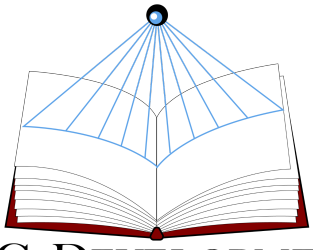
System Requirements Specification Document

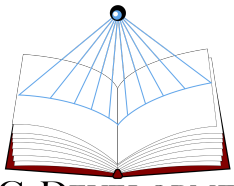
for Macleod: A CLIF Parser, designed for Torsten Hahmann and Jake Emerson



R.C. DEVELOPMENT

Designed by Reading Club Development:
Matthew Brown, Gunnar Eastman, Jesiah Harris, Shea Keegan, Eli Story

Version 1.3
Dec. 13, 2022



R.C. DEVELOPMENT

System Requirements Specification:

Table of Contents

Date	**Reason for Change**	**Version**
10/18/2022	Initial Creation	1.0
10/27/2022	Specified when the Unit Testing Phase occurs.	1.1
12/01/2022	Updated and color-coded requirements	1.2
12/13/2022	Updated based on feedback	1.3

1. Introduction

The Common Logic Interchange Format (CLIF) Parser is a capstone project for Dr. Torsten Hahmann and Jake Emerson, in partial fulfillment of the Computer Science BS degree for the University of Maine, completed by the Reading Club Development (RCD) team. Dr. Hahmann is a professor at the University of Maine with affiliation to the Spatial Data Science Institute and research interests in knowledge representation, logic, and automated reasoning. Jake Emerson works for Jackson Laboratories in Bar Harbor, Maine, where he would implement this parser into the workflow of projects he is involved with. RCD consists of five seniors from the University of Maine: Matthew Brown, Gunnar Eastman, Jesiah Harris, Shea Keegan, and Eli Story.

This SRS will detail the functional and nonfunctional requirements set forth for this project, the deliverables necessary for completion, and document the consent of the team members and the client. The CLIF Parser is to be used by researchers so as to better document their methods for experiments so as to allow for more repeatability in their experiments.

1.1 Purpose of This Document

This SRS is meant to enumerate the functional and nonfunctional requirements set forth for the CLIF Parser that RCD has been tasked with developing. This SRS will also describe the work that RCD is to complete in the course of this project and the two-semester long Capstone class at the University of Maine. The intended readership of this document consists of the client and the RCD team so as to serve as an agreement between RCD and the clients on how to effectively develop the proposed CLIF Parser.

1.2 References

Macleod, Dr. Torsten Hahmann, GitHub, 2022, https://github.com/thahmann/macleod.

1.3 Purpose of the Product

The field of biology is currently facing a “crisis of reproducibility” according to Mr. Emerson. The development of an ontological reader is, for him, paramount due to the congruity of semantics within ontological statements. The CLIF Parser is to be a stepping stone in the progress toward unified Common Logic, allowing for more properly defined variables, and hopefully slightly mitigating this crisis of reproducibility.

On a smaller scale, the purpose of the CLIF Parser is primarily to parse CLIF files to ensure they are syntactically correct according to CLIF standards. The library should also translate from the CLIF syntax to TPTP, or other logic-based conventions. Another purpose of the

product is to build upon the previously developed Macleod IDE, so Common Logic can have an IDE dedicated to supporting it.

1.4 Product Scope

The scope of the project is twofold. Firstly there is the parser. At the moment there is an existing parser that works, but is outdated and is missing some parsing capabilities. At the moment the installation of the parser is very complicated and time consuming. During the scope of this project, we hope to repair and further the development of the parser as well as turn it into a python library via Poetry. This will allow an easy installation for users.

The second part of this project is the IDE. Again there is an existing IDE, however it is slow and cumbersome. The goal to solve this problem is to create a Spyder plugin. This will allow the user to interact with the parser in an environment with many tools to their disposal.

