MINISTRY OF EDUCATION AND RESEARCH
TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

SUMMARY
of the License Thesis entitled:

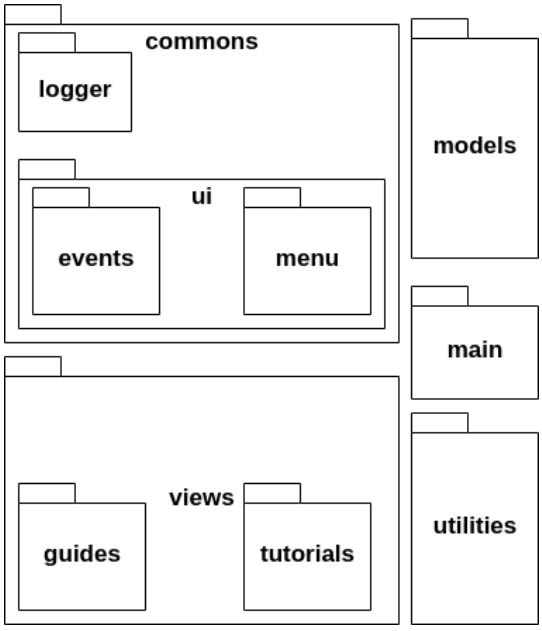
SueC – An Editor and Interpreter for Pseudocode

Author: **Mihai PÎȚU**
Advisor: **Assoc. Dr. Eng. Emil Ștefan Chifu**

1. Requirements:

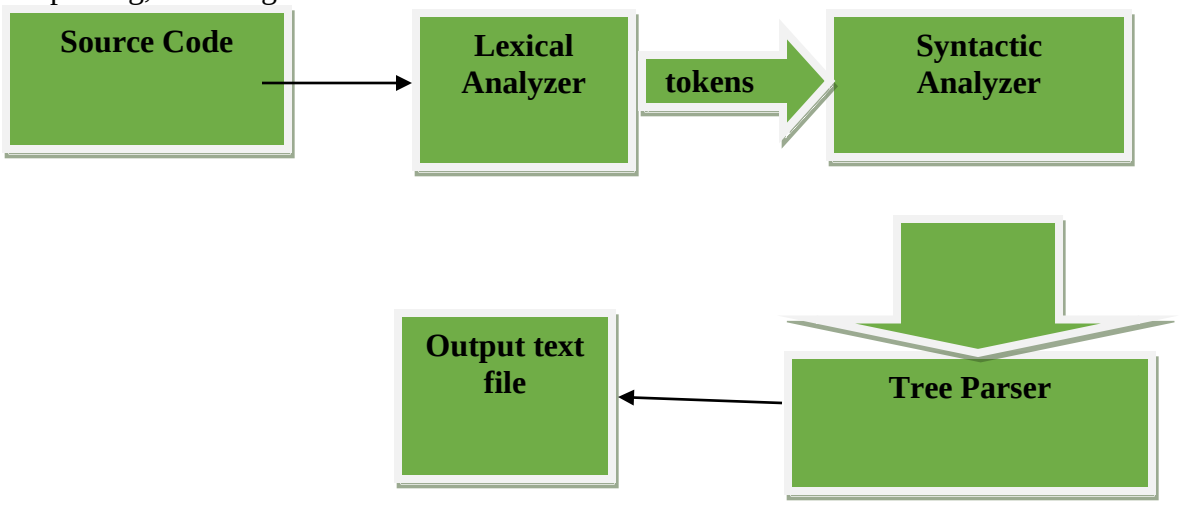
The aim of this project is to design an application that acts as editor and interpreter for a programming language that resembles pseudocode.
My main focus is to design the interpreter for the said language and the editor application that acts as an educational tool, not just as a simple text editor.

2. Proposed solutions:



The project is split in two main components: the editor application and the interpreter.
The editor application is a Java Desktop application, with the views created using Java Swing. Architecturally speaking, the editor app meets the structure of MVC pattern, as it is mainly used for defining the tutorial and guide views.
Besides that, the editor application has a view specific for editing SueC source code files. After editing the files under the file view, these can be compiled. Their relative path is sent to the interpreter in order to interpreted, the result being displayed in the view.
The interpreter is a C executable program that is split into three main components that communicate in a pipeline manner: lexical analyzer, syntactic analyzer and tree parser. The lexical analyzer reads the

input from the source code file and splits the source code into tokens. The syntactic analyzer reads the tokens from the lexical analyzer and checks for specific statement patterns. If found, it generates a tree based on the statement pattern given. The tree parser reads the trees given from the syntactic analyzer and performs a bottom-up parsing, returning the result of the tree.



3. Results obtained:

I have managed to have a functional editor application that works with the interpreter, having a response time close to 1 second when compiling a file or running a tutorial.

4. Tests and verifications:

For a quick demo, I have tested with running small code snippets to check the response time. Besides that, I have checked the functionalities of the tutorials and guides to ensure fluidity and efficiency when running the tutorials.

5. Personal contributions:

I have designed the editor application with all the functionalities for file handling and working with tutorials and guides. I have designed the programming language and interpreter based on the laboratory documentation that I have done at Formal Language and Translators and Translator Design courses.

6. Documentation sources:

The main documentation sources have been online articles and websites in the domain of Formal Languages and Translators, alongside the bibliography provided from the afro-mentioned course. These sources are found under the Bibliography chapter.

Date: 08.07.2020 Author **Mihai PÎȚU**

Coordonator _____