delivery planner, File System

Preconditions :

- The application is successfully launched and ready to receive the delivery planner input.
- A city map has been loaded into the application.

Postconditions : The delivery locations from the selected delivery requests file are displayed on the city map.

Main scenario :

1. The delivery planner clicks on the “Load delivery requests” button.
2. The system opens a choosing file window.
3. The delivery planner selects an XML format delivery file.
4. The system loads the chosen XML file.
5. The system adds the previously loaded delivery requests to the available delivery requests.

Extensions :

- 3a. The selected file is not an XML file :
 - The system displays the following error message “The selected file is not an XML File” and goes back to the main menu without loading a delivery requests file.
- 3b. The selected file is empty or erroneous :
 - The system displays the following error message “Invalid XML File” and goes back to the main menu without loading a delivery requests file.
- 3c. The delivery requests file has already been loaded :
 - The system erases the existing tour file and loads the new tour file chosen by the delivery planner.
- 3d. The delivery locations of the selected file don't match with the intersections of the actual city map :
 - The system displays the following error message “The delivery locations don't match the city map” and goes back to the main menu without loading a delivery requests file.
- 1-3a. The delivery planner clicks on the “Cancel” button :
 - The system aborts its action and goes back to the main menu.

3 : Create a delivery request

Description : Create a delivery request from the application by selecting an intersection and a time-window.

Actors : Delivery planner

Preconditions :

- The application is successfully launched and ready to receive the delivery planner input.
- A city map has been loaded into the application.

Postconditions : The delivery request is created and displayed on the city map.

Main scenario :

1. The delivery planner clicks on the “Create a delivery request” button.
2. The delivery planner selects a time-window.
3. The delivery planner selects an intersection on the city map by clicking on it.
4. The delivery planner specifies if it’s a warehouse or not.
5. The delivery planner clicks on the “Validate the created delivery request” button.
6. The system adds the created delivery request to the available delivery requests.

Extensions :

- 3a. The delivery planner doesn’t select an intersection :
 - The system displays the following error message “No intersection has been selected” and goes back to the main menu.
- 4a. The delivery planner doesn’t specify if it’s a warehouse :
 - The system displays the following error message “You need to specify if it is a warehouse or not” and goes back to the main menu.
- 4b. The delivery planner wants to create more than one warehouse :
 - The system displays the following error message “There is already a warehouse” and goes back to the main menu.

4 : Assign a delivery request to a tour

Description : Attribute a specific delivery request to the current tour, specifying the delivery location and associated time-window.

Actors : Delivery planner

Preconditions :

- The application is successfully launched and ready to receive the delivery planner input.
- A city map has been loaded into the application.
- At least one delivery request is available
- At least one courier is available.

Postconditions : The delivery request is successfully assigned to the tour.

Main scenario :

1. The delivery planner clicks on the “Assign a delivery request” button
2. The system displays a list of all available delivery requests
3. The delivery planner select delivery request(s)
4. The system assign delivery requests to the tour

Extensions :

- 1a. There are no available couriers :
 - The system displays the following message “No courier available” and goes back to the main menu.
- 2a. The delivery location or time-window information is incomplete or invalid :
 - The system displays the following error message “Invalid delivery request details” and prompts the delivery planner to provide valid details.
 - The system goes back to step n°1.
- 1-3a. The delivery planner clicks on the “Cancel” button :
 - The system aborts its action and goes back to the main menu.

5 : Determine an optimized tour

Description : Compute the best possible tour corresponding to the current delivery requests associated with the courier, and then display the tour on the city map.

Actors : Delivery planner

Preconditions :

- The application is successfully launched and ready to receive the delivery planner input.
- A city map has been loaded into the application.
- A delivery request has been assigned to the selected courier.

Postconditions : The optimized tour is displayed on the city map.

Main scenario :

1. The user clicks on the “Compute the best tour” button.
2. The system computes the best tour possible.
3. The system draws the tour on the city map.

Extensions :

- 1a. The delivery planner clicks on the “Cancel” button :
 - The application aborts its action and goes back to the main menu
- 2a. The system doesn't find a tour :
 - The system displays the following error message “Unable to find an optimized tour” and goes back to the main menu.

6 : Generate Road Map

Description : For each Tour, generate a roadmap as an HTML file named after the courier responsible for the tour.

Actors : Delivery planner

Preconditions :

- The application is successfully launched and ready to receive the delivery planner input.
- A city map has been loaded into the application.
- A delivery request has been assigned to the selected courier.
- At least one tour has been successfully computed

Postconditions : For each existing tour, a roadmap is generated.

Main scenario :

1. The user clicks on the “Generate Road Map” button.
2. The system asks for the location to save the file.
3. The system creates the files in the designated location.

Extensions :

- 1a. The delivery planner clicks on the “Cancel” button :
 - The application aborts its action and goes back to the main menu

7 : Manage courier

Description : Ability to add or remove a courier.

Actors : Delivery planner

Preconditions :

- The application is successfully launched and ready to receive the delivery planner input.

Postconditions : A courier is created, modified, and/or removed

Main scenario :

1. The user clicks on the “Manage courier” button.
2. The system opens an interface allowing for courier creation, modification and deletion.
 - a. The user clicks on the “Create a courier” button, input a valid name, and validates the creation.
 - b. The user selects an existing courier, then clicks on the “Modify courier” button, and validates the modification.
 - c. The user selects an existing courier, then clicks on the “Remove courier” button, and validates the deletion.

