Names: Santiago Zúñiga García (A00352139), Mateo Ramírez Rodríguez (A00351422).

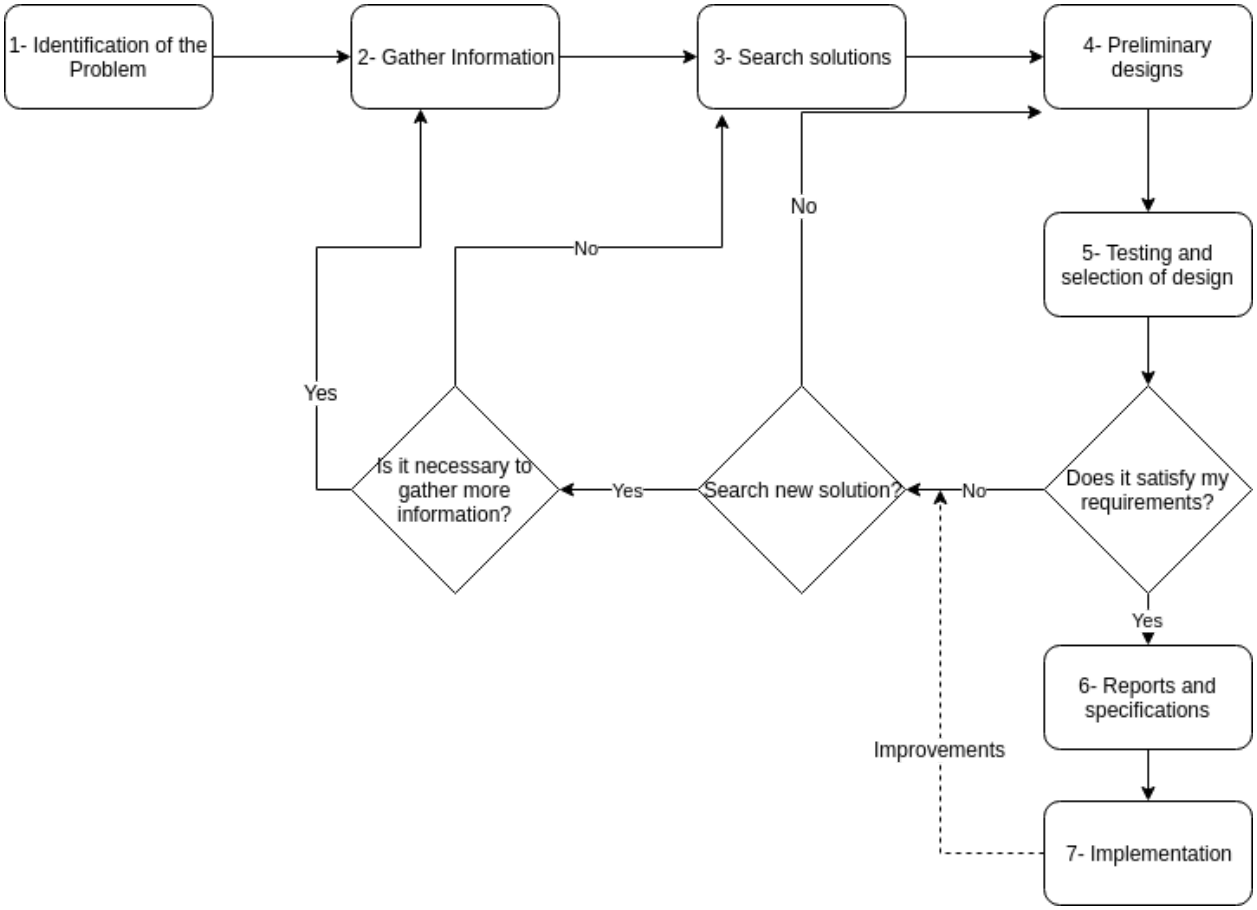
Problematic Context

Berliner Verkehrsbetriebe is the main transport company in Berlin, the capital city of Germany. It manages the U-Bahn (metro), tram, bus, ferry networks. And together with the S-Bahn make up all of Berlin's public transport. They require an application that unites the S-Bahn and the U-Bahn, additionally it must be able to calculate routes between the numerous stations in Berlin.

Development of the Solution

In order to resolve the before announced situation, the engineering method will be used to design and develop following a systematic approach relevant to the problematic context.

Based on Paul Wright's "Introduction to Engineering" description of the engineering method, the following flowchart has been defined; which will be carefully followed throughout the development of the solution.



Step 1: Identification of the problem

The objective is to recognize concretely the necessities of the problematic context and it's symptoms and conditions as well.

Identification of necessities and symptoms

- The users of the public transport service require to plan their daily routes.
- The solution to the problem must be efficient so that the service may be delivered to the utmost quantity of users with minimal resource consumption.

Definition of the problem

The companies Berliner Verkehrsbetriebe and S-Bahn require the development of a module of software that allow its users to plan their routes based on time and transfers.

Step 2: Gathering Information

With the objective of obtaining absolute clarity of the concepts involved, some research about the definitions of the terms relevant to the problematic context is necessary.

Definitions

Sources:

www.lexico.com

https://en.wikipedia.org/

https://www.dictionary.com/

Route

According to Lexico (by Oxford) a route is "A way or course taken in getting from a starting point to a destination." and also "The line of a road, path, railway, etc.".

