SOFTWARE REQUIREMENTS SPECIFICATION

for

Automatic Creation of Fiber Netwoks Synoptics

Version 1.0

Prepared by : Antínio Sérgio Gomes (a67645)

Submitted to : Proef

January 26, 2023

Contents

1 Introduction

1.1 Purpose

In planning, construction and maintenance of network infrastructure is imperative that numerous project files be created. Amidst such files, a Trace Map of the infrastructure's future layout in the desired terrain is created as well as multiple configuration tables. The Trace is created using *"Computer Aided Design" (CAD)* tools and it must be then converted to a logical representation of the same network infrastructure in a *CAD* diagram known as a Networks Synoptic Map. This Map is required for the acquisition of appropriate licensing and also for for future consultation of the network's devices, resources and existing connections for update or maintenance. This conversion from Trace Map to Network Synoptics Map is currently performed manually thus being subject to human error and being slow and time consuming.

1.2 Intended Audience and Reading Suggestions

An application for the automation of creation and update of Fiber Networks Synoptics Maps is targeted and intended for numerous companies who are responsible for the construction of network infrastructures as well as it's update and maintenance.

1.3 Project Scope

Before a network's layout and infrastructure is designed, a detailed survey must be done in order to understand the targeted area's necessities and other existing infrastructures that may need be used in later stages, this task is performed by teams in the terrain.
With the information gathered by the surveys, the team responsible for infrastructure design and project is then able to create a Trace Map, which displays the desired network layout and infrastructure over the geographic Map of the area.
Trace Maps are large and complex and misses crucial information making it impractical for consultation purposes, thus a Syoptics Map is created wich displays the network in a logical diagram, containing all the devices and connections according to the Trace Map and also is completed with adicional information such as device specific details and other types of data.
Networks Synoptics Map is then used by maintenance teams in the terrain and used for license acquisition.
As described above, all the tasks are interconnected and each depend on other tasks, tis notion of relationship can be observed in the following Domain Model Diagram.

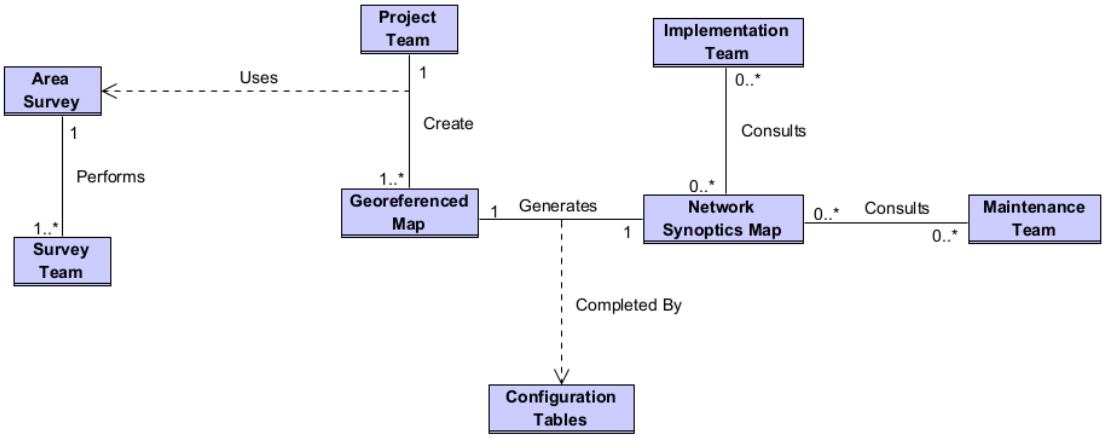

Figure 1.1: Domain Model for Fiber Networks Synoptics

2 Overall Description

2.1 Product Perspective and Functions

The automation of Network Synoptics Map creation processes aims to reduce overall time consumption as well as human error while retaining the adaptability to change.