State chart diagram

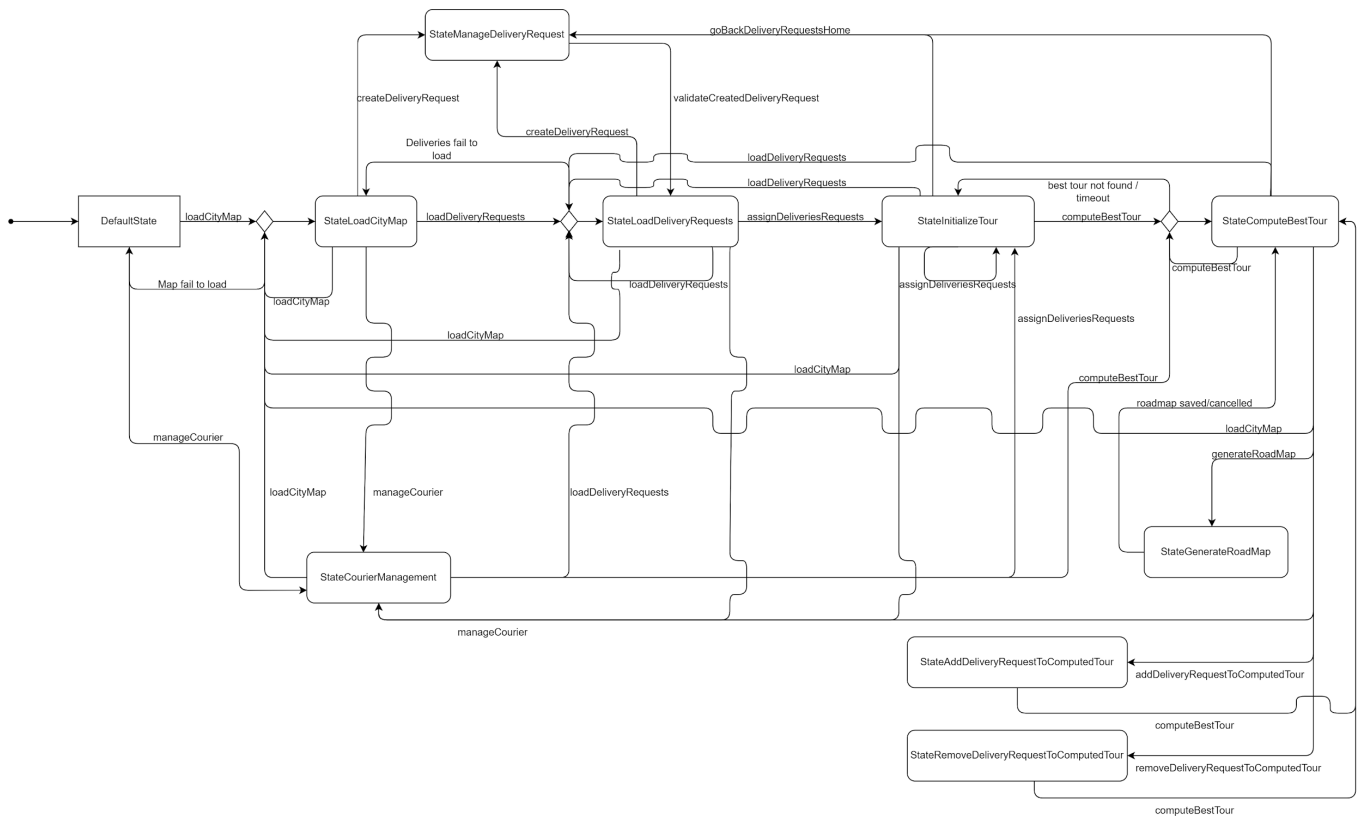


Figure 3. State-Transition diagram

MVC Architecture and design patterns

MVC

A Model View Controller (MVC) approach has been chosen, indeed, the separation of the application in 3 different concerns provides a long-term maintainability, scalability, and flexibility of the application. In the packages, the classes are strongly cohesive and each package (except the controller) are weakly coupled.

Here are the 3 concerns and why their choice is judicious :

1. Model : In such a delivery app the data layer is responsible for the definition, the management, and the manipulation of all the necessary entities.
2. View : Defines the user interface and the presentation of the data. It also involves the way the user will manipulate the data.
3. Controller : Basically, its role is to tune the model to the view.

The MVC architecture also promotes code reusability. For instance, the data model is reused across every concern of the application. In extension the application is easier to evolve. About upgradability, a well segmented code permits the creation of tests which attest the non-regression of the application.

Design patterns

Commands

The Command design pattern is a behavioral design pattern that encapsulates a request as an object, thereby allowing users to parameterize clients with different requests, queue requests, and support undoable operations. It promotes loose coupling between the sender of a request and its receiver.

State

The State design pattern state is used to change the application context, that allows us to navigate in the different allowed modes of the application. The design pattern observer is used to use the IHM with a mouse and to show selected components, it avoids the components to ask for the state everytime so it's better for performances when there are many objects.

Observer

The Observer design pattern is a behavioral design pattern that defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. For instance, the bottom left panel of the app which shows the legend of the city map automatically changes its form according to the state of the city map.