Overall the project will include a parser for managing CLIF files and an IDE to help that management. The two systems will be separate. The parser can be used through the command line separate to the IDE. The IDE will rely on the parser to parse the files, therefore it can not be used as a standalone system.

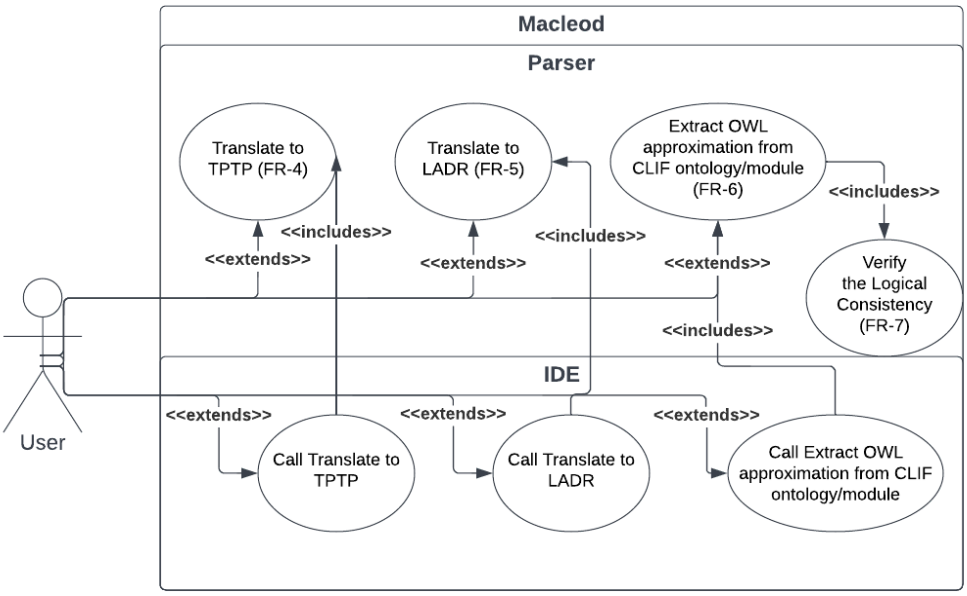


Fig. 1: Primary Use Case Diagram: This diagram depicts how Macleod is to be used as a parser and as an IDE.

This diagram briefly depicts the anticipated interactions between Macleod and the user. The scope of the system is intended to be that the system will translate CLIF files to other file formats and verify the syntactic and logical consistency of pre-existing CLIF files. The ability to edit the CLIF file will be granted if an error has been located.

2. Functional Requirements

In this section, we outline the requirements that detail the functionality of the system. Priority is measured on a scale from 1 to 5, with 5 being the most critical.

Blue = Already done, should work when project is finished
Pink = Needs to be completed
Green = New

1. The system shall identify Quantified Variables (and their scope and use).
2. The system shall identify Predicates.
3. The system shall identify Statements from file.
4. The system shall take CLIF and output TPTP.
5. The system shall take CLIF file and produce LADR output.
6. The system shall extract OWL approximation from CLIF ontology/module.
7. The system shall verify the logical consistency of a CLIF ontology or module.
8. The system shall prove theorems that encode intended consequences (e.g. properties of concepts and relations) of ontologies/modules.
9. The system shall preserve all comments.
10. The system shall bringing up an editor to fix syntax errors.
11. The system shall, identify Instructions from file.
12. The system shall provide a button to parse a CLIF file.
13. The system shall provide a button to parse and convert a file.

Number	FR-1
Name	The system shall identify Quantified Variables (and their scope and use).
Summary	The system shall identify variables in a provided logical statement
Priority	5
Preconditions	An input has been entered
Postconditions	Variables would have been identified
Primary Actor	
Secondary Actors	

Trigger	A logical statement is given as an input
Steps	The system shall examine each individual string of symbols separated by white space, and each string that is not defined by the file format shall be identified as a variable. (examples of non-variables include “for all” symbols, “for each”, punctuation, etc.)
Open Issues	
Tests	FR-1 will be tested during the Unit Testing phase (early 2023) by providing logical statements and asserting that the variables identified by the system are the same as those identified by the developers.