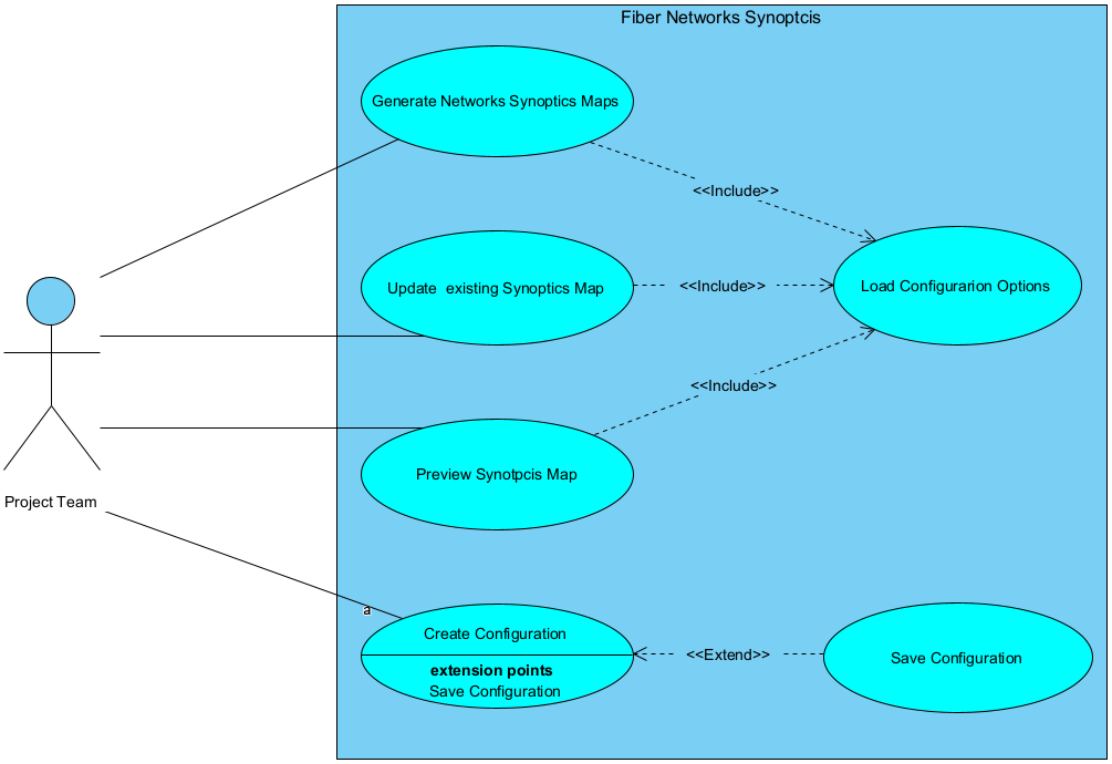

Figure 2.1: Use Case Diagram for Fiber Networks Synoptics

As seen above, a multitude of tasks can be performed, each performing a different action, with some tasks depending on, or using the result of other tasks. The use cases go as follows:

1. A new networks Synoptics Map of a specific Trace Map can be generated, given a conversion settings configuration.

2. An Existing Synoptics Map can be updated, given a number of configuration tables pertaining to devices and connections represented on the Synoptics Map, given a

conversion settings configuration.

3. The resulting Synoptics Map of a Trace Map can be previewed for validation of the Trace Map, given a conversion settings configuration.

4. Conversion settings configurations can be created and saved so the conversion process is dynamic and can adapt to possible changes in networks infrastrucures creation procedures, changes in laws and regulations as well as being able to adapt to different *ISP*.

5. Configurations can be saved for further use, and loaded before a conversion is performed.

2.2 Operating Environment

The application will run mainly on desktop environment on windows operating system. No internet connection is directly required for use of this application.

3 System Features

"Fiber Networks Synoptics" aims for the automatic creation and update of Networks Synoptics Maps, thus most of it's features and requirements pertain to such task and aim to perform it in the most versatile, efficient and yet user friendly way possible.

3.1 Description and Priority

"Fiber Networks Synoptics" features ordered form those with the most priority to those with least priority.

1. Converting an existing Trace Map in it's respective networks Synoptics Map.
2. Updating an existing networks Synoptics with the contents from configuration table files.
3. Configuration of the conversion parameters.
4. Saving and loading configurations.