Class and package diagrams

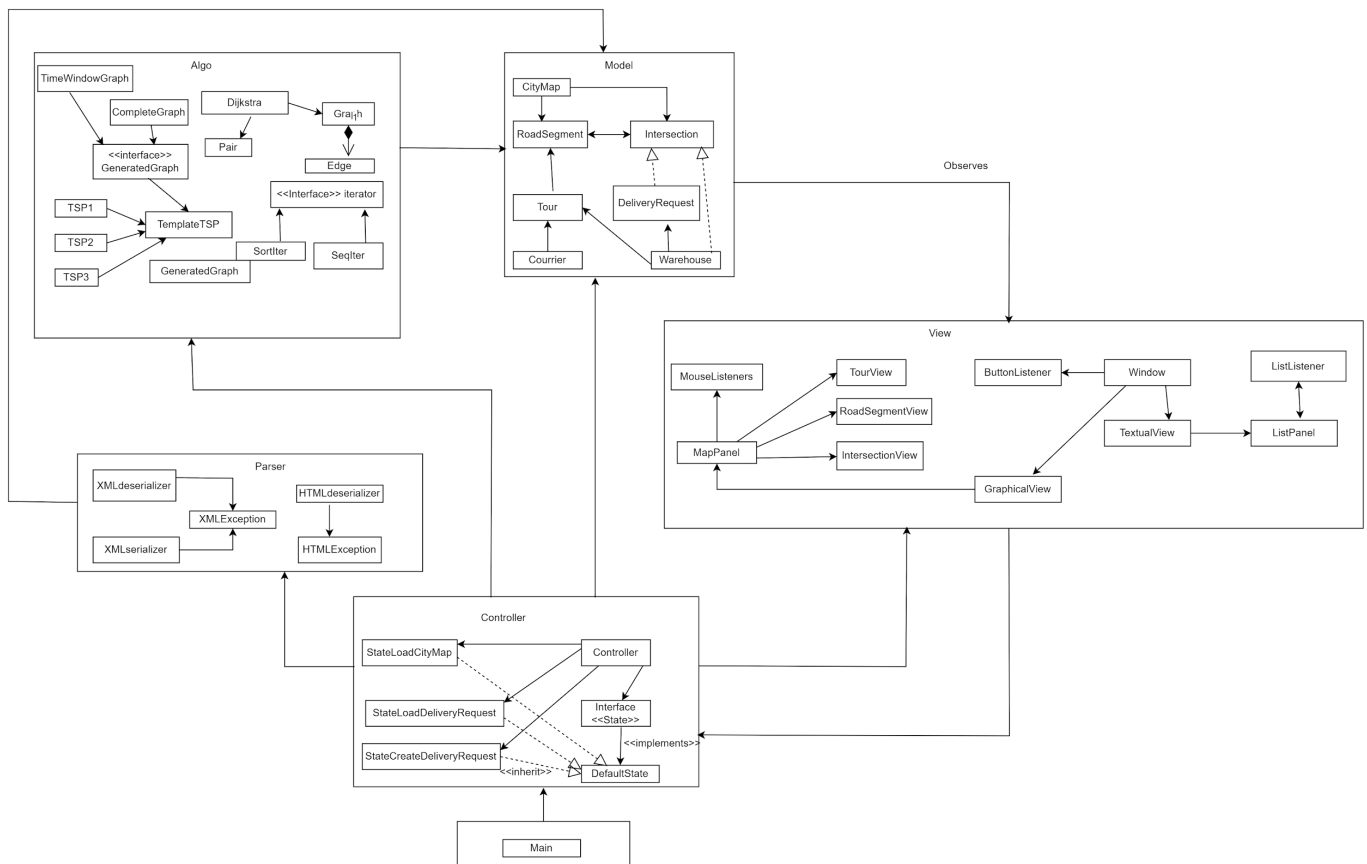


Figure 4. Package diagram

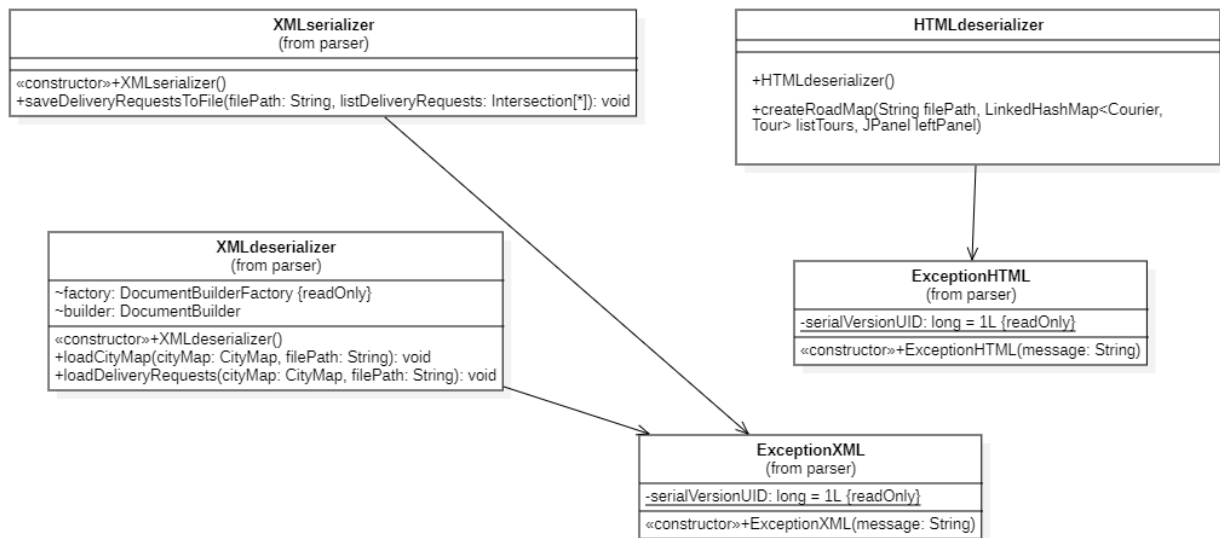


Figure 5. Class diagram of the parser package

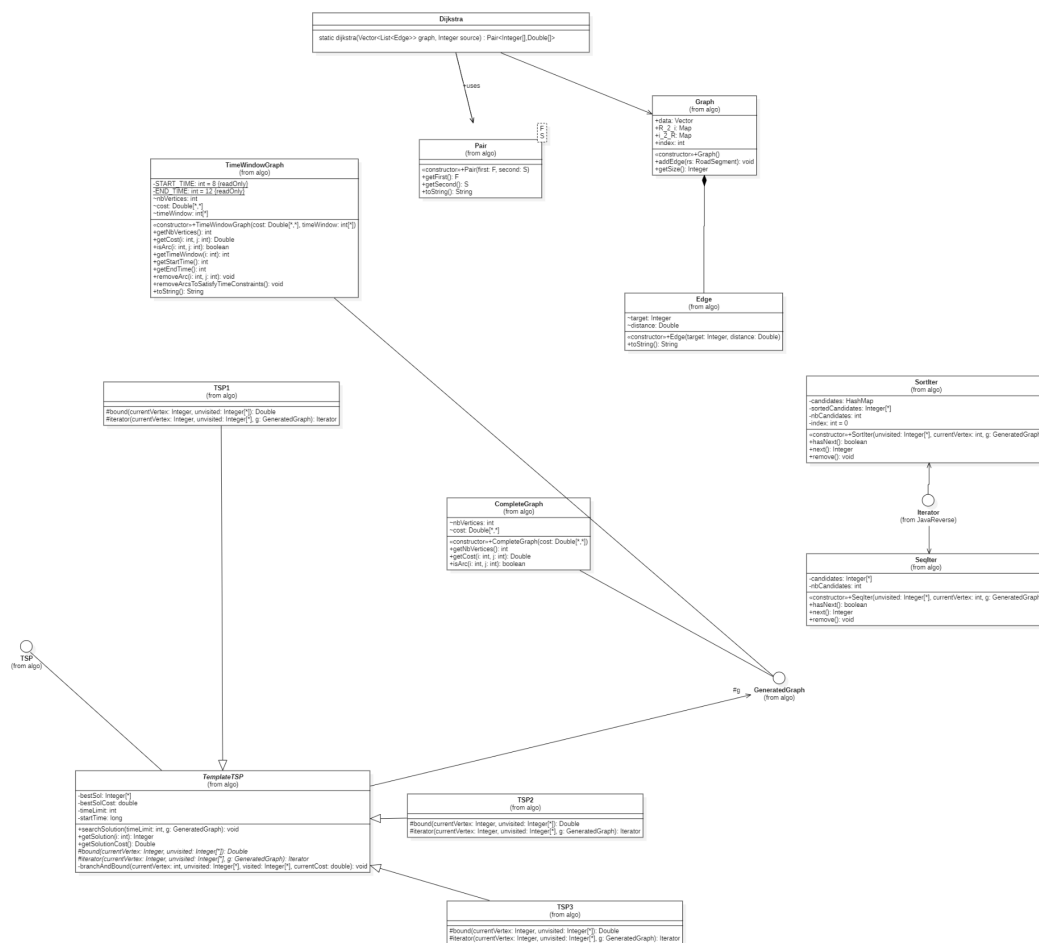


Figure 6. Class diagram of the algo package

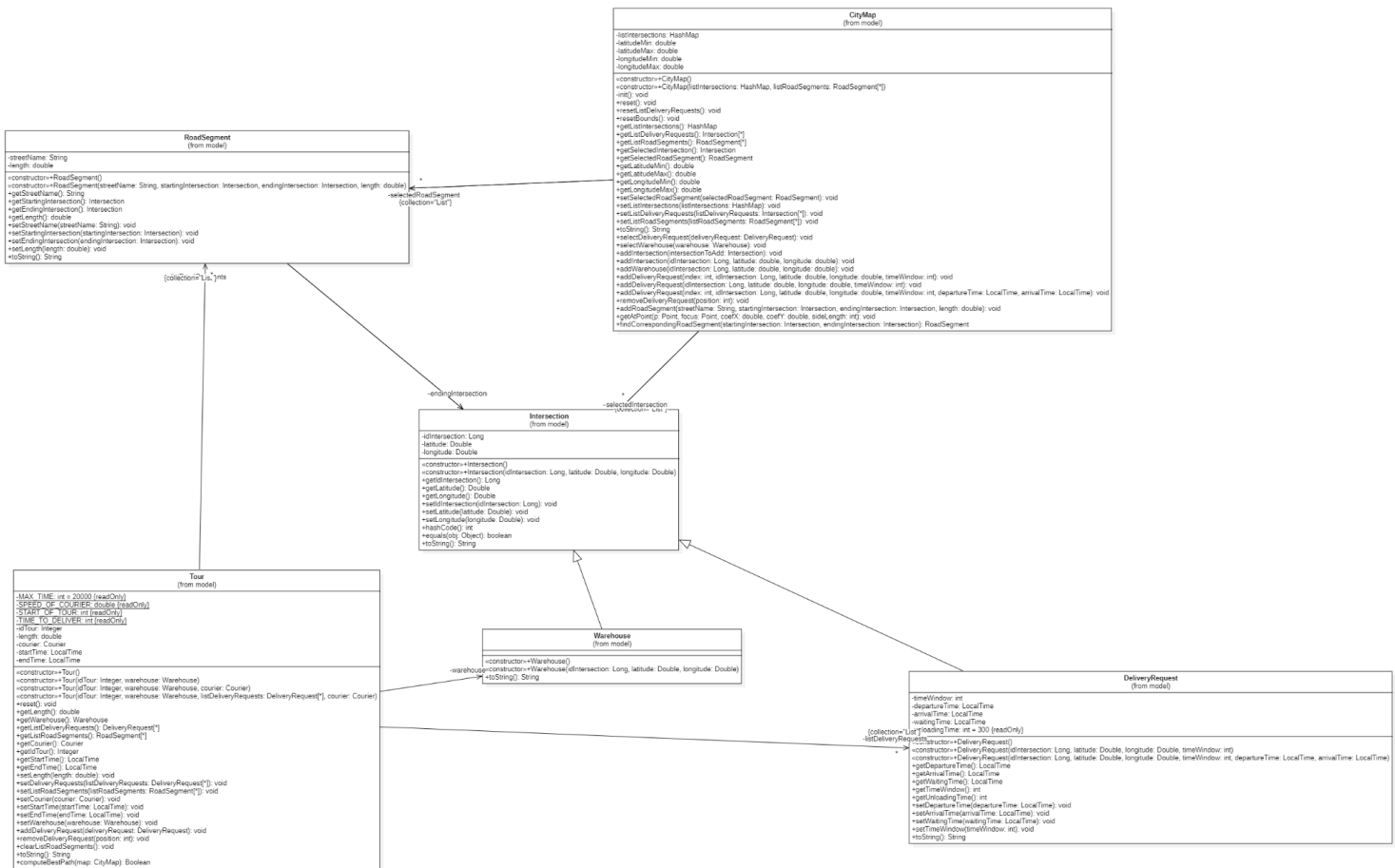


Figure 7. Class diagram of the model package



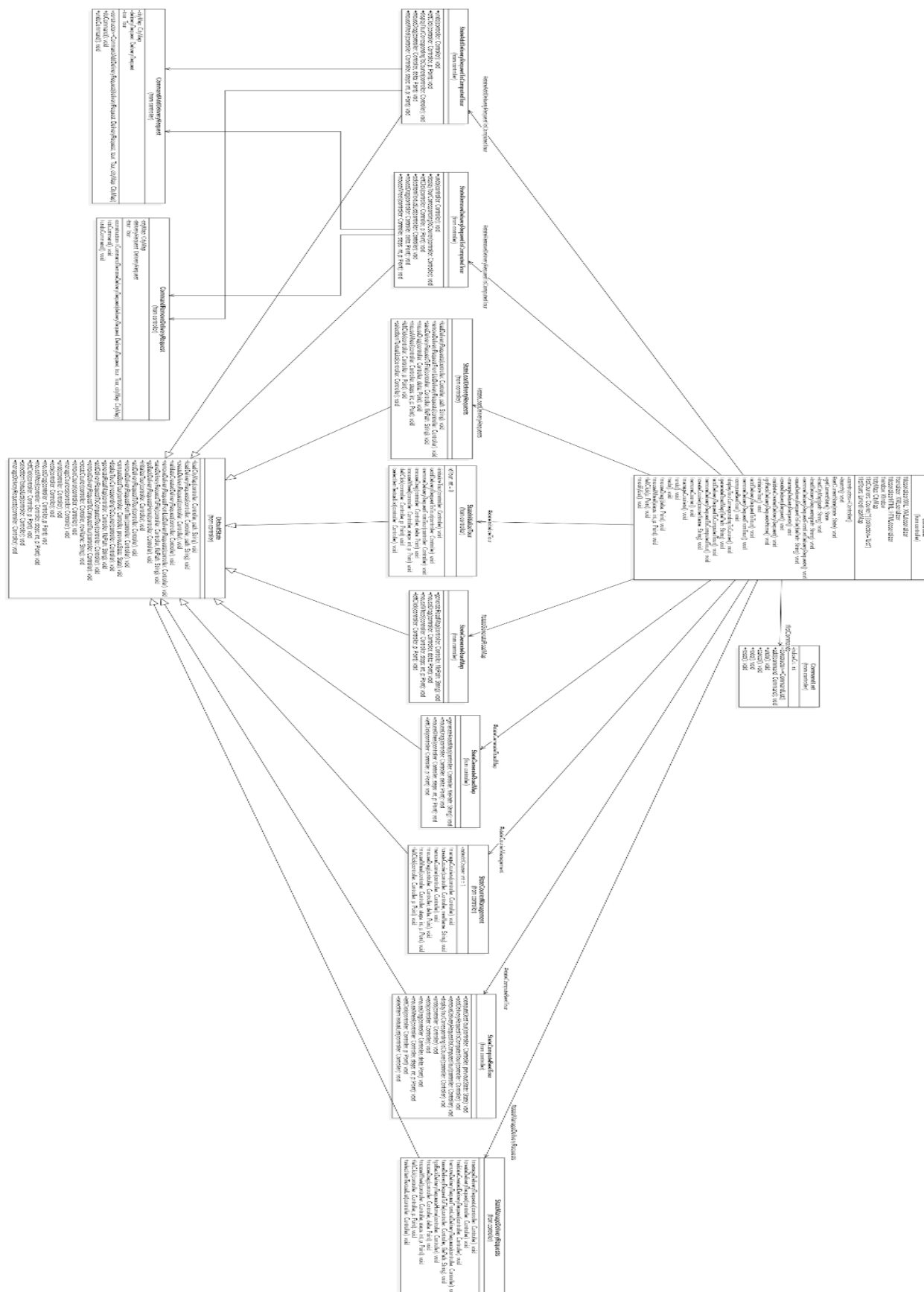


Figure 9. Class diagram of the controller package

Differences between these diagrams and first iteration diagrams

On the one hand, we have created an algo package containing all the classes needed to calculate the best possible path for a tour. Indeed, this package is made up of classes representing a graph on which the Dijkstra and TSP algorithms can be applied.

On the other hand, we added the following classes :

- In the parser package (old xml package) :
 - HTMLdeserializer : le This class provides methods for deserializing a tour into a road map HTML page.
 - ExceptionXML : This class represents an exception specific to HTML handling.
- In the controller package :
 - StateManageDeliveryRequests : This class represents the state for creating, removing and saving delivery requests.
 - StateInitializeTour : This class represents the state when we assign a delivery request to a tour.
 - StateComputeBestTour : This class represents the state when we compute the best tour.
 - StateAddDeliveryRequestToComputedTour : This class represents the state when a delivery request is being added to a tour that has already been computed at least once.
 - StateRemoveDeliveryRequestToComputedTour : This class represents the state when a delivery request is being removed from a tour that has already been computed at least once.
 - StateGenerateRoadMap : This class represents the state when we generate a road map.
- In the view package :
 - IntersectionView : This class contains methods to draw an intersection on the map.
 - RoadSegmentView : This class contains methods to draw a road segment on the map.
 - TourView : This class contains methods to draw a list of tours on the map.

