|  |  |
| --- | --- |
| Prototype & Demo  CSC3003S Capstone Project: Stage 3 | Abstract  [Using use case artefacts and the initial design, to create a prototype for demonstration to client/supervisor. Documentation of the decision on the choice of the prototype (Evolutionary)]  Elvis Sibetyu, Qocokazi Ngesi, Zuqhame Skosana  CSC3003S |

Choice of prototype: **Evolutionary Prototype**

1. **Statement and justification of choice of prototype (horizontal/vertical/combination) and throwaway or evolutionary**

**Answer:**

To demonstrate the initial version of our Inter-Domain Routing Simulator we opted to use a combination of both a horizontal and vertical prototype – although the prototype will be more vertical than horizontal. We approached the prototype this way because we want to show everything that we want to do at this early stage and since both the vertical and horizontal prototypes have their pros and cons, we want to bridge that gap and include everything.

We used the qualities of a horizontal prototype to give a broad view of the full functions of the simulator we implementing – insertion/removal of nodes, insertion/update of weights, insertion/deletion of links and IXP’s and the implementation of different routing policies.

For the vertical part of the prototype we drilled down and specifically elaborated on some features and their implementation is almost complete – which are: addition of a node, assigning name to a node, assigning link cost and deletion of a node. Those function are > 90% complete.

We chose to do an evolutionary prototype because we fully understand the requirements of the project, what is expected from us by the client and we want to use part of the system as the prototype, thus we will add more functionality to this initial version to approach the complete end product. We chose an evolutionary prototype because we want to implement the requirements that we know very well and add more requirements as time goes – more like iterative development.

1. **Class definitions**

**Answer:**

* Node

This class is used to create an object of a node. The data of the node will be stored in variables, the variables are node name, node id, node edge and connected IXP’s. Each node will have information oh how top get to the nodes it is linked with. This information will be stored in an adjacency matrix. The user will specify the name of the node themselves. The node will also have an x coordinate and a y coordinate.

* IXP

The IXP class provides connection links between nodes in Tier 3 to allow direct exchange of traffic at 0 cost. They are used to reduce the traffic in the ISP’s (Tier 2 & 3 nodes).

* Routing Table

This class is used to store information for each node and which paths it can use to get from itself to another node, and which is the shortest path. It will use Dijkstra’s routing algorithm and Bellman Ford’s algorithm.

* Graph

This class will consist of all the edges and nodes connected with each other to form a graph. The graph is undirected since traffic can go back and forth on a link. Information from this class is used to create the graph visualization.

* Networkx

This class is used to create the axis in which the graph will be drawn and visualized in. It inherits everything from the Graph class.

1. **Class Member Functions**

* Add\_node (String name):

This function is in the Node class. It is used to add a node to the graph. The user will have to specify the name of the node.

* Add\_weigh (long weight):

This method is used to add a cost to an edge between two nodes. It is used to link two nodes.

* Modify\_Weight (long weight):

This method modifies a link cost.

* Delete\_Node (String node):

This method is used to delete a particular node rom the topology. The name of the node is specified by the user.

* Show\_shortest\_path (source node, destination node)

This method computes the cost is takes to go from the source node to the destination node and outputs the cost plus the route it took. It will either use Dijkstra’s algorithm or Bellman Ford’s algorithm.

* Draw\_graph ():

This Function draws the visualization of the graph in a set of axis. It will show all the nodes and the edges including the name of the nodes.

* On\_closing ():

This unction deletes all the information that was stored in the text file from a user’s session and it also exists the application.

1. **Class Inheritance**