Number	FR-2
Name	The system shall identify Predicates.
Summary	The system will identify predicates and function symbols in a inputted logical statement
Priority	5
Preconditions	An input has been entered
Postconditions	Predicates would have been identified
Primary Actor	
Secondary Actors	
Trigger	A logical statement is given as an input
Steps	The system shall examine each symbol/word in the input, and each one that matches a list of predicate symbols that will be extracted from the file, shall be

	identified as a predicate.
Open Issues	
Tests	FR-2 will be tested during the Unit Testing phase by providing logical statements and asserting that the predicates identified by the system are the same as those identified by the developers.

Number	FR-3
Name	The system shall identify Statements from file.
Summary	The system can read a file and identify logical statements written inside
Priority	5
Preconditions	A file must be given
Postconditions	Statements are identified
Primary Actor	
Secondary Actors	
Trigger	A file location is given to read
Steps	
Open Issues	
Tests	FR-3 will be tested during the Unit Testing phase by providing a CLIF file and asserting that the logical statements identified by the system are the same as those identified by the developers.

Number	FR-4
Name	The system shall take CLIF and output TPTP.
Summary	The system should be able to take a CLIF file as an input and give a TPTP file as an output
Priority	5
Preconditions	The system is not currently working on another task.
Postconditions	A TPTP file is produced as output
Primary Actor	
Secondary Actors	
Trigger	A CLIF file is given as input
Steps	
Open Issues	
Tests	FR-4 will be tested during the Integration Testing phase, by providing the system with CLIF files of varying length and complexity and asserting that the system translates them to TPTP correctly.

Number	FR-5
Name	The system shall take CLIF file and produce LADR output.
Summary	The system should be able to take in a CLIF file and produce LADR output
Priority	4

Preconditions	CLIF file must be given and accepted by system
Postconditions	The system will produce LADR out from a given CLIF file
Primary Actor	
Secondary Actors	
Trigger	A CLIF file is given to the system
Steps	
Open Issues	
Tests	FR-5 will be tested during the Integration Testing phase, by providing the system with CLIF files of varying length and complexity and asserting that the system translates them to LADR correctly.

Number	FR-6
Name	The system shall extract OWL approximation from CLIF ontology/module.
Summary	The system should be able to extract an OWL (Web Ontology Language) approximation from a CLIF ontology or module
Priority	4
Preconditions	A CLIF ontology or module must be given to the system. The system must have the ability to translate CLIF ontologies to approximate OWL ontologies
Postconditions	An OWL approximation will be

	created/extracted by the system
Primary Actor	
Secondary Actors	
Trigger	A CLIF ontology/module is given to the system
Steps	
Open Issues	FR-6 will be tested during the Integration Testing phase, by providing the system with CLIF files of varying length and complexity and asserting that the system translates them to OWL correctly.

Number	FR-7
Name	The system shall verify the logical consistency of a CLIF ontology or module.
Summary	The system shall be able to verify the logical consistency of a CLIF ontology/module
Priority	5
Preconditions	The system must have access to theorem provers and an inputted CLIF file.
Postconditions	A logical CLIF ontology/module will be identified and ready for processing by the system
Primary Actor	
Secondary Actors	
Trigger	A CLIF ontology/module is given to the system

Steps	
Open Issues	
Tests	FR-7 will be tested during the Integration Testing phase, by providing the system with CLIF files of varying length and complexity, and correctness, and asserting that the system confirms correctly whether or not the files are logically consistent.