Planning

First iteration

Expectation

First iteration									
Tasks \ Names		Elie	Hélène	Billy	Nicolas	An Jun	Seynabo u	Gaspard	Total (Hour/pe rson)
1st sess ion	Preparing the work tools	1					1	1	3
	Use Case Diagrams			4	4	1			9
	Glossary	1	1			1	3	3	9
	Domain model	1	2			2			5
	Planning the 1st iteration	1	1						2
2nd sess ion	Brief description of the main success scenario of all identified use cases	1,5	1,5	1,5	1,5	0,5	0,5	0,5	7,5
	Structured description of all the use cases	1,5	1,5	1,5	1,5	0,5	0,5	0,5	7,5
	Class diagrams	0,5	0,5	0,5	0,5	1,5	1,5	1,5	6,5
	Package diagrams	0,5	0,5	0,5	0,5	1,5	1,5	1,5	6,5
3rd & 4th sess ion	State diagram				7	7			14
	Tests			8					8
	Algorithms		7					8	15
	View	8					8		16
	Deliverable		1		1	1			3
Total hours/person		16	16	16	16	16	16	16	112

Reality

First iteration									
Tasks \ Names		Elie	Hélène	Billy	Nicolas	An Jun	Seynabo u	Gaspard	Total (hour/task)
1st session	Preparing the work tools	1					1	1	3
	Use Case Diagrams			4	4	1			9
	Glossary	1	1			1	3	3	9
	Domain model	1	2			2			5
	Planning the 1st iteration	1	1						2
2nd session	Brief description of the main success scenario of all identified use cases	0,5							0,5
	Structured description of all the use cases	0,5		1,25	1	1	0,5		4,25
	Detailed sequence diagram					3		3	6
	Class diagrams	0,5			1	1	3	4	9,5
	Package diagrams	0,5			1	1	3		5,5
3rd and 4th session	State diagram				4	4	0,5		8,5
	Tests			24					24
	Algorithms		8		4	1		10	23
	View	9					5		14
	Deliverable	1	2		1	1			5
Total (hour/person)		16	14	29,25	16	16	16	21	128,25

Second iteration

Expectation

Second iteration									
Tasks \ Names		Elie	Hélène	Billy	Nicolas	An Jun	Seynabou	Gaspard	Total (Hour/person)
5th session	Review code + refactoring	1,5	1,5				1,5	1,5	6
	Implementation of the computing of a tour							4	4
	Tests of tour			5					5
	View of a tour (map)	3					3		6
	Implementation of the time window constraints		2						2
	Generate a roadmap (textual)	1					1		2
	Assign a delivery request to a tour (view)	2					2		4
Total hours/person		7,5	3,5	5	0	0	7,5	5,5	23

Reality

Second iteration									
Tasks \ Names		Elie	Hélène	Billy	Nicolas	An Jun	Seynabou	Gaspard	Total (Hour/person)
5th session	Review code + refactoring	1,5	1,5				1,5	5,5	10
	Implementation of the computing of a tour							24	24
	Tests of tour			4					4
	View of a tour (map)	3							3
	Implementation of the time window constraints		0,5						0,5
	Generate a roadmap (textual)		7				2		9
	Assign a delivery request to a tour (view)	2							2
Total hours/person		6,5	9	4	0	0	3,5	29,5	52,5

Third iteration

Expectation

Third iteration									
Tasks \ Names		Elie	H��						

Reality

Third iteration									
Tasks \ Names		Elie	Hélène	Billy	Nicolas	An Jun	Seynabou	Gaspard	Total (Hour/person)
6th session	Adding legend when opening a file	1					1		2
	Fix zoom	1							1
	Refuse to open a file if any problem (delivery point not existing)	1							1
	Implementation of the functionality to manage the couriers	1				6	2		9
	Tests on managing the couriers	2		4			2		8
	Define use cases and documentation					3			3
	Managing the couriers (view)	2					2		4
	State-transition diagram					3			
	Improve TSP		4						4
Total hours/person		8	4	4	0	12	7	0	32

Fourth iteration

Expectation

Fourth iteration									
Tasks \ Names		Elie	Hélène	Billy	Nicolas	An Jun	Seynabou	Gaspard	Total (Hour/person)
7th & 8th sessions	Improve UX		6				2	4	12
	Improve UI						2		2
	Implementation of add/suppress delivery request (controller pattern)	4				4			8
	Test/improve the robustness			8		4	4		16
	Precise information in the roadmap		2		4				6
	Show different path possibilities	4			4			4	12
Total hours/person		8	8	8	8	8	8	8	56

Reality

Fourth iteration									
Tasks \ Names		Elie	Hélène	Billy	Nicolas	An Jun	Seynabou	Gaspard	Total (Hour/person)
7th & 8th sessions	Improve UX	2	8				3		13
	Improve UI	2					3		5
	Implementation of add/suppress delivery request (controller pattern)	3							3
	Test the robustness	2				2	3	4	11
	State-transition diagram					8			
	Precise information in the roadmap	2					2		4
	Show different path possibilities	5					4		9
Total hours/person		16	8	0	0	10	15	4	45

Total

Expectation

Tasks \ Names	Elie	Hélène	Billy	Nicolas	An Jun	Seynabou	Gaspard	Total (Hour/person)
Total hours/person	36	31,5	33	28	28	36	29,5	216

Reality

Tasks \ Names	Elie	Hélène	Billy	Nicolas	An Jun	Seynabou	Gaspard	Total (Hour/person)
Total hours/person	46,5	35	37,25	16	38	41,5	54,5	257,75

Test Report

Tests were made in Java, with JUnit on Eclipse.