3.2 Functional Requirements

Num.	**Name**	**Description**	**Valid. Method**	**Valid. Entity**	**Pri.***
FR.01	Trace Map Loading	Must be possible to load a Trace Map in the form of a *.DWG* or *.DXF CAD* file.	Manual testing.	Proef	5
FR.02	Primary Synoptics Map Generation	Must be able to generate Synoptics Maps related to a loaded Trace Map.	Direct comparison between Synoptics Maps obtained from the system and manually created ones for the same Trace Maps.	Proef	5

FR.03	Secondary Synoptics Maps Generation	For each *JSO* type block on a Trace Map, the application must be able to generate one secondary Synoptics Map.	Direct comparison between secondary Synoptics Maps generated by the application and manually created ones based on the same Trace Map.	Proef	5
FR.04	Synoptics Map Update	System must be able to update a Synoptics Map with information from Microsoft Excel *.XLSX* files given by the user.	Direct comparison between Synoptics Maps updated by the system and manually updated ones.	Proef	3
FR.05	Synoptics Map Generation Preview	System must be able to create a preview of a Synoptics Map generation for a given Trace Map.	Direct comparison between a preview and a Map previously generated.	Dev. Team	2
FR.06	Conversion Rules Cofigurability	It must be possible to change the conversion rules for every conversion.	Comparing Synoptics Maps obtained from differently natured Trace Maps and comparing the results to manually created ones.	Proef	3
FR.07	Changing Block Names	Must be possible to process different block names for similar structures for Trace Maps.	Analysing the resulting conversions of different Trace Maps.	Proef	4
FR.08	ISP Adaptability	System must be able to handle Trace Maps and Synoptics form different ISP.	Analysing the resulting conversions of different Trace Maps.	Proef	5
FR.09	Configuration Table Generation	Create a configuration table for each *Splitting Junction* type block of a Trace Map.	Exhaustive testing for each possible Trace Map type.	Dev. Team	3
FR.10	Automated Link and Splitter allocation	Each configuration table, in *.XLSX* format, must be filled according to optimization rules and performance targets, given the base information of that specific *Splitting Junction* block.	Exhaustive testing for each possible Trace Map type.	Dev. Team	2

FR.11	Link Table Generation	Upon the creation of all Synoptics Maps, for a given Trace Map, a link table must be created containing information regarding the optical cables that arrive and depart each *Optical Splitting Junction*.	Exhaustive testing for each possible Trace Map type.	Dev. Team	3

Table 3.1: Detailing of App Functional Requirements.

4 Nonfunctional Requirements

4.1 Usability, Performance and System Requirements

This application is going to be used by a wide array of people with differing skills and expertise, thus the need for a easy and friendly user interface arises. It is assumed that every user of this application as prior knowledge regarding Trace Maps and Synoptics Maps, as well as knowledge regarding network infrastructures projects. The description and targets for usability requirements can be represented in the following table.

Number	**Name**	**Description**	**Valid. Method**	**Valid. Entity**
NR.01	Synotpics Generation Usability	An unexperienced user must be able to generate a new Synoptics Map from any type of Trace Map in under 3 minutes without the need for external guidance, other than the user's manual.	Acceptance testing	Proef
NR.02	Configuration File Editing Usability	A user with prior knowledge in networks infrastructures projects must be able to create a correct configuration file for a given *ISP* in under 10 minutes.	Acceptance Testing	Proef
NR.03	Synoptics Generation Performance	The functional requirement **2**, **3**, **9**, **10** and **11** must be all performed in under 3 seconds after the generation begins.	System Testing	Dev. Team
NR.04	Configuration File Size	The size of configuration files must be less than 4 MB.	System Testing	Dev. Team

Table 4.1: Detailing of App Non-functional Requirements.

5 Solution Design

5.1 User Interface

An appropriate solution must satisfy the technician's necessities to the highest degree possible, one such necessity is often an intuitive yet complete user interface capable of performing all required tasks whilst allowing for anyone with prior knowledge of network design to be able to use most if not all of the application's capacities when devoid of special training. Figure 5.1 shows an example of such graphical user interface, previously discussed and agreed upon by both parties, namely the costumer enterprise and the development team. The left hand menu is targeted at the common and initial tasks of creating synoptics maps and respective tables. In here we can observe:

- **Trace Map:** File browser to load the trace map to be processed.
- **Destination:** File browser to select the directory to which the end result will be saved to.
- **Config File:** File browser to load the configuration file to be used in the processing and generating of CAD files.
- **Preview** button to display an initial render of the synoptics maps.
- **Generate** button to generate the synoptics CAD files and XLSX table files.