Path

In the mathematical field of graph theory, a path graph or linear graph is a graph whose vertices can be listed in the order v1, v2, …, vn such that the edges are {vi, vi+1} where i = 1, 2, …, n − 1.

Circuit

According to dictionary.com a circuit is “a circular journey or one beginning and ending at the same place; a round.” or simply “a roundabout journey or course.”.

Step 3: In search of creative solutions

Although we could think in our own solutions, turns out that this problem is a classic graph problem, thus we researched in specialized texts that contained diverse strategies to solve the problem of the shortest and minimum path graph problem. Using as literature Thomas Cormen’s book “Introduction to Algorithms”, the methods are:

Alternative 1. Breadth-First Search

Citing textually from the book: “Breadth-First search is one of the simplest algorithms for searching a graph and the archetype for many important graph algorithms… breadth-first search systematically explores the edges of the graph to ‘discover’ every vertex that is reachable from a vertex. It computes the distance (smallest number of edges) from s to each reachable vertex.”

Therefore this alternative is good when we want to traverse the least amount of stations to a destination.

Alternative 2. Depth First Search

As its name implies, DFS explores the graph by systematically going ‘deeper’ whenever possible. Thenph by systematically going as it reaches a dead-end, it ‘backtracks’ to explore edges. This process continues until the whole graph is discovered.

Useful to find paths between nodes, in this case stations.

Alternative 3. Dijkstra’s Algorithm

This algorithm solves single-source shortest-paths on a weighted, directed graph, where all edges are nonnegative. This method, in every step travels to the ‘closest’ or ‘lightest’ node, which makes it a ‘greedy’ algorithm. Thus not all greedy algorithms yield optimal solutions. However Dijkstra’s algorithm does indeed calculate shortest-paths.

Alternative 4. Floyd-Warshall’s Algorithm

This algorithm is used to find the shortest paths between all pairs of vertices in a graph. This is an approach of dynamic programming, as it only has to be computed once, after you can recover its values. Basically this algorithm considers distances with intermediate vertices, thus exploiting the relation between a path p and the other shortest-paths i to j.

Alternative 5. Kruskal's Algorithm

This algorithm is for making minimum spanning trees. Kruskal's algorithm finds a safe edge to add to the growing forest by finding, of all the edges that connect any two trees in the forest, an edge of least weight. So finally it gives us a way to traverse the graph in minimum distance.

Step 4: Transition from ideas to preliminary designs

The first thing to do in this step is to rule out (discard) ideas that are not useful or feasible. From this point of view we are excluding **Alternative 5 (Kruskal's Algorithm)** due to it being a minimum spanning tree algorithm. Thus, logically a user would not normally want to traverse the whole U-Bahn and S-Bahn system in a normal situation (although a worker would e.g. check the rails in the whole system).

A deeper analysis of the other alternatives leads us to:

Alternative 1. Breadth-first search

- It's a simple algorithm.
- It's the archetype of many other algorithms.
- It computes the paths with the smallest possible number of edges.
- Not recommended on graphs with high branching factor as it must keep all the nodes in memory.

Alternative 2. Depth-first search

- This algorithm exhausts all possibilities, and allows us to check which one is the best/count the number of all possible ways.
- It's a recursive algorithm.
- Cannot be used on infinitely deep graphs because of its recursive nature.
- Allows Cycle detection in a graph.
- Allows for bipartite graph detection.
- Finding longest path.

Alternative 3. Dijkstra's Algorithm

- Finds the shortest path between a single vertex and all other vertices

- Time complexity of $O(ElogV)$
- Not a recommended implementation on distributed systems
- Does not work with negative edges
- Greedy algorithm

Alternative 4. Floyd-Warshall's Algorithm

- All-pairs shortest path algorithm
- Time complexity $O(V^3)$
- Works for distributed systems (such as graphs of graphs).
- Approach of dynamic programming

Step 5. Evaluation and Selection of the Best Solution

Criteria

We must define the criteria that will allow us to evaluate all the different alternatives and based upon these select the solution which better satisfies the necessities of the problem. The chosen criteria will be enumerated next. Alongside each of them, there is a numeric value representing a "weight" that indicates which of these criteria is more desirable.

- Criteria A. Time Complexity:
  - [4] O(V+E)
  - [3] O(E logV)
  - [2] $O(V^2)$
  - [1] $O(V^3)$
- Criteria B. Easiness of algorithmic implementation
  - [3] Easy
  - [2] Medium
  - [1] Hard
- Criteria C. Memory impact
  - [2] Low Memory Impact
  - [1] High Memory Impact
- Criteria D. Graph implementation
  - [2] Adjacency list

[1] Adjacency Matrix

Evaluation

	Criteria A	**Criteria B**	**Criteria C**	**Criteria D**	**Total**
Alternative 1. Breadth-first search	4	3	2	2	11
Alternative 1. Breadth-first search	2	2	1	1	6
Alternative 2. Depth-first search	4	2	2	2	10
Alternative 2. Depth-first search	2	3	1	1	7
Alternative 3. Dijkstra's Algorithm	3	2	2	2	9
Alternative 4. Floyd-Warshall's Algorithm	1	3	1	2	7

Selection

As a result of the previous evaluation, we find out that we must select Alternative 1, implemented with an adjacency list and Alternative 3 for finding shortest paths with the same kind of implementation. Both alternatives are relevant to the solution of the problem, because the software will manage more than one factor to find a path between stations.