# PLD Agile

**Duration of the PLD :** 8 sessions of 4 hours each

**Teachers :** E. Egyed-Zsigmond, F. Laforest, K.Barrere, P. Lachat, co-authored with C. Solnon

## 1 Description of the Application

You must design and implement an application for optimising delivery tours in cities. For more sustainable cities, these deliveries are done with bicycles.

When launching the application, the user loads an XML file which describes a city map. This file gives the list of all intersections (such that each intersection has a latitude and a longitude) and the list of all road segments (such that each road segment has an origin intersection, a destination intersection, a name and a length in meters). The XML file also specifies the address of the warehouse, from which couriers start their tours. XML files are available on Moodle.

When a map is loaded, the application displays it. By default, the number of couriers is set to 1, and the user may change this number. Initially, there is no delivery request. When the user receives a new delivery request (by e-mail, SMS, phone, etc), he selects a courier, the location of the pickup and the location of the delivery. Each time a new request is entered, the system computes the best possible tour corresponding to the current requests associated with the selected courier. This tour must start from the warehouse at 8 a.m., visit each pickup and delivery, and return back to the warehouse.

We assume that the travel speed of all couriers is constant and equal to 15 kilometres per hour. We also assume that the time needed to perform each pickup and delivery is equal to five minutes. The best tour is the tour that minimises the arrival time on the warehouse.

If there is no tour that satisfies all time-window constraints, then the application asks the user to select another courier ; if there is no other courier, then the delivery request is rejected. The application displays the tour of each courier on the map ; it also displays for each pickup and each delivery, the address, the arrival time and the departure time. The user may save the current tours in a file, and he may also restore a set of tours from a file.

## 2 Organisation

The project is done by groups of five to seven students, using an iterative and agile development process. The project will be composed of at least three iterations. The first iteration will be 2 sessions long. For the next 6 sessions, you may choose to do between two and four iterations.

**First iterations :** This iteration corresponds to the USDP inception phase. The goal is to identify the main use cases, design a first architecture of your application, and implement very few use cases in order to have a first minimal but operational software (MVP minimal viable product). Deliverables that are required at the end of the first iteration are :

— Glossary
— Persona and use case diagram with a rapid description of each use case
— Detailed description of the MVP implemented and delivered use cases
— Package and class diagrams
— Sequence diagrams for the delivered use cases, if relevant
— Real planning of the iteration : for each activity, duration and team member in charge
— Design documents : architectural choices and used design patterns if relevant
— Documented code of the MVP with its tests (provide access to your git with a nice README ?)

Prepare a demo of the MVP.

**Next iterations :** At the beginning of each iteration, tell your client which features will be delivered at the end of the iteration.

At the end of each iteration, provide an extension of the former delivered elements :
- Details of newly implemented use cases are added to the previous ones
- Update of package and class diagrams (reverse-engineering ?) : explain the updates in comparison to the previous version
- Update of existing and new sequence diagrams
- Updated code with its documentations and tests
- Expected and real planning with explanations of the differences
- Iteration retrospective
- etc.

**Last iteration :** At the last iteration, provide an extension of the former delivered elements as before, and add :
- Tests coverage report
- Technical and human feedback on the project

**Development environment :** We recommend using Java for implementing the application. If you wish to use another object oriented language, you must first discuss your choice with us.

You must use an IDE and tools for automating unit tests, for evaluating test coverage, for controlling versions, and for automatically generating online code documentation. You may use any IDE but it must be the same for all the members of your team.

You may use some plug-ins like ObjectAid[1] for reverse engineering diagrams from code, and JUnit[2] for unit tests. You may use the demo version of StarUML[3] to draw UML diagrams. The online documentation of your code will be generated with JavaDoc, and you will apply the style guide of Oracle[4] To compute a tour, you may use the Java code available on Moodle. The class TSP1 implements the most basic (and naive) variant of algorithms studied in AAIA. You may implement more efficient variants by overriding methods bound and iterator. You may also implement another solving approach (dynamic programming or tabu search, for example), or use open source libraries (provided that you credit authors of these libraries !).

1. https ://www3.objectaid.com/
2. https ://junit.org/
3. https ://staruml.io/
4. http ://www.oracle.com/technetwork/java/codeconventions-150003.pdf