The right hand menu, as seen in Figure 5.1, is a configuration table allowing for consultation, editing and creation of configurations files to be used in the conversion process. Each block's structure can be defined beforehand by use of a simple nomenclature style in which the block name, unique identifier and attributes can be written in.

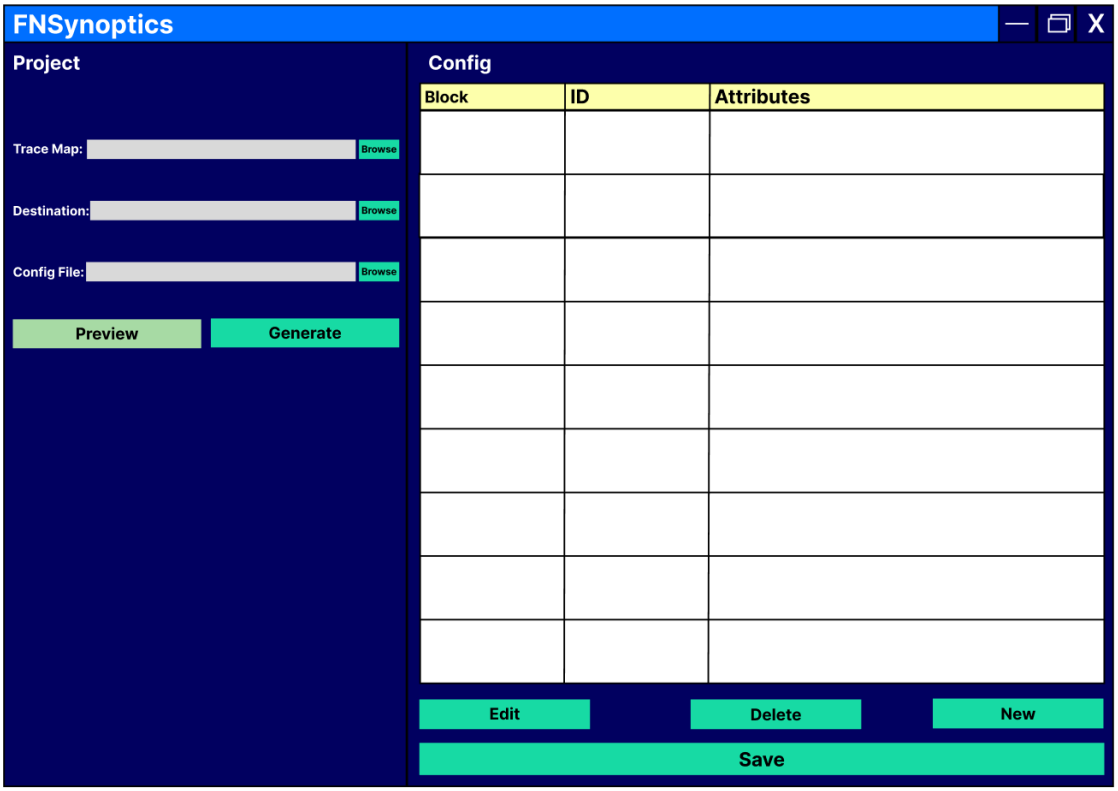

Figure 5.1: Mockup of GUI main menu

5.2 Software Architecture

A software's architecture can be represented by use of many forms of diagrams and structures, for this effect a class diagram based in UML was chosen. In here the software's packages and most important classes can be represented, along with the main attributes and methods for each class and relationships between classes. In the case of this application, two distinct languages where chosen for the conception and thus two distinct software pieces must be developed, an application GUI package, written in Java and a processing and conversion package written in Python. The first is composed by the following Java classes as shown in figure 5.2:

- GUI class responsible for all the methods relating to the user interface.
- Config class responsible for creation and usage of configuration files.
- CAD Class responsible for loading and processing of DXF files, being trace maps or synoptics maps.
- Table Class responsible for the loading and processing of XLSX files., being link tables or configuration tables.

- Main class that brings the code structure together.

Most importantly, the processing logic package is responsible for most of the work and thus being more structured and complex. This package is defined by the following Python classes, as defined in figure 5.2.