Number	FR-8
Name	The system shall prove theorems that encode intended consequences (e.g. properties of concepts and relations) of ontologies/modules.
Summary	The system should have theorem proving capabilities that can encode the intended consequences of ontologies/modules given to the system. Examples of intended consequences are properties of concepts and relations or competency questions.
Priority	3
Preconditions	The system must have a theorem prover or theorem proving capabilities that can process given ontologies/modules.
Postconditions	Intended consequences of an ontology or module will be proven and reported by the system
Primary Actor	
Secondary Actors	

Trigger	
Steps	
Open Issues	
Tests	FR-8 will be tested during the Integration Testing phase, by providing the system with CLIF files of varying length and complexity and asserting that the system translates them to TPTP correctly.

Number	FR-9
Name	The system shall preserve all comments
Summary	The system shall maintain any comments present in input CLIF files, and make them present in the associated output files.
Priority	3
Preconditions	A CLIF file with comments has been submitted as input.
Postconditions	The comments are still present in the translated output.
Primary Actor	User
Secondary Actors	
Trigger	
Steps	
Open Issues	
Tests	FR-9 will be tested during the Integration Phase by providing the system with CLIF files that have comments in various places

	and of various lengths, and ensuring those comments are intact in the output.

Number	FR-10
Name	The system shall bringing up an editor to fix syntax errors.
Summary	The system shall open CLIF files and point out errors.
Priority	3
Preconditions	The system is reading through a CLIF file
Postconditions	A CLIF editor has been opened to the location of the error so that it can be fixed.
Primary Actor	User
Secondary Actors	
Trigger	The system finds a syntax error
Steps	The system marks the location of the error, opens a CLIF editor, and scrolls to the location of the error.
Open Issues	
Tests	FR-10 shall be tested during the Integration Testing phase, by providing the system with CLIF files that have various syntactical errors in various places.

Number	FR-11
Name	The system shall, identify Instructions from file.

Summary	The system shall be able to identify instructions outside of the logic of the ontology, such as import statements and module declarations
Priority	5
Preconditions	A file must be given
Postconditions	Instructions are identified
Primary Actor	User
Secondary Actors	
Trigger	A file location is given to read
Steps	
Open Issues	
Tests	FR-11 will be tested during the Unit Testing phase by providing CLIF files with import statements, module declarations, and both, and assuring that those identified by the system match those identified by the developers.

Number	FR-12
Name	The system shall provide a button to parse a CLIF file.
Summary	The system shall provide one button that will parse a CLIF file and ensure that it is syntactically correct.
Priority	5
Preconditions	A CLIF file is open in spyder.

Postconditions	The file has been ensured to be syntactically correct or errors have been raised.
Primary Actor	User
Secondary Actors	
Trigger	The button is pressed
Steps	
Open Issues	
Tests	FR-12 will be tested during the Integration Testing phase by providing CLIF files with and without issues and testing them in spyder.

Number	FR-13
Name	The system shall provide a button to parse and convert a file.
Summary	The system shall provide one button that will parse a CLIF file and ensure that it is syntactically correct and translate it to an indicated other language.
Priority	5
Preconditions	A CLIF file is open in spyder.
Postconditions	The file has been translated or errors have been raised.
Primary Actor	User
Secondary Actors	
Trigger	The button is pressed

Steps	
Open Issues	
Tests	FR-13 will be tested during the Integration Testing phase by providing CLIF files with and without issues and testing them in spyder to see that they are translated correctly.

Table 1: Functional Requirements of Macleod: This table outlines the core functionalities that the system shall have upon completion.

3. Non-Functional Requirements

In this section, we will outline the requirements that do not directly involve the functionality of the system.