For the test, we use 12 classes : two for the package algo, three for the package parser, and the seven others for the package model. 141 tests were made covering 71,7% of the project. As you can see :

Element		Coverage
AGILE		71,7 %
src		71,7 %
> model		97,7 %
> tests		95,1 %
> algo		83,0 %
> parser		75,3 %
> view		74,3 %
> controller		18,0 %
> (default package)		0,0 %

Figure 10. Package test reports

Model

CityMapTest

Tests cover 95,2% of instructions of this class, from constructors to more complex functions like *resetBound*. Adding an intersection with an arrival time and a departure time doesn't work correctly. Moreover there is a problem with the rise of an exception, the wrong exception is raised but the controller manages it.

CourierTest

Tests cover 99% of instructions of this class, there is no problem with the corresponding class.

DeliveryRequestTest

Tests cover 100% of instructions of this class, there is no problem with the corresponding class.

RoadSegmentTest

Tests cover 100% of instructions of this class, there is no problem with the corresponding class.

IntersectionTest

Tests cover 100% of instructions of this class, there is no problem with the corresponding class.

TourTest

Tests cover 98,9% of instructions of this class, there is no problem with the corresponding class.

WareHouseTest

Tests cover 100% of instructions of this class, there is no problem with the corresponding class.

















▼  model		97,7 %
>  DeliveryRequest.java		100,0 %
>  Intersection.java		100,0 %
>  RoadSegment.java		100,0 %
>  Warehouse.java		100,0 %
>  Courier.java		99,0 %
>  Tour.java		98,9 %
>  CityMap.java		95,2 %

Figure 11. Model test reports

Algo

GraphTest & DijkstraTest

Cover 83% of instructions of the package algo, there is no problem with the corresponding class. Some methods are not completely tested.

▼  algo		83,0 %
>  Graph.java		100,0 %
>  TSP3.java		100,0 %
>  SortIter.java		99,3 %
>  SeqIter.java		98,2 %
>  Dijkstra.java		97,4 %
>  TemplateTSP.java		95,9 %
>  CompleteGraph.java		85,7 %
>  TimeWindowGraph.jav		67,4 %
>  Pair.java		65,2 %
>  Edge.java		52,9 %
>  TSP1.java		0,0 %
>  TSP2.java		0,0 %

Figure 12. Algo test reports

Parser

XMLdeserializerTest

Tests cover 69% of instructions of this class, there is no problem with the corresponding class.

XMLserializerTest

Tests cover 100% of instructions of this class, there is no problem with the corresponding class.

HTMLdeserializerTest

Tests cover 78,5% of instructions of this class, there is no problem with the corresponding class.







~	parser		75,3 %
>	ExceptionXML.java		100,0 %
>	XMLserializer.java		100,0 %
>	HTMLdeserializer.java		78,5 %
>	XMLdeserializer.java		69,0 %
>	ExceptionHTML.java		0,0 %

Figure 13. Parser test reports

The controller did the testing of complex methods like the computing of a tour or creating a road map. At first there was a *ControllerTest* class, but testing can be made with the GUI (import a wrong map, try to raise errors). In fact we make a lot of scenarios, so the testing part of this class was made by testing the robustness. For some methods, even if the scenario was clear the testing was complex, because initializations requisite for them.

Technical review

- What should have been improved ? :
 - When one modifies a delivery request in a tour, we should have improved the calculated best path instead of recalculating it.
 - We should have reduced the time before programming.
 - We should have been more efficient in the time before programming by working all at the same time instead of working by groups of only two, and then correcting each other at the end.
 - We should have taken much more time to learn Git in order to reduce bad usage during the project (having files out of the git and coding into them and then replacing the files in the git after that).
 - We should have used a branching strategy.
 - We should have committed with better names of commit.
 - We should have taken time to put in place CI/CD.
 - We should have done more tests.

The rest of the technical aspect is quite good.

Human review

- What should have been improved ? :
 - We should have said what we will be doing before doing it to avoid working on the same thing at the same time and wasting time.
 - We should have communicated better and had meetings to explain what we have done more often.
 - We should have stressed less and be more confident about our abilities.
 - We should have discussed more about what we will (choosing it instead of having to do it right now because we are running out of time).
 - We should have attributed another person to the tests some times.
- What is good to know ? :
 - We found solutions to our problems very fast.
 - We didn't have big problems because we had a good base before going to the next step.
 - We could discuss about anything
 - We manage to dispatch the tasks between each other
 - We managed to always help each other.

Environmental and Social issues

For the environmental part, the most impactful is the lifecycle of the hardware products. Indeed, some components are made of rare-earth metals, which are by name “rare” since there is a limited amount available, that forces us to dig deep in earth resulting in a lot of long term pollution, and usage of energy. Unfortunately we are currently unable to recycle these materials from hardware since they are not pure anymore (mixed with other materials). This results in a lot of waste thrown in some poor countries like Ghana for instance. It is also a societal issue because work conditions in rare-earth metals extraction are deplorable (child employment, hazard), since it happens in low-regulated countries. For our application, we only use one computer for the delivery planner. We can also advise him to repair his laptop instead of buying a new one if there is any problem and take care of it to ensure it will last longer.