- TraceMap class being an objectified representation of a CAD Trace Map, completed with methods for manipulation and processing of such data.
- Block class, used by TraceMap as a building block of trace maps, represents a generalized block with it's unique identifier, attributes and relations with other blocks.
- SynopticsMap class being an objectified representation of a CAD Synoptics Map, in similarity with trace maps.
- Table class, as an extention of ConfigTable and LinkTable classes, which defines a table's strucuter and manipulation methods.
- Main class that brings the code structure together.

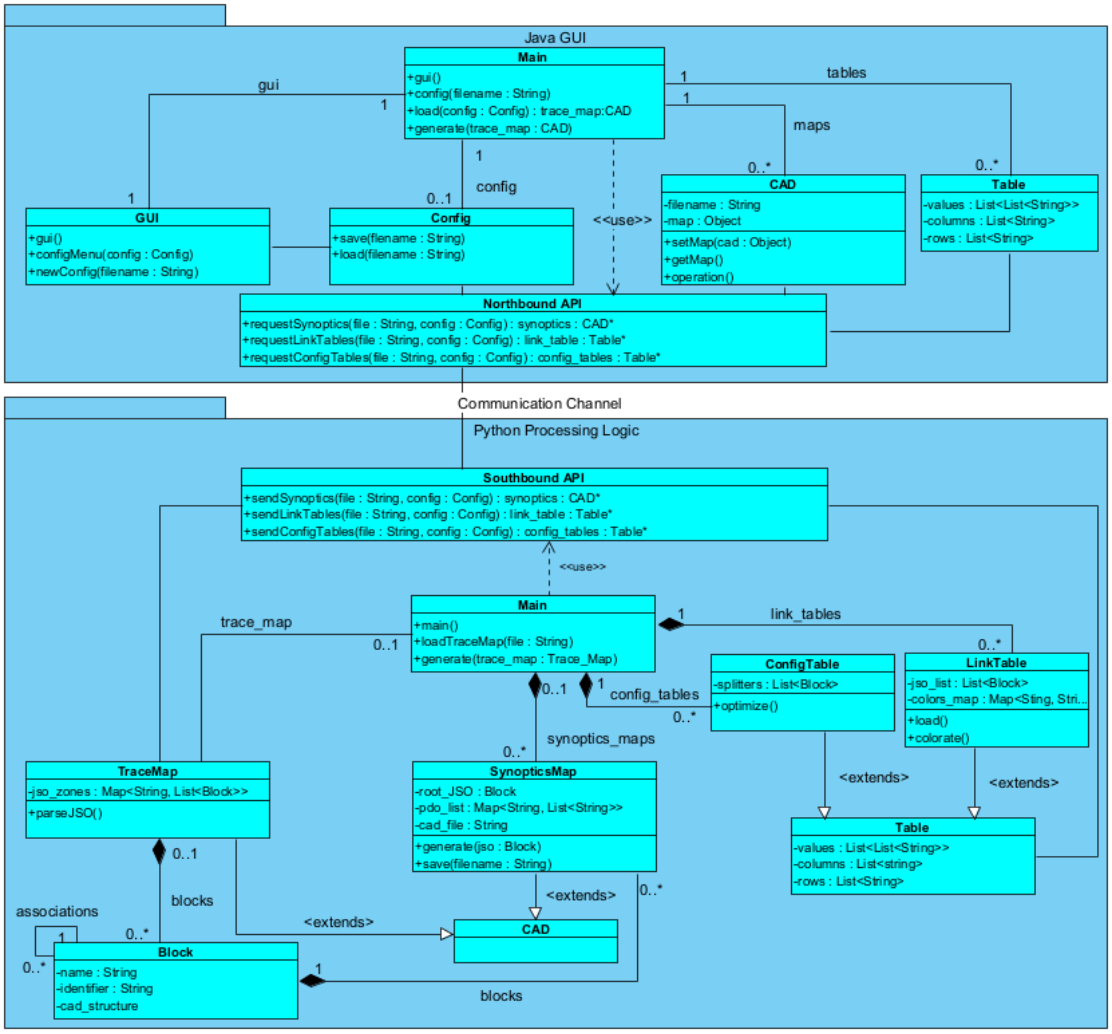

Figure 5.2: Class and package definition for GUI and Logic modules.