Blue = Already done, should work when project is finished
Pink = Needs to be completed
White = Should just work
Green = New

1. The system shall work on Linux, Mac, and Windows.
2. The system shall be installable via pip as a python library.
3. The GUI will respond to all inputs with at least a “loading” message within 2 seconds 90% of the time.
4. The system shall correctly translate at least 95% of input content.
5. The system shall translate an individual logical statement within 1 second 95% of the time, to ensure that the translation of a document does not take a prohibitive amount of time.
6. The system shall parse a CLIF file within 3 seconds 90% of the time.
7. The system shall be accompanied by a User Guide, which should allow users to use the system within 30 minutes.
8. The system itself shall not keep a record of any files it translates.
9. The system shall be installable as a spyder plugin.

Number	NFR-1

Description	The system shall work on Linux, Mac, and Windows.
Priority	5
Tests	All tests shall be performed on a Linux, Mac, and Windows system.

Number	NFR-2
Description	The system shall be installable via pip as a python library.
Priority	5
Tests	Test NFR-2: Once completed, we shall attempt to install the system using pip.

Number	NFR-3
Description	The GUI will respond to all inputs with at least a “loading” message within 2 seconds 90% of the time.
Priority	4
Tests	Test NFR-3: Measure the time it takes for the GUI to respond to various requests.

Number	NFR-4
Description	The system shall correctly translate at least 95% of input content.
Priority	4
Tests	Test NFR-4: Have the system translate

	small and large files and ensure that there are no errors within at least 95% of the resulting files.

Number	NFR-5
Description	The system shall translate an individual logical statement within 1 second 95% of the time, to ensure that the translation of a document does not take a prohibitive amount of time.
Priority	4
Tests	Test NFR-5: Have the system translate the same files as in Test NFR-4, and record how long it takes for the system to translate those files.

Number	NFR-6
Description	The system shall parse a CLIF file within 3 seconds 90% of the time.
Priority	4
Tests	Test NFR-6: Have the system parse files with similar variation as in Test NFR-4, and record how long it takes for the system to parse those files.

Number	NFR-7
Description	The system shall be accompanied by a User Guide, which should allow users to use the system within 30 minutes.

Priority	5
Tests	Test NFR-7: Our sister team in the Capstone class will be given access to the system and the User Guide, and we will determine whether they are able to make use of the system after 30 minutes.

Number	NFR-8
Description	The system itself shall not keep a record of any files it translates.
Priority	2
Tests	Test NFR-8

Number	NFR-9
Description	The system shall be installable as a spyder plugin
Priority	5
Tests	Test NFR-2: Once completed, we shall attempt to install the system as a spyder plugin

Table 2: Non-Functional Requirements of Macleod: This table outlines the requirements that do not involve the functionality of the system.

4. User Interface

This section briefly outlines any user interface technicalities that will need to be completed for this project.

See “User Interface Design Document for CLIF Parser.”

5. Deliverables

This section will enumerate the deliverables RCD is to produce throughout the Fall 2022 and Spring 2023 University of Maine semesters in accordance with the requirements set forth both by the Capstone class and the client's desired outcome.

The following deliverables shall be produced and given to the client:

Electronic files containing the following:

- Systems Requirement Specification
    - This will be shared via Google Docs.
    - We will send the SRS to the client digitally October 18, 2022.
- System Design Document
    - This will be shared via Google Docs.
    - We will send the SDD to the client digitally November 8, 2022.
- User Interface Design Document
    - This will be shared via Google Docs.
    - We will send the UIDD to the client digitally November 21, 2022.
- Code Inspection Report
    - This will be shared via Google Docs.
    - We will send the CIR to the client digitally in the spring, on approximately February 16, 2023
- User Manual
    - This will be shared via Google Docs.
    - We will send the UM to the client digitally in the spring, on approximately March 1, 2023
- Administrator Manual
    - This will be shared via Google Docs.
    - We will send the AM to the client digitally in the spring, on approximately March 14, 2023
- All source code
    - This will be stored on a GitHub repository which the client will have permanent access to.
- Macleod Verion 1.1
    - We will send the CIR to the client digitally in the spring, on approximately May 1, 2023
- Any other software required for installation and execution of the delivered program.
    - This will be stored on a GitHub repository which the client will have permanent access to.