Furthermore, when we use computers in our everyday life, it consumes energy, but it also produces CO₂ emissions. Indeed, everytime we need external data, your computer has to request it from a network. It results in a flow of information that emits CO₂. Some data like video for instance are heavier than others, thus this flow will emit more CO₂ than text. Data is stored in data centers, they generally use a lot of energy to answer requests in a reasonable time, and cool the infrastructure. Since the largest amount of emissions result from the travel of data, it is preferable to place data centers near to need. In our application we are not linked to a network, we only use local files thus this impact is negligible. Nevertheless there exists an impact of our running application. To limit this impact there exist good practices of eco-conception like having an advanced consideration of the use cases and client needs (not to develop useless functionalities), clean code (to be able to easily maintain it). For our project, we tried to implement these good practices. Nonetheless, it was sometimes hard due to deadlines or lack of knowledge.

For the social part, there are two major issues: accessibility and privacy. Accessibility is to ensure applications or websites are accessible for disabled people. Indeed, there exists a digital divide: some persons are excluded from the digital world because it is not adapted to them. There is a list of good practices and standards like the WCAG (Web Content Accessibility Guidelines) to respect. For example, you have to provide contrasted texts for visual disabilities, a structured text version for screen readers to help blind people or keyboard navigation for reduced mobility impairments. For our project we were unfortunately not able to ensure our application to be accessible since we did not have time to design the interfaces well and a lack of knowledge concerning technical implementation of accessibility for a local application. Ensuring privacy, especially for sensitive data is also important. For our project we did not manipulate sensitive data, thus we are not concerned by this issue.

Personal review and commentary

ANDRIANARISOLO Elie

Being part of this app project was a mix of good and tough moments. Figuring out how to use algorithms like Dijkstra's and the Traveling Salesman Problem (TSP) to find the shortest paths in the graph was really interesting. Collaborating within a team environment while adhering to Agile methodologies fostered a synchronized workflow, facilitating seamless progress toward project milestones. My familiarity and learning curve with Git expanded significantly, enhancing my proficiency in managing collaborative code bases. The whole experience was a big learning curve, boosting my skills and how I tackle problems. Nevertheless, there wasn't enough time to dig deeper into things, and we faced some issues with Git that slowed us down. These problems pushed us to get creative in finding solutions and handle the project's ups and downs.

DOS SANTOS Hélène

I was afraid of this project since it is a whole application to implement with 5 people, but it made me realize it was possible. Nevertheless, I felt like I would have needed more time to think through the conception of the application. I felt like when we started implementing we were not certain of how one is articulated with the other, especially on functionalities we were not personally working on. It was satisfying to have a long project because we better understand the whole process, but I am upset we did not have enough time to dig deeper. I found the lecture about environmental and social issues really interesting, but it is a shame we did not really manage to implement these good practices.

SARR Seynabou

I found working in a team really enjoyable; the camaraderie and collective effort brought a dynamic energy to the project. However, I didn't quite resonate with the Agile method. I felt it didn't entirely suit our project's needs, making the workflow feel a bit disjointed at times. Another challenge was the project's lack of guidance, which left certain aspects open-ended and made it harder to steer in the right direction. Nonetheless, I appreciated the opportunity to enhance my Java skills. Yet, the project's tight timeline created a significant hurdle, limiting the depth we could explore and preventing us from fully realizing some innovative ideas.

SERPINET Gaspard

I really enjoyed collaborating within a team environment while implementing the Agile methods for our project. It allowed us to stay adaptive and focused, fostering effective communication and iterative development. Sharing my knowledge on Git was a highlight. Moreover, upgrading my Java skills was incredibly fulfilling. I found it rewarding to enhance my abilities in a language crucial to our project's success. However, the challenge arose from the limited time allocated for the project, which unfortunately constrained our ability to delve deeper into certain aspects and potentially explore more innovative solutions.

THAIZE Nicolas

Co-developing such a software with 5 other people seemed to me quite difficult considering that i've never collaborated with other people during development. Having a role of support on every plan of the project was enjoyable because it allowed me to help and see everything that has been done so far. I think the part where we struggled the most was implementing each "bricks" together to make a great "wall", our software. I think we could have made more efforts to make our collaboration more agile and less siloed. Many times the group was asking what people were doing because of a lack of communication. That being said, the PLD was really enjoyable. Trying to find technical and social solutions to make everything work was quite a challenge but has been a trainer.

TONG An Jun

The AGILE PLD is a relevant opportunity to put into practice the concepts covered in 3rd year on the handling of graphs and shortest path algorithms. The PLD is also an opportunity to familiarize ourselves with the organization of AGILE teamwork and its implications. The use of a versioning tool such as Git becomes indispensable for synchronous and asynchronous collaboration. There's a certain amount of freedom when it comes to using the development environment, frameworks and IT technologies, even if the whole thing is fairly tightly controlled.

On the other hand, one thing I don't like about PLD may be the way in which the division of tasks between the two iterations is organized. For me, the crux of the problem isn't necessarily a lack of time - although some of my team-mates and comrades have mentioned this - but rather an uneven distribution of tasks, resulting on the one hand in the first iteration seeing its objectives met well before the deadline, and having to get ahead of the subsequent iterations, and on the other hand, a much heavier workload for the iteration(s) following the first, due to the more complex and numerous functionalities to be added, making it particularly difficult to divide up the work within the group, but also to fragment the last 4 sessions into several coherent iterations.

Of course, it was an enriching experience, but I firmly believe that a more thoughtful organization could significantly enhance the value and contribution of the PLD AGILE.

VILLEROY Billy

I would like to say a lot of things... At first I was scared because I have less skills than my comrades. After the end of the project, I am disappointed in myself. I want to be helpful but I don't think it is the case. In this project my role was tester, I don't think I really helped my group because they did really few errors. Moreover, it was difficult because each time and iteration, there were a lot of instructions to review and test (more than 10000). Next time I will call for some help because I couldn't achieve all the tests I wanted to do. But at the end of the day, it is really great to see what the group has done. The subdivisions of tasks made the work easier. Thank you everyone !