6. Open Issues

This section will enumerate issues that have been raised, but are as of yet lacking a solution. These issues will be addressed later in development.

Open Issue	**Approximate Resolution Date**
1. Download and get Macleod up and running on our machines.	10/21/22
2. Isolate the first conventions of CLIF syntax we are to parse.	10/21/22
3. Make a Python library with Poetry release	4/1/23
4. Simplify configuration after install	5/1/23
5. Need to update CLIF BNF grammar	3/1/23
6. Cleanup GUI	5/1/23
7. Remove pyparsing dependency	2/1/23
8. Parser.py doesn't use prefix when importing	3/1/23
9. Parser doesn't like the /** <comment> **/ comments	2/1/23
10. Fix/Test setup.py	2/1/23

Appendix A

This appendix details the expectations that RCD shall uphold to the client upon completion of this document, and how future changes to this document shall be made.

RCD and the client, upon the signing of the document, are agreeing that this SRS contains a compilation of the nonfunctional and functional requirements necessary for the CLIF Parser. RCD and the client agree that these requirements are to be developed, tested, and integrated over the course of the Fall 2022 and Spring 2023 University of Maine semesters. The client, in signing this SRS, agrees that these requirements are sufficient for completion of this project. The team, RCD, agrees that these requirements are meant to be agile and flexible in nature, so if the need arises, the requirements may change in accordance with the client's wishes.

Any changes made to this document must be approved by all members of RCD and the client via signatures to an additional appendix wherein the changes are enumerated and detailed. Changes to this document include, but are not limited to, updating requirements, removing requirements, adding requirements, and changing structure as to be in accordance with the Capstone requirements for the University of Maine course this project is managed through. The signing of this appendix consents all members of RCD and the client that this structure of implementing changes is acceptable.

Name:	Signature:	Date:
Torsten Hahmann	_____	__/__/__
Jake Emerson	_____	__/__/__
Matthew Brown	_____	__/__/__
Gunnar Eastman	_____	__/__/__
Jesiah Harris	_____	__/__/__
Shea Keegan	_____	__/__/__
Eli Story	_____	__/__/__

Appendix B

This appendix will contain the agreement that all members of RCD have read and consent to the document in its entirety.

Through signing this appendix, we, as the members of RCD, agree that we have reviewed this document fully, we agree to the formatting of this document, and agree to the content that is located within this SRS. Each member of RCD may have minor disagreements with certain parts of this document, though by signing below, we agree that there are not any major points of contention within this SRS. We have all agreed to these terms and placed our signatures below.

Name:	Signature:	Date:
Matthew Brown Comments:	_____	__/__/__
Gunnar Eastman Comments:	_____	__/__/__
Jesiah Harris Comments:	_____	__/__/__
Shea Keegan Comments:	_____	__/__/__
Eli Story Comments:	_____	__/__/__

Appendix C

This appendix will outline the approximate contributions of each of the team members of RCD to the completion of this SRS.

Matthew Brown

- Formatted the document.
- Wrote all that was required from the template except the functional and nonfunctional requirements.
- Conducted review for RCD team on whole SRS.
- Attended client approval meeting.
- Edited the entire SRS.
- Worked on all sections.
- Contributed about 30% of the work.

Gunnar Eastman

- Created the document
- Co-authored the functional requirements.
- Authored the nonfunctional requirements.
- Conducted initial grammatical review.
- Edited the entire SRS.
- Worked on all sections.
- Contributed about 30% of the work.

Jesiah Harris

- Co-authored the functional requirements
- Attended the client approval meeting.
- Worked on §2
- Contributed about 15% of the work.

Shea Keegan

- Co-authored the functional requirements.
- Worked on §2
- Contributed about 15% of the work.

Eli Story

- Co-authored the functional and nonfunctional requirements.
- Constructed the context diagram.
- Worked on §1,2
- Contributed about 10% of the work.